Grit Mobile: Data Collection App for the Grit Research Project

By Ryan Chang

Computer Science Department

College of Engineering

California Polytechnic State University

March 20, 2015

Contents

Abstract **iii**

1 Introduction **1**
1.1 Problem Description 1
1.2 Solution Description 2

2 Requirements **3**
2.1 Initial Registration 3
2.1.1 Login 3
2.1.2 Class Selection 5
2.1.3 Class List Viewing and Editing 6
2.2 Class and Non-Class Activities 8
2.2.1 Activity Viewing and Editing 8
2.2.2 Activity Entry 10
2.3 Activity Entry Push Notifications 13
2.3.1 Class Activity Notification 13
2.3.2 Non-Class Activity Notification 15
2.4 Database Querying 16

3 Design, Implementation, and Tools **17**
3.1 Design and Implementation 17
3.2 Architecture 18
3.2.1 Class Diagram 18
3.2.2 Database Schema 19
3.2.3 Example Code 20
3.3 Tools and Frameworks 22
3.3.1 Tools 22
3.3.2 Frameworks 23

4 Things That Went Wrong **25**
4.1 Uncompleted Features 25
4.2 Android SDK Issues 26

5 Future Improvements **26**

6 Conclusions **27**

References **29**

Abstract

This project consists of one part of a larger University research project aimed at determining if active learning builds the character trait of grit in students while they learn in the domain of mechanical engineering. The research project has three goals: the development of a mobile app to measure the quantity and quality of active learning, a model of student success using the quantity and quality of active learning, and a characterization of the relationship between active learning and grit growth. This senior project focuses on the mobile app as part of the larger research project. The mobile app will provide users with the means to submit research data via their mobile phones. The mobile app, named "Grit Mobile", is developed for the Android platform. The mobile app is written in Java, using the Android SDKs. Additionally, the Parse platform is used for backend data collection and user management. Most project requirements were implemented, but some more advanced features were not completed due to time constraints.

1 Introduction

Grit Mobile is an Android phone and tablet app which students use to submit research data to the University Grit Research Project. The research data collected can be broken down into two parts: Class Activity Data and Non-Class Activity Data. Class Activity Data can further be broken down into something called ICAP Data. ICAP stands for Interactive, Constructive, Active, and Passive. These categories are used to classify the type of learning activities students engage in while in a classroom or laboratory.

The Grit Research Project seeks to have students specify the number of minutes they spend on each learning activity during each class or laboratory period. Secondly, the researchers would like to collect information about time spent on the class outside of classroom or laboratory hours. Non-Class Activity is simply a student description of what activities, such as homework or studying, the students perform outside class hours. By collecting this Class and Non-Class Activity Data, the researchers hope to develop methods to increase student success in first-year Mechanical Engineering and Computer Science courses. Grit Mobile serves as the data collection portion of the Grit Research Project.

1.1 Problem Description

Since the Grit Research Project relies on student data, the project requires a way for the students to submit their data easily and frequently. The researchers determined that a mobile app would be the best method to collect student data. In

developing a mobile survey app for the Grit Research Project, the researchers seek to tackle several objectives.

The objectives that Grit Mobile intends to fulfill are:

1. Submission of Class and Non-Class Activity Data for a specific class section.
2. Automatic reminders to students to enter Class and Non-Class Activity Data promptly.
3. Provide a way for the researchers to collect and query student data entries.
4. App should be simple for students to use.

1.2 Solution Description

Grit Mobile intends to meet the previously described issues.

The main features of Grit Mobile are:

1. Students are given a registration code in person in order to create an account on the app. Grit Mobile does not handle registration in the Grit Research Project. The registration code ensures student privacy by only associating their code with their submitted data. Only one or two researchers will know the mapping from registration codes to student names.
2. Allows students to submit data for only their registered courses from a pre-populated list of classes.
3. After initial registration, students can select a class from a list and choose to submit either a Class or Non-Class ICAP Activity.

4. Push notifications are sent to devices with reminders to enter Class Activity Data immediately after the class period ends, and at configurable intervals after class for Non-Class Activity Data.

5. The backend at Parse.com provides a relational database for queries to be run against the data collected by the app.

2 Requirements

This section contains the different requirements for the project. Each section describes a part of Grit Mobile that contributes to the overall solution.

2.1 Initial Registration

2.1.1 Login

Upon launching Grit Mobile for the first time, students are prompted to register, as shown in Figure 1.

Figure 1: Grit Mobile Registration

By registering, students associate their entries with their anonymous user ID in the Parse backend, as well as register their device to receive push notification reminders to enter activity data. The information collected during registration may change in the future, depending on the requirements of the Research Project.

2.1.2 Class Selection

After successfully registering, students are prompted to select the classes they are in that are participating in the Grit Research Project, as shown in Figure 2.

Figure 2: Select Participating Classes

In order to simplify class input, students select from a pre-populated dropdown list of classes to choose from. Students using Grit Mobile are assumed to be registered in at least one of the pre-populated classes. By having a pre-populated list, students

do not have to manually enter information such as class number, section number, instructor name, and class times, for example. This method also prevents students from making errors when entering class information.

2.1.3 Class List Viewing and Editing

Students are then presented with a list containing the classes selected during class registration, as shown in Figure 3.

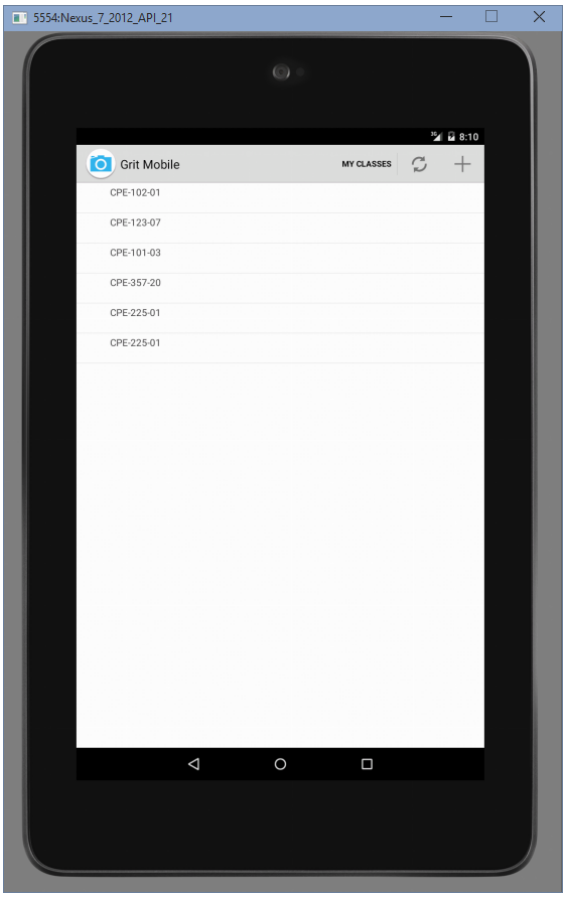


Figure 3: List of Selected Classes

In the top right corner, the "+" button will bring the student back to the Class Registration screen. By performing a long press on a Class Section, such as CPE-102-01, students can edit or delete the class from their account, as shown in Figure 4.

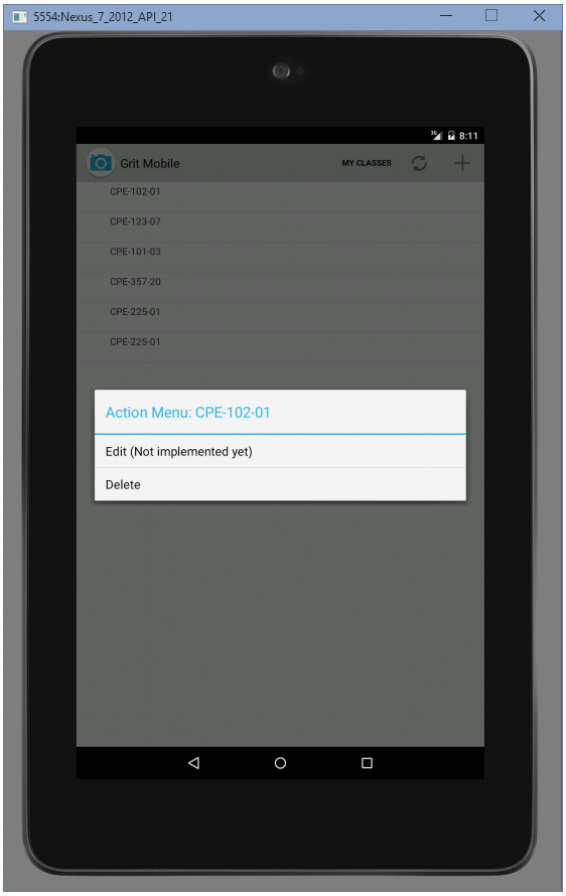


Figure 4: Edit Classes Dialog

By performing a quick single tap on a Class Section, students are brought to the Activities page for the selected Class Section, as described in the following section.

2.2 Class and Non-Class Activities

2.2.1 Activity Viewing and Editing

Upon single tapping a Class Section, students are brought to the list of submitted Activities for that Class Section, as shown in Figure 5.

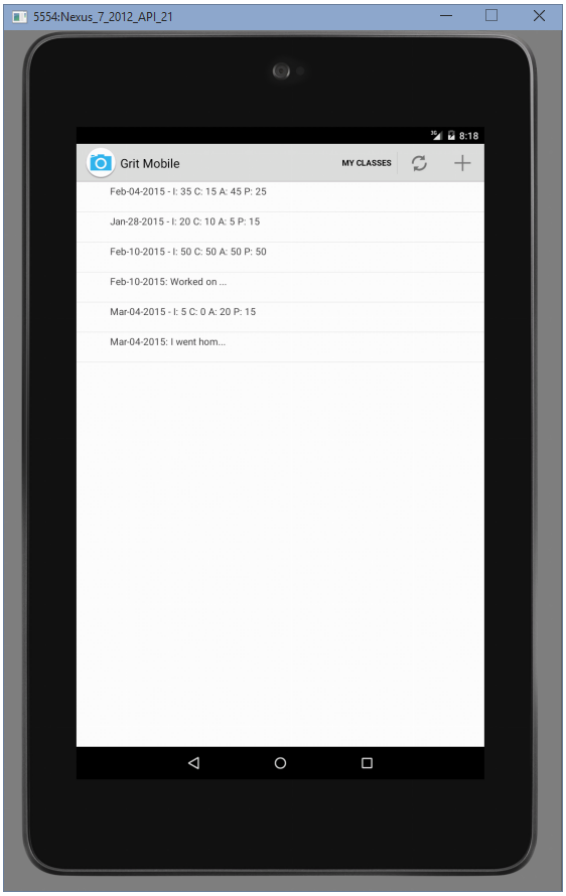


Figure 5: Viewing Class and Non-Class Activity Entries

The Activities list shows all Class and Non-Class Activities for the selected class. For Class Activities, the time entries for ICAP data is shown in minutes. For Non-

Class Activities, a truncated description of the Non-Class Activity is shown. By performing a quick single tap on a Non-Class Activity, students can see the full Non-Class Activity description entry, as shown in Figure 6.

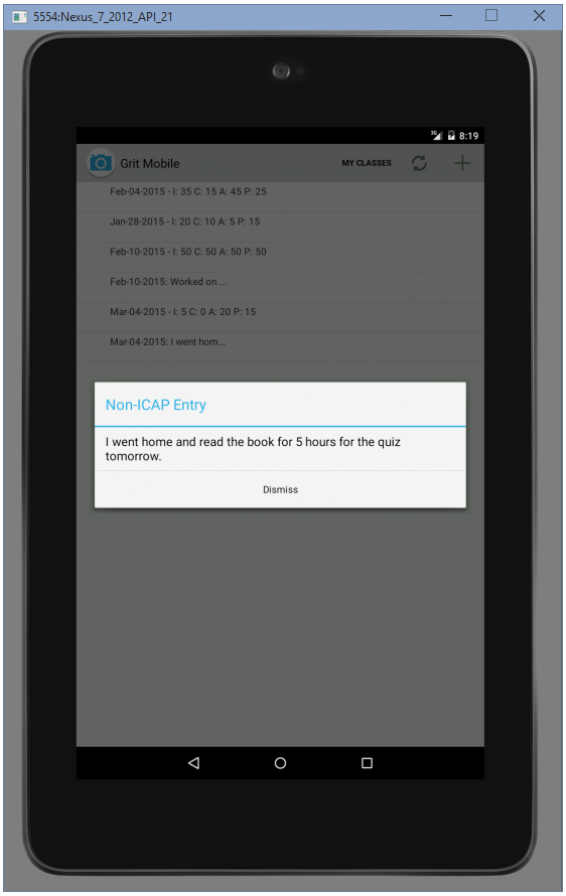


Figure 6: Full Non-Class Activity description

In the top right menu bar, the "My Classes" button returns students to the Class List. Figure 7 shows the "+" button, which on this screen will prompt students to select either a Class Activity or Non-Class Activity entry to add.

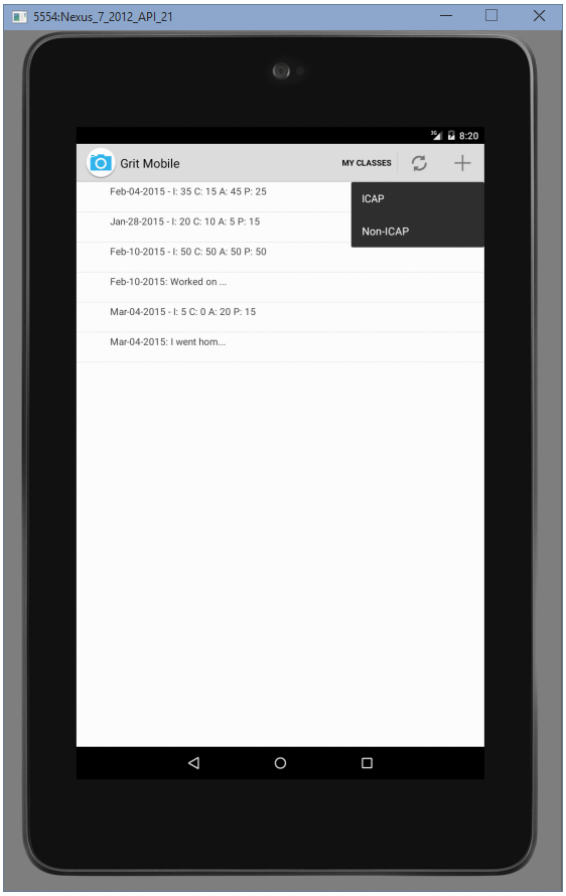


Figure 7: Add Entry Dropdown Menu

2.2.2 Activity Entry

2.2.2.1 Class Activity If Class Activity (ICAP) is selected, students are brought to an entry page, shown in Figure 8.

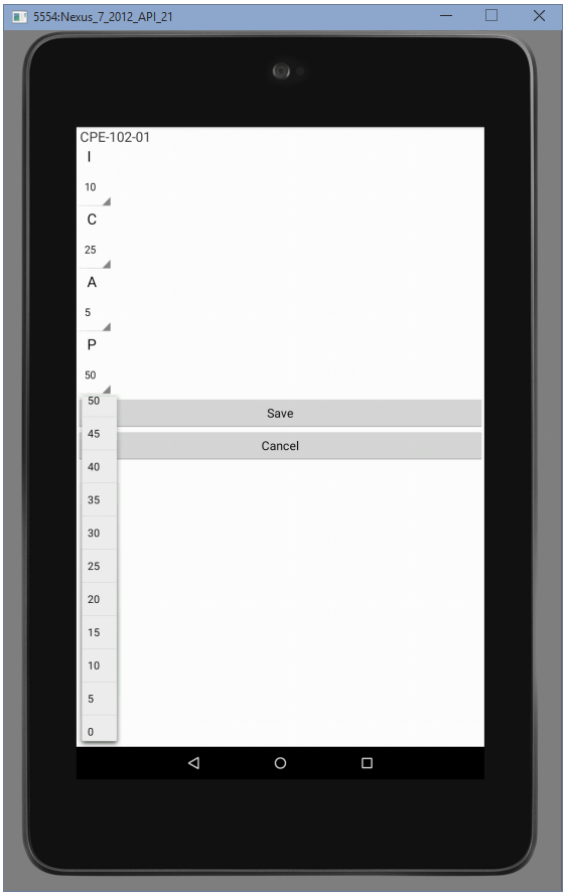


Figure 8: Class Activity Entry Page

On the Class Activity entry page, there are dropdown menus with minutes for each of the ICAP learning activity components. Students select from each dropdown menu the amount of time in minutes spent on each learning activity. After clicking "Save," the data is submitted to the Parse backend, and students are returned to the Class List. If there is no data connection, the app will cache the entry until a data connection is re-established. Checking for a data connection, data caching, and

data re-submission is handled automatically by the Parse API and is transparent to students.

2.2.2.2 Non-Class Activity If Non-Class Activity (Non-ICAP) is selected from the "+" dropdown menu, students are brought to the entry page shown in Figure 9.

Figure 9: Non-Class Activity Entry Page

Students simply enter a description of what their Non-Class Activity was, such as studying, reading the book, or working on homework assignments, for example.

Clicking "Save" will submit the entry to the Parse backend and return students to the Class List. Again, if there is no data connection, the app will cache and automatically re-submit the entry after a data connection is re-established.

2.3 Activity Entry Push Notifications

One of the main reasons for the development of a mobile app to collect research data is to remind students on their mobile devices to submit Class Activity data after the class period has finished, and also to prompt Non-Class Activity data entry at regular intervals after a class ends. This feature was not implemented due to time constraints but is included as a reference for how it would work.

2.3.1 Class Activity Notification

Shortly after a class ends, a push notification would be generated from the Parse backend and sent to only the students in a particular class. Figure 10 shows what the notification would look similar to.

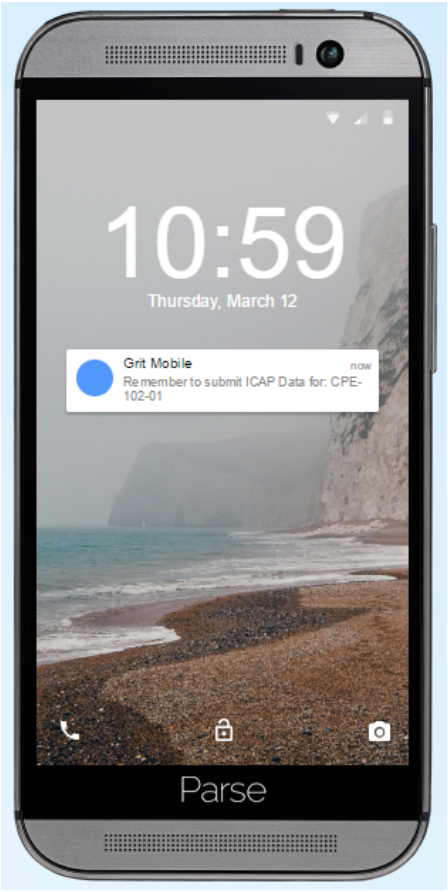


Figure 10: Class Activity Push Notification Example

The push notification reminds students to enter Class Activity data for the class that has just ended. Clicking on the notification would ideally bring students straight to the Class Activity page for the specific class, where they would enter the ICAP minute breakdowns.

2.3.2 Non-Class Activity Notification

During the evening, for example, the researchers would like to know what kind of activities students are doing for the class while outside class times. The app would ideally allow students to set an interval to be reminded to submit Non-Class Activity data. Figure 11 shows what the push notification would look similar to.

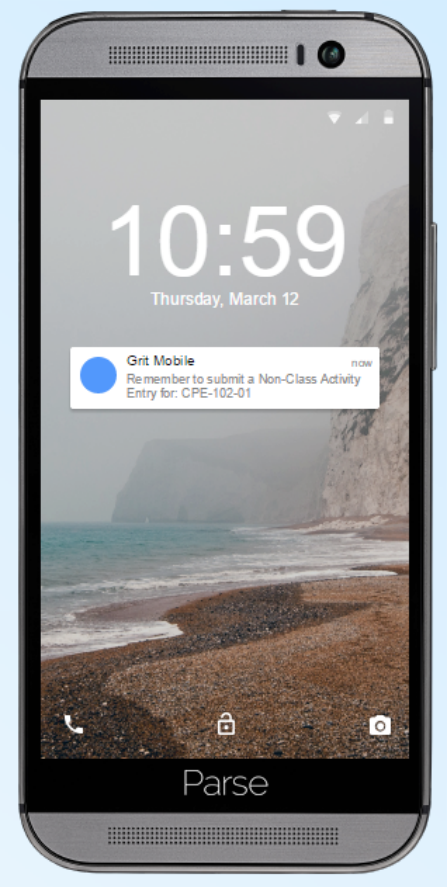


Figure 11: Non-Class Activity Push Notification Example

The push notification would resend every few hours at the interval specified, and bring students straight to the Non-Class Activity entry page for the specific class.

2.4 Database Querying

The researchers need a way to view and analyze the data collected from the students via the app. The backend at Parse.com provides a way to view and run queries on the data submitted. Figure 12 shows the Class Activity table.

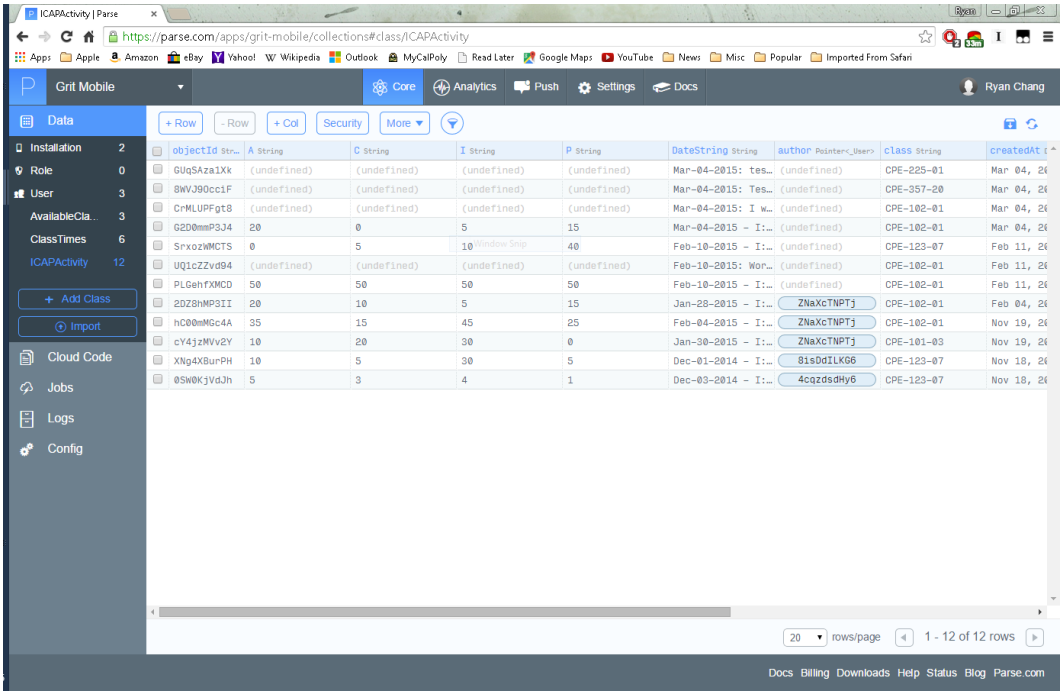


Figure 12: Class Activity Table

The Parse database is much like a relational database and contains columns, primary keys, and foreign keys. Queries are made by clicking the funnel option on the menu at the top, as shown by Figure 13.

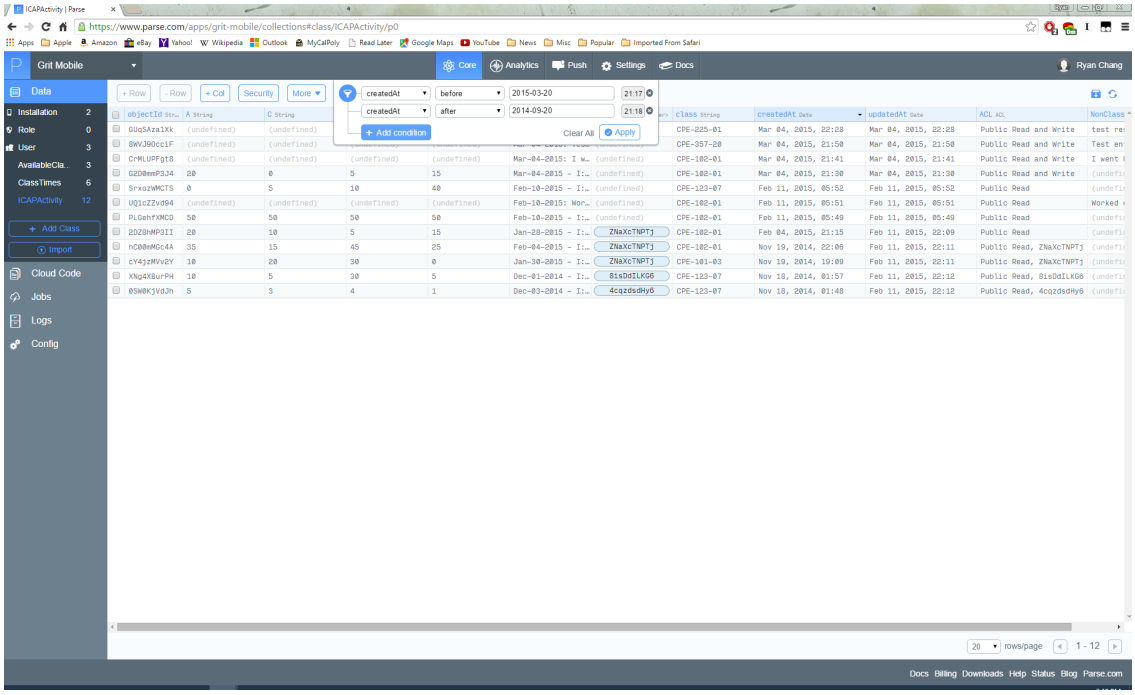


Figure 13: Query Options

After selecting a column, the Parse query interface changes the possible conditionals to query by. For example, if a column's type is "Date", then the Parse query interface will only allow date conditionals. The example query in Figure 13 selects Activity entry rows with submission dates between September 9, 2014 and March 20, 2015.

3 Design, Implementation, and Tools

3.1 Design and Implementation

The design of Grit Mobile was aided by two Parse example projects, which demonstrated how to interact with the backend, perform registration and login, and display

data in Android paradigms. The two example projects are called MealSpotting and ParseLogin.

3.2 Architecture

3.2.1 Class Diagram

The following diagram shows the classes of the Grit Mobile Android app:

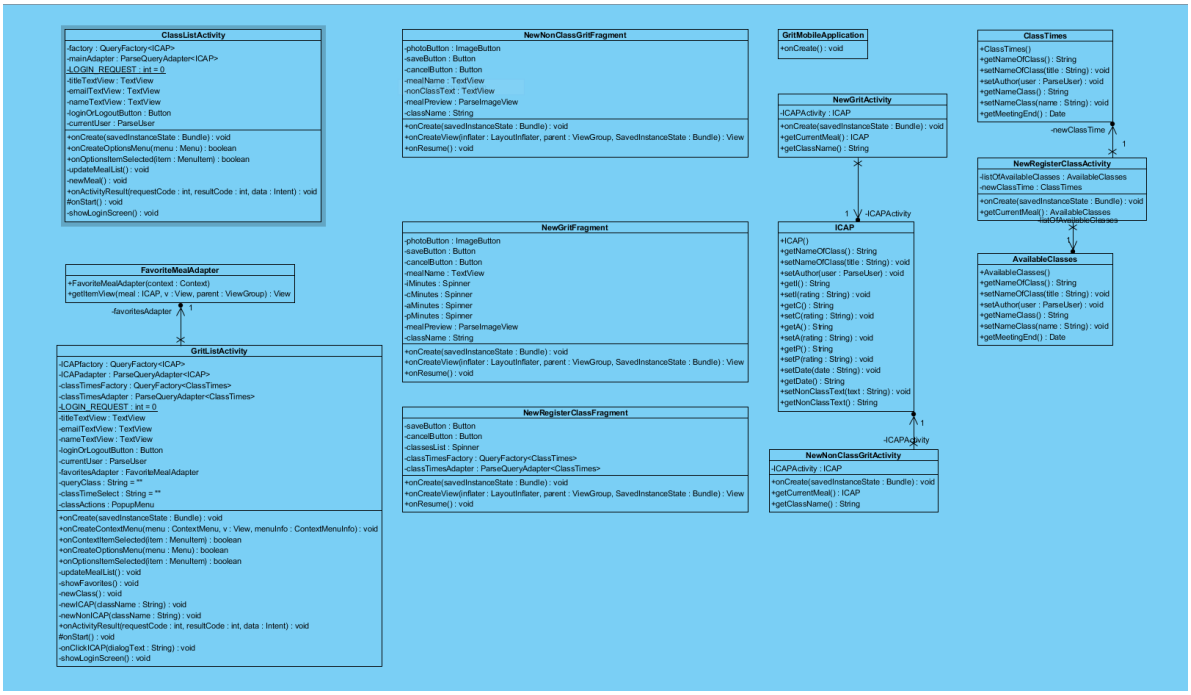


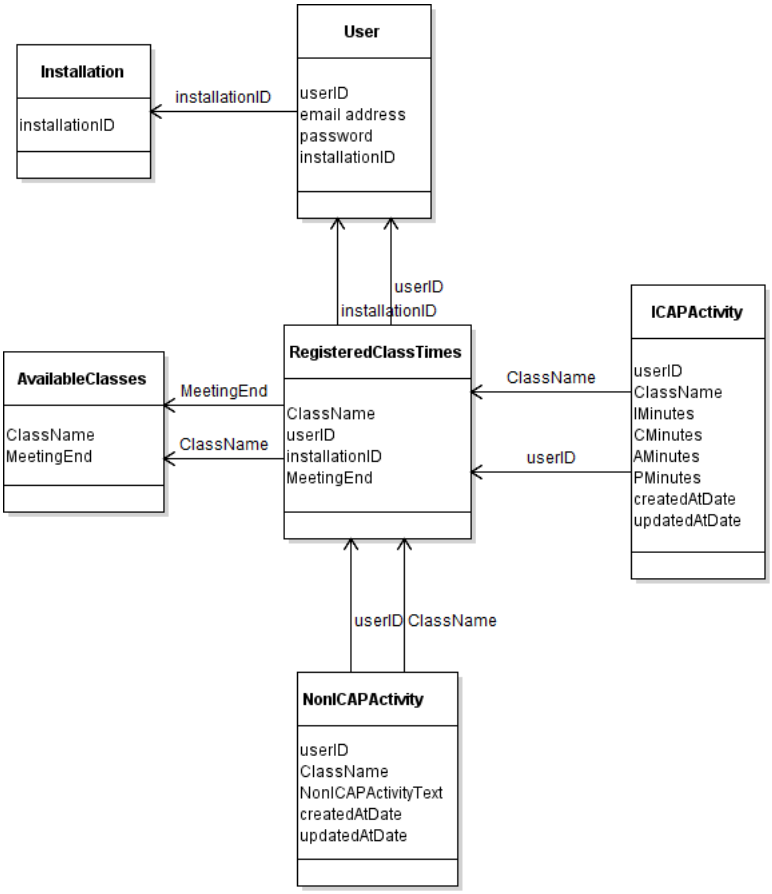
Figure 14: Class Diagram of Grit Mobile

The operation of Grit Mobile is handled through one main Activity, which then switches out fragments for the various functions. Login/Registration, Class, and

Non-Class Activities are all fragments called from GritListActivity. In GritListActivity, the ListAdapter is switched out based on whether the Class List or Activity List is currently in view.

3.2.2 Database Schema

The following diagram shows the basic schema of the backend database:



Arrows represent a relation between columns and tables (Parse calls this a "Pointer"). In every Parse database schema, the Installation and User tables come by default. The Installation table keeps track of the UUIDs of all the devices that install the

app. This table is used to send push notifications to devices. The User table is populated with information provided during the registration process and also associates a unique user ID with a device ID. This way, if students need to switch devices, push notifications can be sent to the new device once they log in again. The AvailableClasses table contains the list of classes that pre-populate the dropdown menu for students to select during registration. Currently, it contains the name of the class, as well as the time the class is over, in order to generate a push notification at that time. RegisteredClassTimes holds the list of classes students selected during the registration process. RegisteredClassTimes associates the classes selected with the students who selected them, so only classes a particular student selected will show up in the class list for that student. Additionally, push notifications will be sent to only those students actually in a given class. ICAPActivity and NonICAPActivity both have columns for the unique user ID of the student (to associate entries with a particular student), name of the class the entry is for, as well as the date the entry was created and updated (if any). The researchers would like to know how long the delay is between class ending and students entering Activity data. Additionally, ICAPActivity has fields for the ICAP minute entries, while NonICAPActivity has one field for the Non-Class Activity description text.

3.2.3 Example Code

3.2.3.1 Parse Integration Code Tables, such as ICAPActivity, can be defined and created in the app code using the Parse API, as well as be manually created in the Parse backend. However, in order to reference and update tables through the

app, a representation of the table must be defined in the app code. The code snippet in Figure 15 shows an example of how tables are defined. The `@ParseClassName`

```
@ParseClassName("ICAPActivity")
public class ICAP extends ParseObject {
  public String getNameOfClass() {
    return getString("class");
  }

  public void setNameOfClass(String title) {
    put("class", title);
  }

  public void setAuthor(ParseUser user) {
    put("author", user);
  }

  public int getI() {
    return getInt("I");
  }

  public void setI(int rating) {
    put("I", rating);
  }

  ...
}
```

Figure 15: Parse Database Table Definition

annotation specifies the table name. Then, setters and getters simply need to be defined for each column. The type will be inferred from the parameter. For example, the `rating` parameter in `setI` is an int type, so the column will be set as an int type in the table. Column names are set by passing a String into the first parameter of the `put()` method.

3.2.3.2 Fragment Swapping Code

The Class and Non-Class Activity entry screens are separate fragments with a helper activity, and contain the code to enter and submit Activity data. Figure 16 shows the code to call these activities. These

```
private void newICAP(String className) {
  Intent i = new Intent(this, NewGritActivity.class);
  i.putExtra("theClassName", className);
  startActivityForResult(i, 0);
}

private void newNonICAP(String className)   {
  Intent i = new Intent(this, NewNonClassGritActivity.
     class);
  i.putExtra("theClassName", className);
  startActivityForResult(i, 0);
}
```

Figure 16: Class and Non-Class Activity Methods

methods are called when students tap on the "+" to add either a new Class Activity or Non-Class Activity entry for a selected class. The String `className` contains the name of the class to add the entry under, and is passed into the activity using the `putExtra()` method. After students tap the "Save" button, the fragment and helper activity close and the app returns to GritListActivity.

3.3 Tools and Frameworks

3.3.1 Tools

Android Studio was the IDE used to develop Grit Mobile. Android Studio is based off of IntelliJ IDEA, but has built-in tools to manage the Android SDK, compile the

app, and run the app in the Android Emulator. Figure 17 shows the layout of Android Studio.

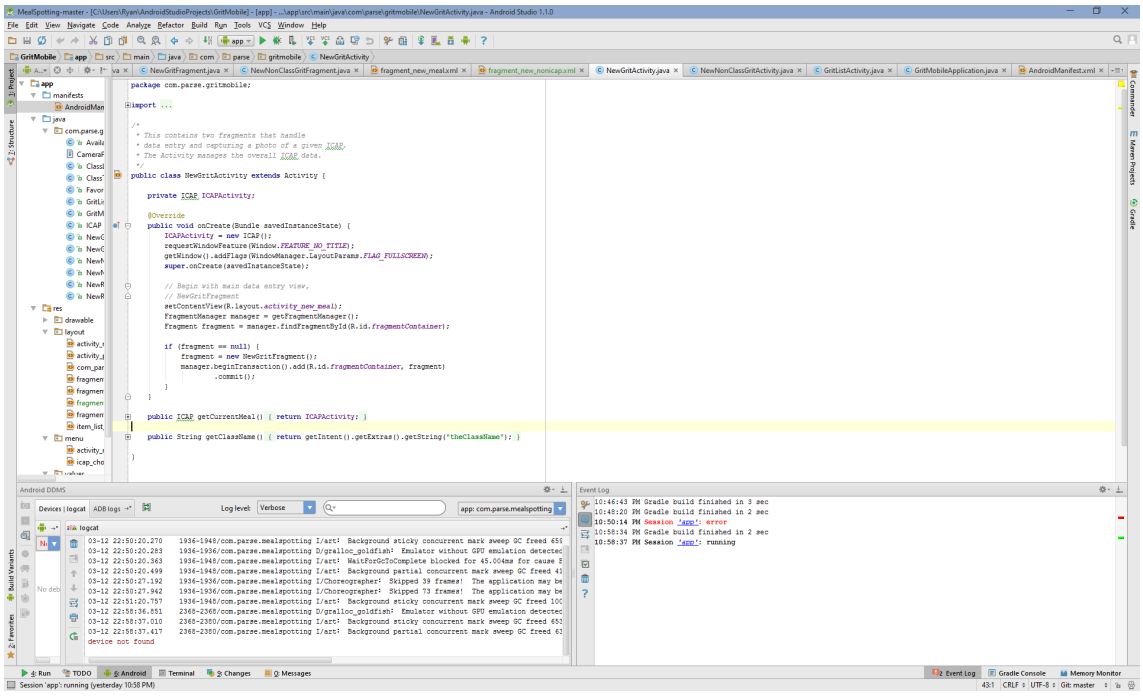


Figure 17: Android Studio IDE

The Version Control System used was Git, with Bitbucket as the repository. Additionally, a Samsung tablet was used for on-device testing.

3.3.2 Frameworks

The Android SDK is the primary method for developing apps on Android. In addition to the Android SDK, many features of Grit Mobile are handled by the backend, run by Parse at Parse.com. The Parse backend handles the database that

stores Activity entries, manages and registers users, and generates push notifications. Parse also has a feature called "Cloud Code," which is JavaScript code run on the backend. A "Cloud Code" script runs regularly and schedules push notifications to be sent to specific devices soon after the class period ends. A proof of concept "Cloud Code" script is included in Appendix A. Additionally, the Parse backend contains the database browser and basic analytics about app usage. Figure 18 shows the analytics panel displaying a chart graphing the number of API requests made by Grit Mobile throughout testing and development.

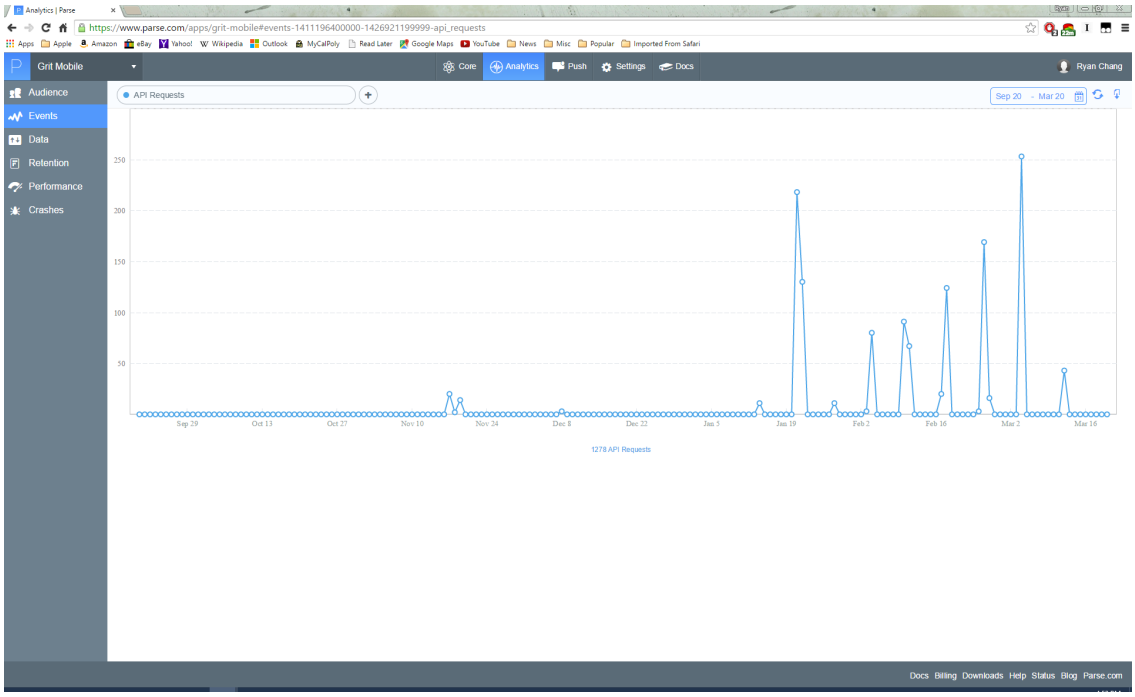


Figure 18: Parse Analytics Viewer Graphing API Requests

In another section of the analytics panel, Figure 19 shows the number of Class Activity Entries submitted throughout testing and development.

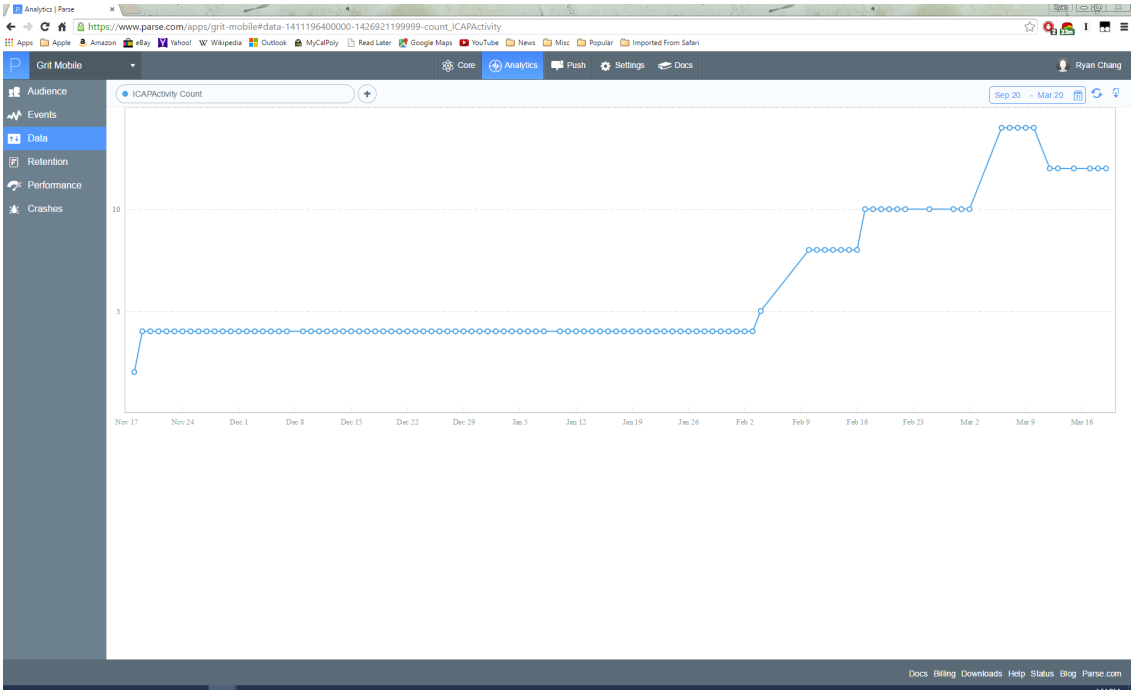


Figure 19: Parse Analytics Viewer Graphing Number of Class Activity Entries

4 Things That Went Wrong

4.1 Uncompleted Features

Several features were not completed due to time constraints.

- Push notifications are not implemented. The "Cloud Code" script exists as a proof-of-concept, but it currently does not send out push notifications when class periods end.

- There is no validation checking for input.

- There is no ability to submit an "empty" submission; for example, when class is canceled.
- In instances where class is canceled, it is a holiday, or the Cal Poly schedule changes for a day, the app should not remind students to submit a Class Activity entry.
- Editing classes or Activity entries is not implemented yet.

4.2 Android SDK Issues

Android Studio is a relatively new development environment. As a result, there are several bugs remaining. Android Studio uses a build tool called Gradle, which sometimes believed that dependencies were incorrectly missing. Furthermore, developing on a specific Android SDK version requires manually downloading all the required files for that SDK version, which can be a lengthy process. Additionally, the Android Studio syntax highlighter did not recognize some specific Parse SDK classes, although the project did compile successfully.

5 Future Improvements

For future improvements, the implementation of uncompleted features would be a high priority. Additionally, several more improvements can be made:

- General Refactoring: Remove all references to the example Parse projects that Grit Mobile is based on.

- UI Improvements: The UI is currently basic, and several UI quirks can be fixed. Multiple visual remnants of the example Parse projects remain. A graphics designer can be brought on to design a more aesthetically pleasing UI.
- App stability: The app crashes under certain circumstances.
- Deployment: Deployment of an Android app requires certain certificates and accounts to be registered with Google.
- Development of iOS and/or Windows Phone Grit Mobile apps: The Parse backend has an API for nearly all popular platforms, allowing a relatively easy development of Grit Mobile on additional platforms to utilize the existing backend. iOS would be the next platform to develop for.

6 Conclusions

This project has been an excellent learning experience in end-to-end development of a mobile app along with the related backend. While I did have prior Android development experience, it was from 5 years ago, and mostly outdated. I had no experience developing with the Parse platform, and this project provided an opportunity to develop using an established framework and backend. Implementation of all basic features, such as user management, class registration, and Class and Non-Class Activity event submissions, were completed.

The implementation of push notifications was a major goal for this project, and although it was not able to be implemented in time, the foundation is there and

would be straightforward to add given more time. Additionally, the Parse framework is very modular, and allows easy addition to the database schema. The current version of the app cannot be deployed for production use, since it has not been signed for deployment. The app also crashes under certain circumstances and could use a visual overhaul. Overall, however, all major features except push notifications were successfully implemented in the course of this project, with a very good foundation for continuing and finishing the project in the future.

References

[1] *Android SDK*

Available at: http://developer.android.com/sdk/index.html

[2] *Bitbucket Repository*

Available at: https://bitbucket.org/rchang05/gritmobile3/

[3] *Android Studio*

Available at: http://developer.android.com/tools/studio/index.html

[4] *Parse*

Available at: http://www.parse.com/

[5] *Parse SDK*

Available at: http://www.parse.com/docs/

Appendix

A "Cloud Code" JavaScript Proof of Concept

```
function scheduleNotifications()
{
  var classTable = Parse.Object.extend("ClassTimes");
  var parseAttr = Parse.Object.extend("ClassName");
  var pushQuery = new Parse.Query(Parse.Installation);
  pushQuery.equalTo("day",day);
  pushQuery.exists("deviceToken");
  var classQuery = new Parse.Query(classTable);
  var classObject = classQuery.first();
  var className;

  classQuery.get(request.params.ClassName, {
    success: function(object) {
      className = object;
    },

    error: function(object, error) {
      response.error("Error retrieving class")
    }
  });
```

```
var promise = new Parse.Promise();

Parse.Push.send({
  where: pushQuery,
  data: {
    alert : "Remember to enter ICAP data for " +
       className
  }}, { success: function() {
    // Push was successful
  },
  error: function(error) {
    // Push was unsuccessful
  }}).then (function(result){
    //Marks this promise as fulfilled,
    //firing any callbacks waiting on it.
    promise.resolve(result);
  }, function(error) {
    //Marks this promise as fulfilled,
    //firing any callbacks waiting on it.
    promise.reject(error);
});
```

```
  return promise;
}

Parse.Cloud.job("schedulePushes", function(request, status
   )
{
  var promiseArray = [];

  promiseArray.push(scheduleNotifications());

  //Returns a new promise that is
  //fulfilled when all of the input promises are resolved.
  Parse.Promise.when(promiseArray).then(function(result) {
    console.log("successful push")
    status.success("successful push");
  }, function(error) {
  console.error("Push Error: " + error.message);
  status.error(error);
  });
});
```