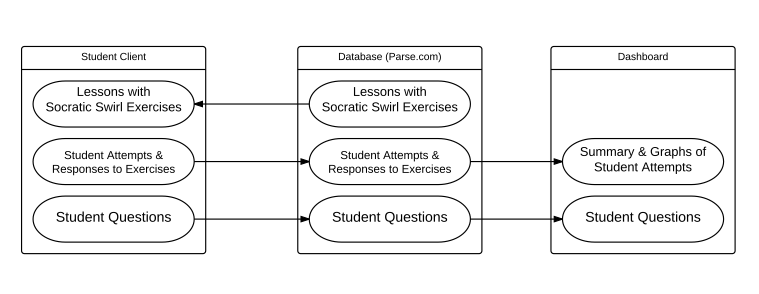
Introduction

Socraticswirl, developed on top of the existing swirl, lets instructors of the R programming language offer in-class, interactive programming exercises and lets the instructors view student answers and progress in real-time. The package also lets instructors manage student lists and course lessons in their own instance of the Socraticswirl application database, and launch a dashboard to watch their students' progress.

Socraticswirl has three major components, the student software, a parse.com database, and the instructor software, i.e. the dashboard and course management utilities.



[A] Steps to Setup Socraticswirl

1. Register an account at parse.com for each group that will be independently using Socraticswirl. Using that account, create two parse.com applications for each class, one for test and the other for production. Obtain Application Keys and REST API Keys for both apps. *[D]*
2. Perform the following steps, first for test and then for production copies of the software.
   1. Install Socraticswirl software
      1. The student software https://github.com/dimagor/socraticswirl
      2. The instructor software https://github.com/dimagor/socraticswirlInstructor
      3. Configure the software using the appropriate pair of keys. *[B]*
   2. Load course information into the parse.com databases
      1. The list of students, with precept assignment if any. *[F]*
      2. The courses and lessons. *[E]*
   3. Setup a Shiny web server with an authentication and/or authorization mechanism for the dashboard and run the shiny server. *[G]*
   4. Launch the dashboard on the Shiny server and access the dashboard at [*http://s.univ.edu/*](http://s.univ.edu/), where *s.univ.edu* is the shiny web server.
   5. Test … both the student and instructor sides.
3. Revise the instruction for students to include your method for distributing the parse.com keys, and any other local modifications. Find a secured way to give each student the application key, the api key and his/her id to be used in Socraticswirl for the class. *[C]*
   1. Student needs to initialize the Socraticswirl software once on each of their personal computer accounts.
   2. On shared computer accounts, students may need to initialize more than once, as multiple students use the same account.

Note:

(1) As the software is currently configured, the above steps need to be done for each class in a given semester, but all of the applications can be created by the same parse.com id.

(2) Whenever there is an update of code or courses or lessons, it is always recommended to try them on the test server before moving it to production.

Usage:

**Please make sure that the configuration files parse\_keys.R and server\_instance.R under the subdirectory "data" is updated accordingly.**

The dashboard application runs in the directory where server.R and ui.R are.

To use the dashboard, you may do the following: \* load packages socraticswirlInstructor and shiny \* change working directory to where server.R and ui.R are.

If you are using RStudio, this will start the dashboard application on your computer to show your students' progress and answering activity in real time:

library(socraticswirlInstructor)

library(shiny)

setwd(paste0(getwd(), "/inst/dashboard")) # the directory where server.R and ui.R are

runApp()

If you plan to running R at the command line, please change working directory to where server.R and ui.R are. In the socraticswirlInstructor package, it is the subdirectory inst/dashboard. Once you start the R session, you may do the following

library(socraticswirlInstructor)

library(shiny)

runApp()

To view the dashboard via web browser, **please make sure that the configuration files parse\_keys.R and server\_instance.R under the subdirectory "data" is also updated accordingly**, in addition to parse\_keys.R. You need to specify if it is the test server or production server.

Then, you may launch the dashboard on the Shiny server. Once the Shiny server (s.univ.edu) runs, instructors and preceptors can access your Socraticswirl dashboard at the following web address:

http://s.univ.edu/

We recommend that two servers are set up, one for test and the other for production. The production will be the one to monitor students' progress on the exercises for the class.

[B] Software Configuration

Socraticswirl uses pasrse.com to store all the data.

* Student data – either manually input at parse.com web interface, or use the provided utility program **uploadStudents.py** *[F]*
* Course data – upload automatically by the provided the R function **upload\_course("path to course")** *[E]*

The parse.com application key and REST API key may be retrieved at the parse.com web interface. *[D]*

With the keys obtained from parse.com, the configuration includes:

1. Socraticswirl instructor dashboard
   1. Go to the subdirectory **data**, and update the **parse\_keys.R** file with the appropriate (test or production) parse.com keys
   2. Go to the subdirectory **data**, and update the **server\_instance.R** with instructor account and password you created by socratic\_swirl\_signup(). *[E]*
      1. Please also fill in the server\_instance for either test or production.
2. Socraticswirl student side
   1. Each user needs to run the R function socratic\_swirl\_init with the correct student ID, the application key, the REST API key, and instance identifier.
      1. The default value of instance is “prod”, which means the production server. For the test server, the instance identifier is “test”.
      2. For student ID, we recommend using student emails.
      3. The keys may be distributed via e-mail or posted at blackboard so that students can cut and paste into their socratic\_swirl\_init command.
   2. The information will be stored in a RDS file for future use so the initialization process is only needed once for each personal computer account. However, for rare cases where several students share a same computer account, the students may need to initialize again if the software has been initialized by other students after his/her previous use.

The above needs be performed individually for test and production. For test, preceptors or testers may pretend to be students to make sure the whole system works. The production is for the class.

[C] Instruction to Students

This section assumes that you already have the application key, api key and student IDs for each students in the class. *[D]*

Courses need to be uploaded to the parse.com database. *[E]*

The following is a sample introduction to the students. Please feel free to revise as needed.

**1 Install devtools**

Once R and RStudio have been installed in your computer, the next step is to install SocraticSwirl. First, we need to install the R package devtools and the database software rparse.

• In RStudio, type:

install.packages("devtools")

• Similarly to install rparse:

devtools::install\_github("dgrtwo/rparse")

**2 Install SocraticSwirl**

The final step will be to install the R package SocraticSwirl. For that, type:

devtools::install\_github("dimagor/socraticswirl", ref="socraticswirl")

**3 Initialize SocraticSwirl “Once”**

After you receive the parse.com application key, REST API key and your instructor’s ID from your instructor, you need to initialize the SocraticSwirl software once in R or RStudio.

Suppose you already have the application key, api key and id.

appkey <- “cut-and-paste your application key”

apikey <- “cut-and-paste your api key”

id <- “your student id”

instructor <- “your instructor’s id”

To initialize your copy of SocraticSwirl, please type:

library(socraticswirl)

socratic\_swirl\_init(id, appkey, apikey, instructor)

You should see "Initialization completed.", which indicate the initialization process succeeded.

You only need to initialize Socraticswirl once for each computer account, as long as the keys and the instructor are not changed.

*However, if you are using a shared computer account (i.e. several people share the same account, which is rare but could happen), it is possible that you may need to initialize Socraticswirl again, as others may also initialize the software of this shared account after your previous use.*

**4 To Run a Lesson and the First Assignment**

After a successful installation of Socraticswirl, we will be ready to run the first swirl lesson for Chapter 1 in the book.

library(socraticswirl)

socratic\_swirl("INTRO", "INTRO1", "your id")

start()

When you are done, try the second lesson.

socratic\_swirl("INTRO", "INTRO2", "your id")

start()



In general, to run a different lesson, change the first two arguments of the socratic\_swirl() call. For example, to run the second lesson in the Causality chapter:

socratic\_swirl("CAUSALITY", "CAUSALITY2", "your id")

start()

If you want to interrupt your swirl session to perform a different task in R, you can type:

## you will be able to perform tasks outside the lesson

play()

## will resume swirl's lesson at the point where it was interrupted

nxt()

For multiple choice questions, enter 0 to exit the question and then use play() and nxt() to experiment in R. At times you may see R warnings on the screen as code is evaluated but you will still be in the Swirl environment and can continue with the lesson as usual. If you have any questions, please do not hesitate to contact any of the preceptors.

[D] Parse.com database

To create parse.com databases, please visit <http://www.parse.com>

Create an account for each group that will be independently using SocraticSwirl.

Create two new applications, one for test and one for productions. The app names can be whatever you’d like.

Select these types:

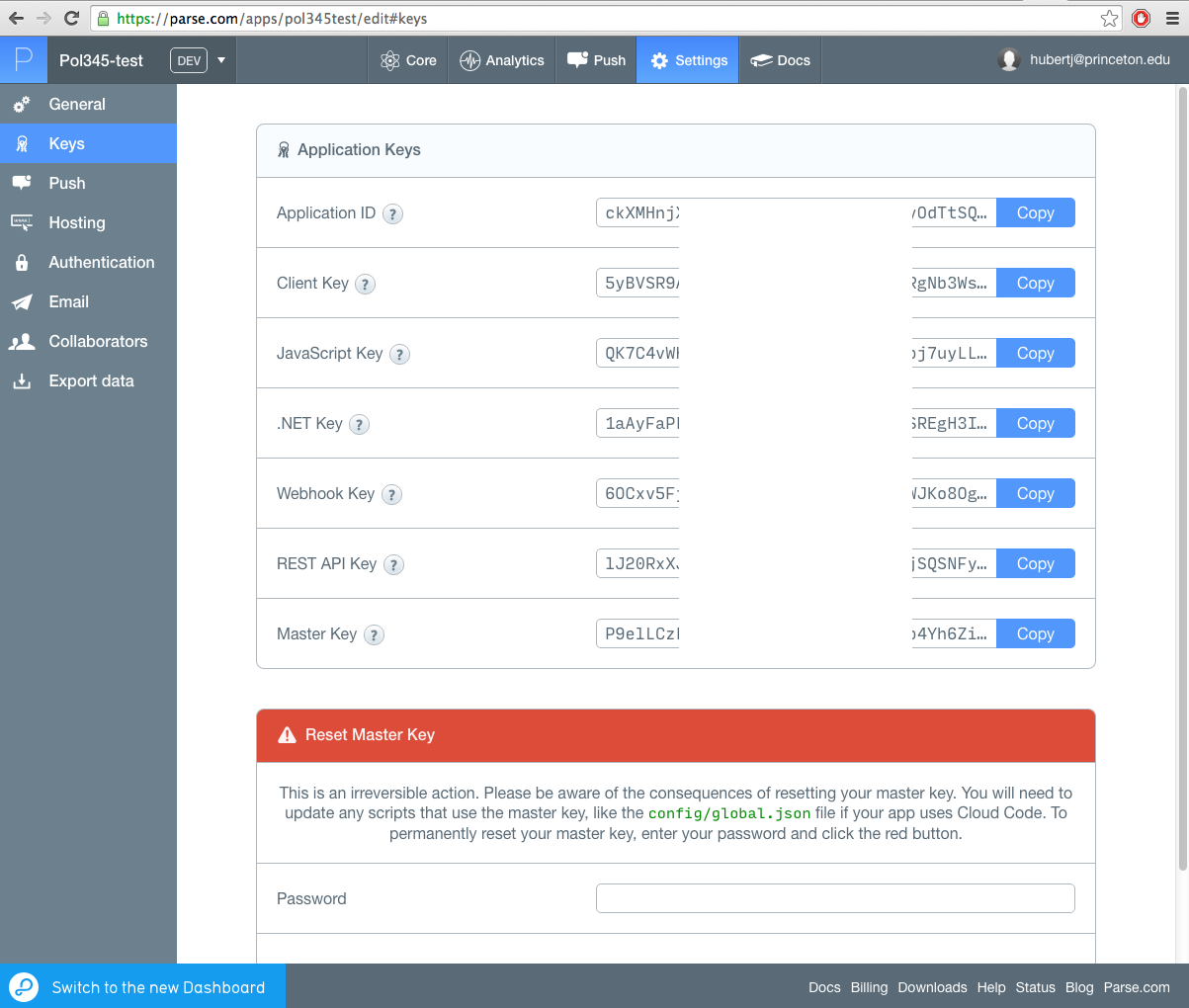
Select product: Data

Select environment: Web

Look in Settings > Keys. You will need the Application ID, REST API Key, and Master Key.

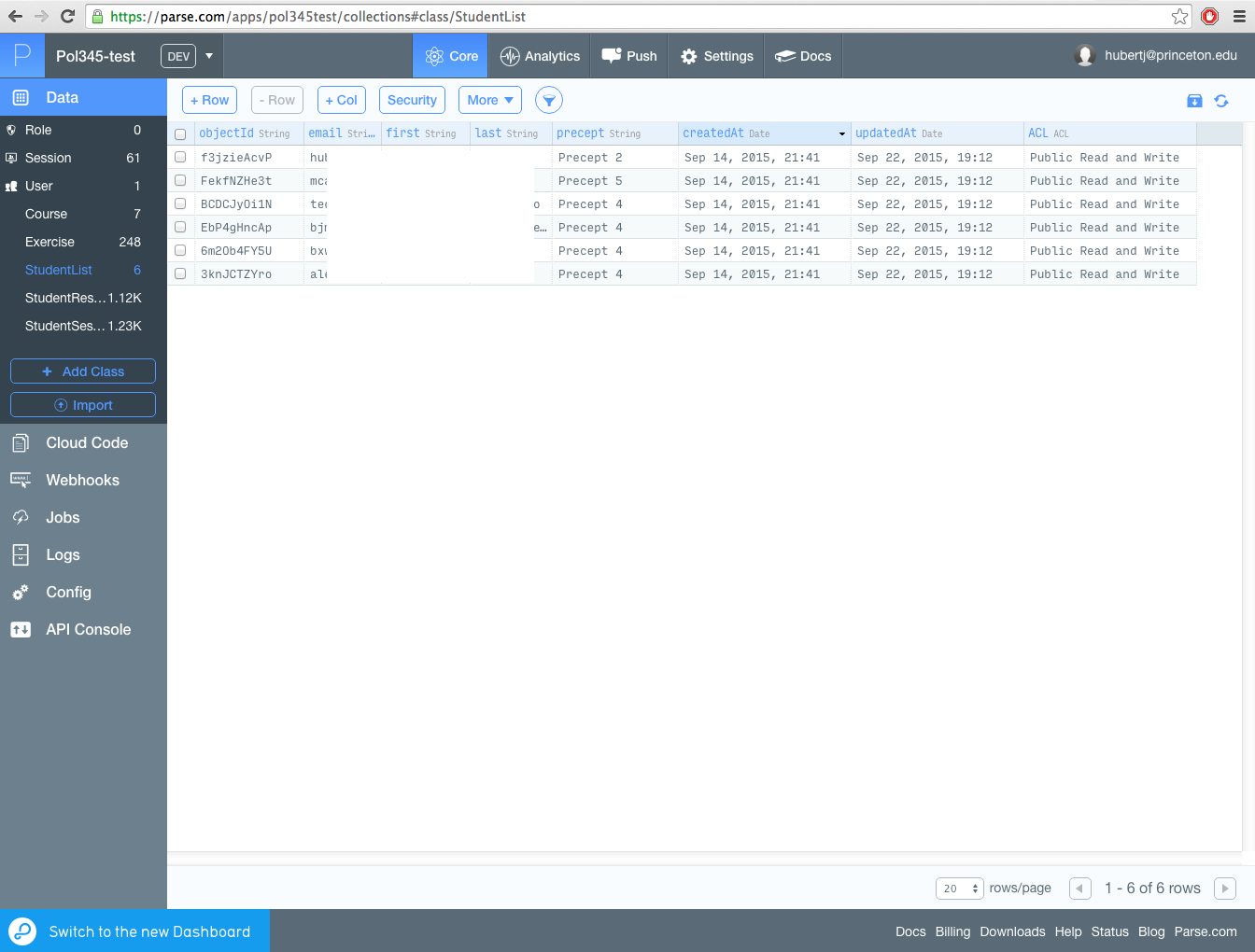
Instructor or admin can always log on at parse.com to see the information of the test and production servers.

For example, here are the keys for a sample database:



Application key and REST API key are needed in configuring the software, both the student side and the instruction side (dashboard). In addition, the Master key is needed in configuring the utility program **uploadStudents.py**.

Parse.com provides a GUI web interface to access database, for the students, class lessons and etc. In addition to the application and API keys, the master key is also needed to configure uploadStudents.py, a utility program to upload or update student information at the parse.com database.



In addition to use **uploadStudents.py** to *add or change* the list of students, the instructor or admin may also *add or change* student record individually. This may happen when a student assigned to a different precept or drops from the class (assigned to dropped).

[E] Upload a course

There are swirl courses and/or lessons available at <https://github.com/kosukeimai/qss-swirl>.

To get the course from the GitHub:

git clone <https://github.com/kosukeimai/qss>

For socraticswirlInstructor, i.e. the instructor side software, the following steps in R are needed to upload courses to the parse.com database.

1. The first time you upload a course to either to the test or production server, you need to create an instructor name and password as follows:

library(socraticswirlInstructor)

# For the test instance:

socratic\_swirl\_signup("<instructor name>", "<instructor password>", "<instructor email>", instance="test")

# or, for the production instance:

socratic\_swirl\_signup("<instructor name>", "<instructor password>", "<instructor email>")

1. After that, you may upload the courses using the instructor names and passwords as follows:

library(socraticswirlInstructor)

# For the test instance:

socratic\_swirl\_instructor("<instructor name>", "instructor password", instance="test")

# or, for the production instance:

socratic\_swirl\_instructor("<instructor name>", "instructor password")

upload\_course("/path/to/qss/swirl/INTRO")

upload\_course("/path/to/qss/swirl/MEASUREMENT")

upload\_course("/path/to/qss/swirl/CAUSALITY")

upload\_course("/path/to/qss/swirl/DISCOVERY")

upload\_course("/path/to/qss/swirl/PREDICTION")

…

[F] Upload the student list.

The Python program uploadStudents.py is used to upload a student roster to the parse.com databases.

In uploadStudents.py replace:

<parse.com application ID>

<parse.com REST API key>

<parse.com master key>

with the keys from parse.com. The program is set up for two courses, referred to as “course1” and “course2”, and each has a test and a production database. You need only use one of these.

To view the help text, use the “-h” flag:

uploadStudents.py –h

To create the database schema (the first time) and upload a student roster, type the following:

uploadStudents.py –create –filename student\_roster.txt –add –n course1 –i test

This would upload the student list for course1 into the test database. student\_roster.txt should be a tab-separated file. (You can save a tab-separated file in Excel by saving as a csv file. Be sure not to save the file with quoted fields.) The expected format is one line for each student, as follows:

lastname <tab> firstname <tab> email <tab> precept

[G] Dashboard web server setup

Install the following Linux packages if they are not already installed:

R

Development version of curl library

Install the Shiny web server:

Here are three options for installing shiny and instructions:

<https://www.rstudio.com/products/shiny/download-server/>

<https://github.com/rstudio/shiny-server>

http://rstudio.github.io/shiny-server/latest/

Clone the SocraticswirlInstructor code to the directory where it will be hosted. The directory should be specified for shiny in the configuration file: ‘/etc/shiny-server/shiny-server.conf’ as the ‘site-dir’.

git clone https://github.com/dimagor/socraticswirlInstructor

Thus, an instance of shiny-server is ready.

Install the following R packages:

install.packages("devtools")

install.packages("ggplot2")

install.packages("shiny")

install.packages("shinydashboard")

install.packages("dplyr")

install.packages("tidyr")

devtools::install\_github("dgrtwo/rparse")

Even though the code is already in the ‘site\_dir’ you still have to do this installation:

devtools::install\_github("dimagor/socraticswirlInstructor")

Put the Application ID and REST API Key and the instructor ID and password used to upload the course [E] in the parse\_keys.R file.

[H] Dashboard

To use the Socraticswirl dashboard, you'll have to log in first:

socratic\_swirl\_login("your\_name", "your\_password")

Once you've logged in, you can access your Socratic Swirl dashboard with:

dashboard()

This will start the dashboard application on your computer to show your students' progress and answering activity in real time.

To view a demo, try:

dashboard(demo = TRUE)

To view the dashboard via web browser, you need to launch the dashboard on the Shiny server. Once the Shiny server (s.univ.edu) runs, instructors and preceptors can access your Socraticswirl dashboard at the following web address:

http://s.univ.edu/

We recommend that two servers are set up, one for test and the other for production. The production will be the one to monitor students' progress on the exercises for the class.

1. Dashboard Features



* Exercise Dashboard - Information on student answers on an exercise.
* Lesson Overview - Graphical summary on exercises within lessons.
* Submitted Questions - To be implemented.
* Student Dashboard - Detailed exercise statistics per student on a variate of measurements.
* Response Details - Keep track all the students' answers. The records are searchable.

Note:

(1) The information may be grouped by Course, Lesson, and Precept.

(2) Refresh button will reload the up-to-date statistics from parse.com.

2. Exercise Dashboard



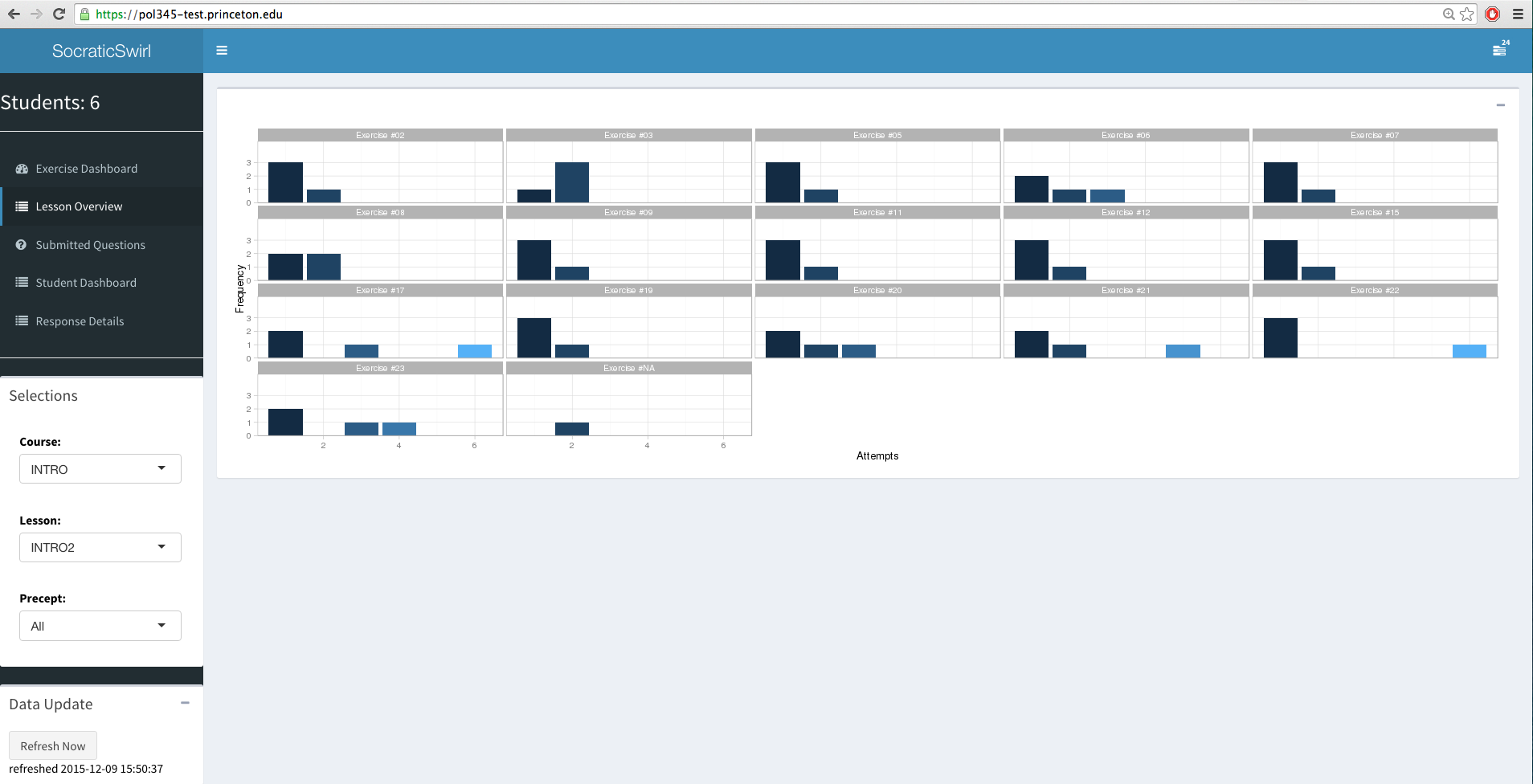
Per each exercise in the lesson/course, the following ststistics are provided for the students within the same selected Precept:

* The percentage of students attempted and completed as of this moment.
* Attempt Frequency - statistics on the number of attempts per student.
* Progress Tracking - statistics on the attempt/complete progress by time.

It also displays the exercise prompt and the correct answer. This information may be used to understand students' area of weakness in conjunction with the incorrect answers and the type of common errors that the dashboard collected and displayed for the exercise and the precept group.

3. Lesson Overview

It provides a sequence of histogram plots to show students attempts on all the exercises in the lesson for the selected precept group.

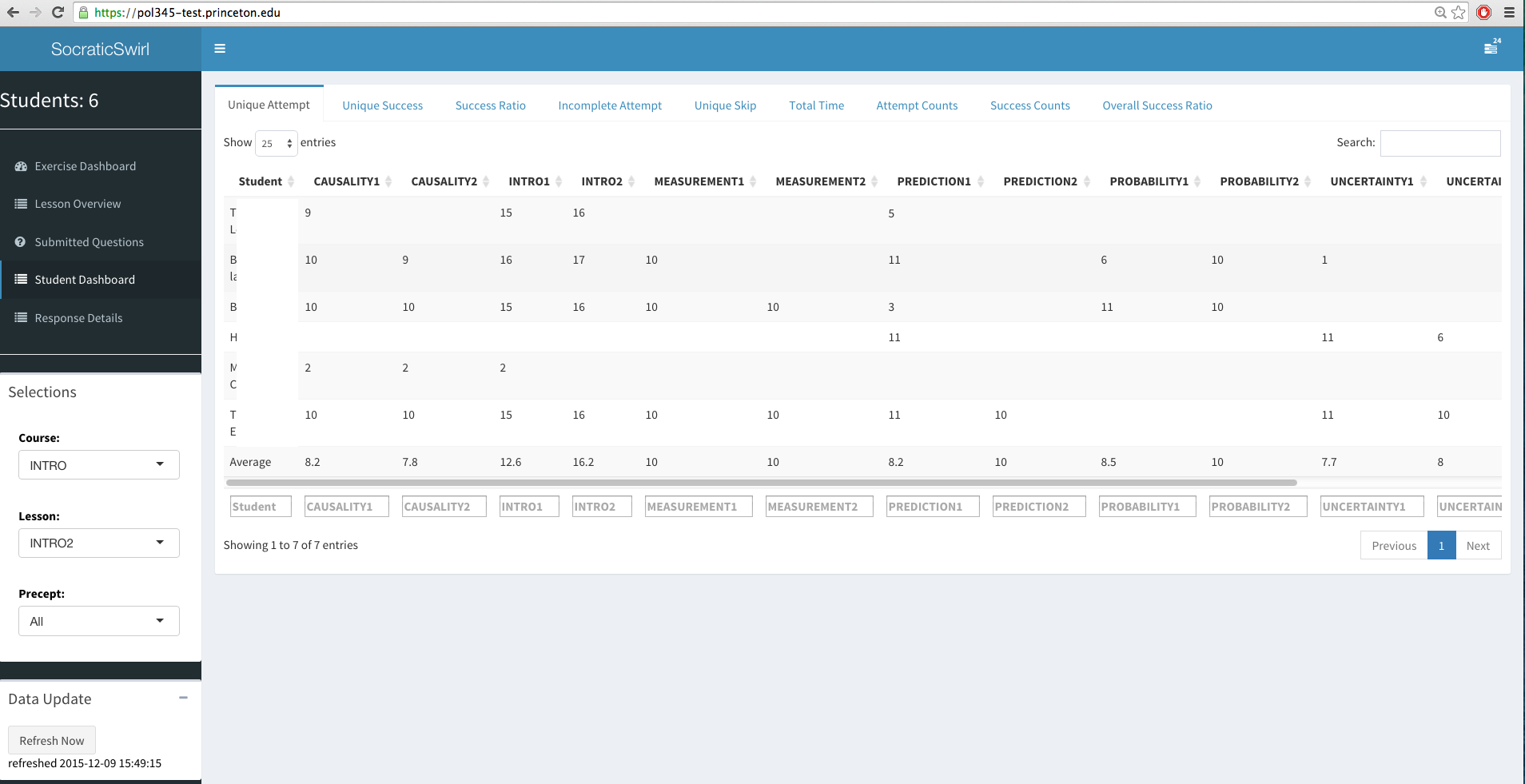


4. Submitted Questions

This field is intended for displaying Q&As between the students and instructors/preceptors.

5. Student Dashboard

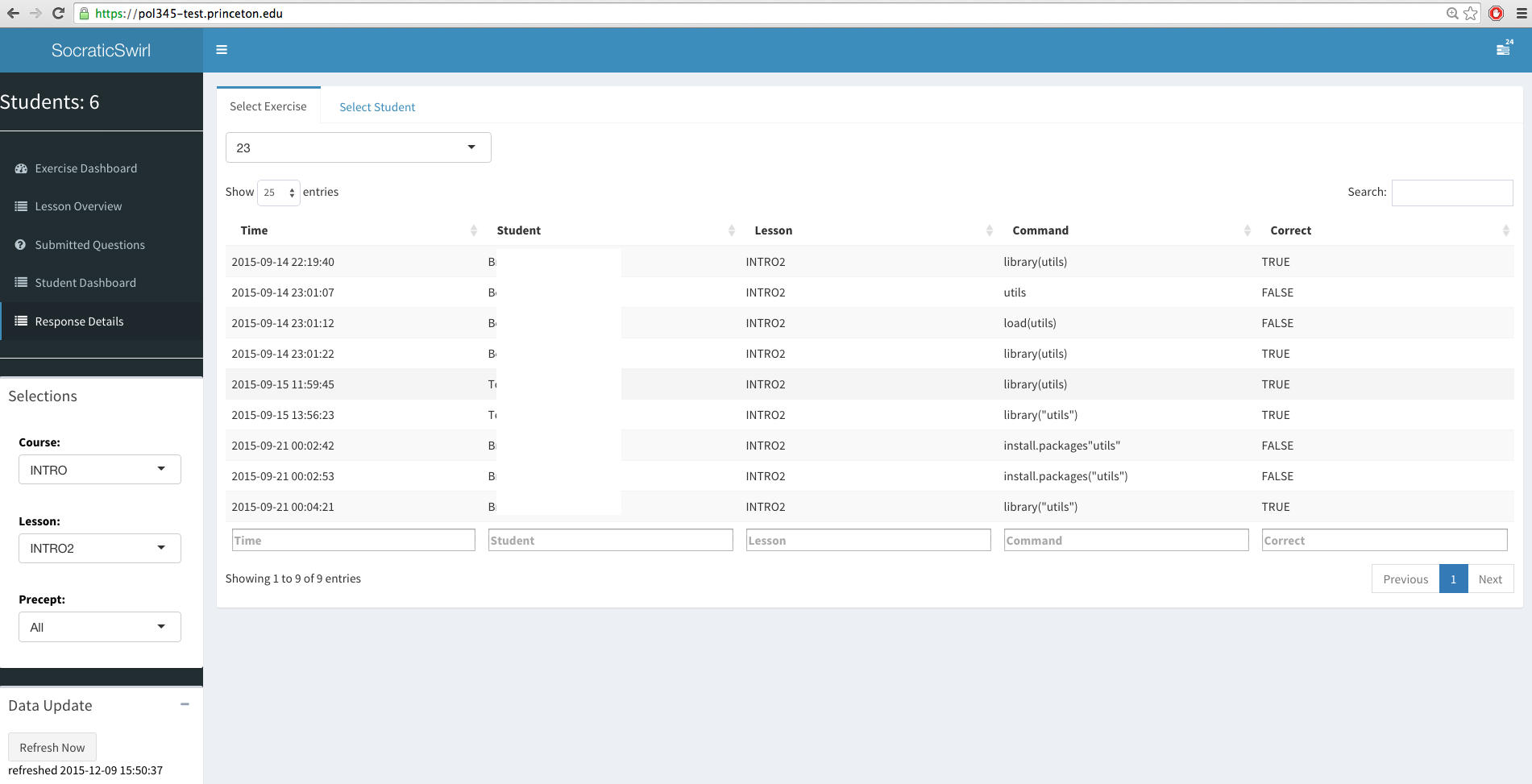
A list of tables is provided to show the progress of each student in the selected precept group on the lessons.



These tables are:

* Unique Attempt - the number of exercises answered by the student, with duplications ignored.
* Unique Success - the number of exercises answered correctly by the student, with duplications ignored.
* Success Ratio - "Unique Success" over "Unique Attempt", as defined above.
* Incomplete Attempted - the number of exercises answered, but never correctly answered. Exercises never attempted are excluded.
* Unique Skip - the number of exercises never attempted by the student, with duplications ignored.
* Total Time - the time elapsed between the first and last encounters of any exercise in the lesson.
* Attempt Counts - the number of exercises answered by the student, with duplications included.
* Success Counts - the number of exercises answered correctly by the student, with duplications included.
* Overall Success Ratio - "Success Counts" over "Attempt Counts", as defined above.

6. Response Details



All the student answers are kept per exercise for the selected lesson/course precept group. The fields are “Time”, “Student” (Name), “Lesson”, “Command” (the student answer) and “Correct” which indicates if the answer is right.

The student answers may be displayed by selecting corresponding exercise or student name for quick access to the intended information.

The student answers are also searchable, as long as the search string matches any fields in the records. Searching within a field is also possible. All matches are case insensitive.