LiveClass A/B Test Detail

We **avoid biasing our users in the testing process by letting two different users to perform either the student's side or the instructor's side tasks, making sure they do not have prior knowledge on either in advance. In addition, we only provide general instructions without giving a step-by-step hint when they perform the tasks, and do not introduce what they should be expecting under each screen, by only telling them the goal of this app and each process.*

Methods(instructions):
Step 1: Setup
The very first step is to ensure our app is in the latest updated version and open the stimulator before testing. Make sure the app works fluently to promote an effective user test.

Step 2: Inform the user about this test
Script:
(say before conducting the testing)
"Thank you for volunteering to test our application, LiveClass. We highly appreciate your time and effort. You will be randomly assigned to the role of a student or an instructor that uses our APP. .During the test, please feel free to give instant feedback by thinking aloud. Tell us everything you thought about while completing the tasks that will be given to you later. We are open to any comments on your product. However, we will not clarify your confusion during the test. Instead, we want you to try figuring out the functionalities by yourself. We could answer your question later when tasks are finished.
Please read and sign this consent form before beginning. Let me know if you have any concerns."
(Participants read and sign the form http://ixd.ucsd.edu/home/w17/assignments/A7/consent.pdf)
"Thank you for signing the consent form, we will now begin testing. Please remember to think out loud."

Step 3: Execution
We will show the stimulator on our screen to the user and introduces their tasks as the following:

Tasks:
Student side:
We would like the student users to experience most of our functionalities. Following is the two main task followed by several specific tasks that they need to complete:

1. Create Account
    a. Register a new account as a student
    b. Login with your account
2. In class feedback
    a. Add class: COGS120 by Klemmer Scott (class code:123123)

b. During the lecture, choose your feeling about the class content
c. Comment the class material
d. View full class comments

Instructor side:
We would like the instructor user to be able create and delete classes. The following are the three main tasks (one more than the students' side) to complete:

1. Create an account
   a. Login in as an instructor (using account: p@gmail.com 123123)
   b. Logout
2. Create a class
   a. Delete existing class in your class list
   b. Create a new class
3. In-class feedback
   a. Begin the class
   b. read students' comments and reactions during one of the lectures.
   c. End class

Design questions:

1. What do you like the most and least about this application? Why? (By asking this question, we could know where we need to improve and where we did successfully)
2. What other functions would you like to have? Why? (we could update the functionality of this application by receiving feedback from users)
3. Would you recommend your friends and your professors to use this application? Why? (we could get a sense of how practical this application is and infer the usability)

Complete functionality
What we complete this week:

A. Dynamic feedback on the instructor side when a student input information
B. Color code emoji
C. Change the comment history to the history of full class comments
D. No need for view full button
E. Replace the help button
F. Show full instructions above the emoji

How we access with a different programming language:
We used the programming language swift to make our iOS app. For each rubric item below, I've outlined the exact code that accomplish the functionality that would've written in html. I also provided gif files for you to better understand the translation across platforms.

How our workflows implement user tasks:
Since our application is made for instructors and students, there are two unique workflows for each respective user type. The user tasks are explained in detail in the Task section above.

Logins:
Firebase auth login：
Instructor side:
p@gmail.com
123123

Student side:
d@gmail.com
123123

Read data from JSON:
Search for “fetchCourses” instead, this function is what fetches the JSON data from the database.
Or here is the function fetchCourse:

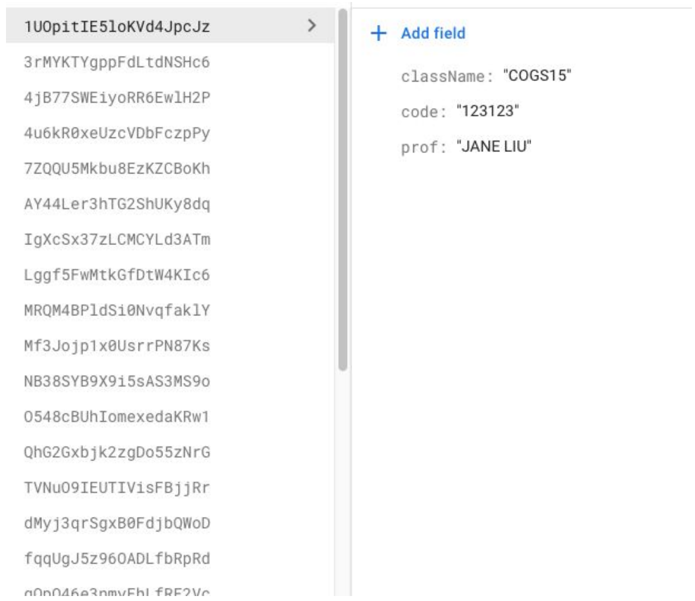
```
fileprivate func fetchCourses() {

let query = Firestore.firestore().collection("classes")
query.getDocuments { (snapshot, err) in
if let err = err {
        return
}
snapshot?.documents.forEach({ (document) in
        let courseDict = document.data()
        self.courses.append(Course(dictionary: courseDict))
})
self.filteredCourses = self.courses
self.collectionView.reloadData()

}
}
```

If you print out the json data after the method fetchCourses
You will see this

```
▿ 0 : Course
   ▿ className : Optional<String>
     - some : "COGS15"
   ▿ prof : Optional<String>
     - some : "JANE LIU"
   ▿ code : Optional<String>
     - some : "123123"
 ▿ 1 : Course
   ▿ className : Optional<String>
     - some : "COGS24"
```

```
  ▿ prof : Optional<String>
   - some : "Rangel Boyle"
  ▿ code : Optional<String>
   - some : "123123"
 ▿ 2 : Course
  ▿ className : Optional<String>
   - some : "COGS12"
  ▿ prof : Optional<String>
   - some : "Sam"
  ▿ code : Optional<String>
   - some : "123123"
 ▿ 3 : Course
  ▿ className : Optional<String>
   - some : "COGS25"
  ▿ prof : Optional<String>
   - some : "Rangel Boyle"
  ▿ code : Optional<String>
   - some : "123123"
 ▿ 4 : Course
  ▿ className : Optional<String>
   - some : "COGS13"
  ▿ prof : Optional<String>
   - some : "Sam"
  ▿ code : Optional<String>
   - some : "123123"
 ▿ 5 : Course
  ▿ className : Optional<String>
   - some : "COGS18"
  ▿ prof : Optional<String>
   - some : "Howard Huang"
  ▿ code : Optional<String>
   - some : "123123"
 Etc.
```

What is looks like in firestore database:



```
1UOpitIE5loKVd4JpcJz
3rMYKTYgppFdLtdNSHc6
4jB77SWEiyoRR6EwlH2P
4u6kR0xeUzcVDbFczpPy
7ZQQU5Mkbu8EzKZCBoKh
AY44Ler3hTG2ShUKy8dq
IgXcSx37zLCMCYLd3ATm
Lggf5FwMtkGfDtW4KIc6
MRQM4BPldSi0NvqfaklY
Mf3Jojp1x0UsrrPN87Ks
NB38SYB9X9i5sAS3MS9o
O548cBUhIomexedaKRw1
QhG2Gxbjk2zgDo55zNrG
TVNuO9IEUTIVisFBjjRr
dMyj3qrSgxB0FdjbQWoD
fqqUgJ5z96OADLfbRpRd
```

```
className: "COGS15"
code: "123123"
prof: "JANE LIU"
```

<u>Screen that reads our json data:</u>
http://g.recordit.co/IlrcqjKtEy.gif

Handlebar template:
In iOS, instead of handlebars templates, to display multiple items of the same type, for example, in this case, display all the class objects in the “filteredCourses”. The first method tells the system how many items of the same type are there or how many classes are there.
Then then the second method, from 0 to items.count, loop through all the items and creates a cell that displays each item.

Looks like this: http://g.recordit.co/LRFLqFGL5g.gif

1.
   ```
   override func collectionView(_ collectionView: UICollectionView,
   numberOfItemsInSection section: Int) -> Int {
   return filteredCourses.count
   }
   ```

2.
   ```
   override func collectionView(_ collectionView: UICollectionView, cellForItemAt
   indexPath: IndexPath) -> UICollectionViewCell {

   let cell = collectionView.dequeueReusableCell(withReuseIdentifier: cellId, for: indexPath)
   as! ClassSearchCell

   cell.course = filteredCourses[indexPath.item]
   return cell
   }
   ```

Utilize routes and jQuery selectors to asynchronously update your app's screens where appropriate :
When a student or instructor adds a new class they are attending. The screen refreshes and a new class object is added to the screen.

```
func selectCourse(course: Course) {
courses.append(course) //adds new json to local cache
studentAddClassToFireStore(course: course) //saves new json to database
collectionView.reloadData() //refresh screen
}

fileprivate func studentAddClassToFireStore(course: Course) {
guard let currentUID = Auth.auth().currentUser?.uid else {return}
user?.classes.append(course.classID ?? "")
Firestore.firestore().collection("users").document(currentUID).updateData(["classes":
user?.classes]) //the exact json data uploading to database
courses = [Course]()
fetchClasses()
}
```

Here's the demo:
http://g.recordit.co/F5T02jcVmE.gif

Prototype writes JSON data :
This method here writes JSON data to the database

```
fileprivate func instructorAddClassToFireStore(className: String, code: String) {
guard let currentUID = Auth.auth().currentUser?.uid else {return}
let filename = UUID().uuidString
let dict = ["className": className, "code": code, "prof": user?.name]
let docId = Firestore.firestore().collection("classes").addDocument(data:
dict).documentID
user?.classes.append(docId)
Firestore.firestore().collection("users").document(currentUID).updateData(["classes":
user?.classes])
courses = [Course]()
fetchClasses()
}
```

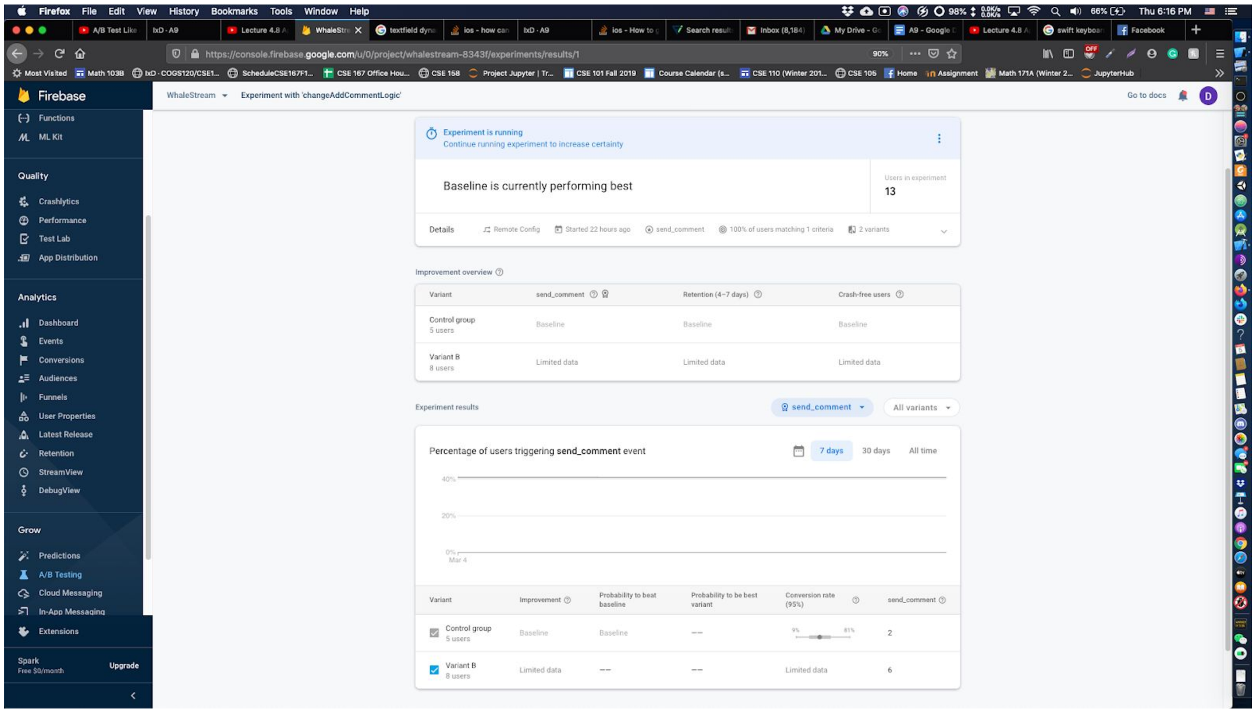
This method adds/writes a new class object to the database. In particular

```
let dict = ["className": className, "code": code, "prof": user?.name]
```

This line of code constructs the new json data.

Compile and Analyze

Screenshots of analysis page (we are not using Google Analytics because we are not doing a web-based app):



Experiment is running
Continue running experiment to increase certainty

Baseline is currently performing best

Users in experiment
13

Details — Remote Config — Started 22 hours ago — send_comment — 100% of users matching 1 criteria — 2 variants

Improvement overview

Variant	send_comment	Retention (4–7 days)	Crash-free users
Control group 5 users	Baseline	Baseline	Baseline
Variant B 8 users	Limited data	Limited data	Limited data

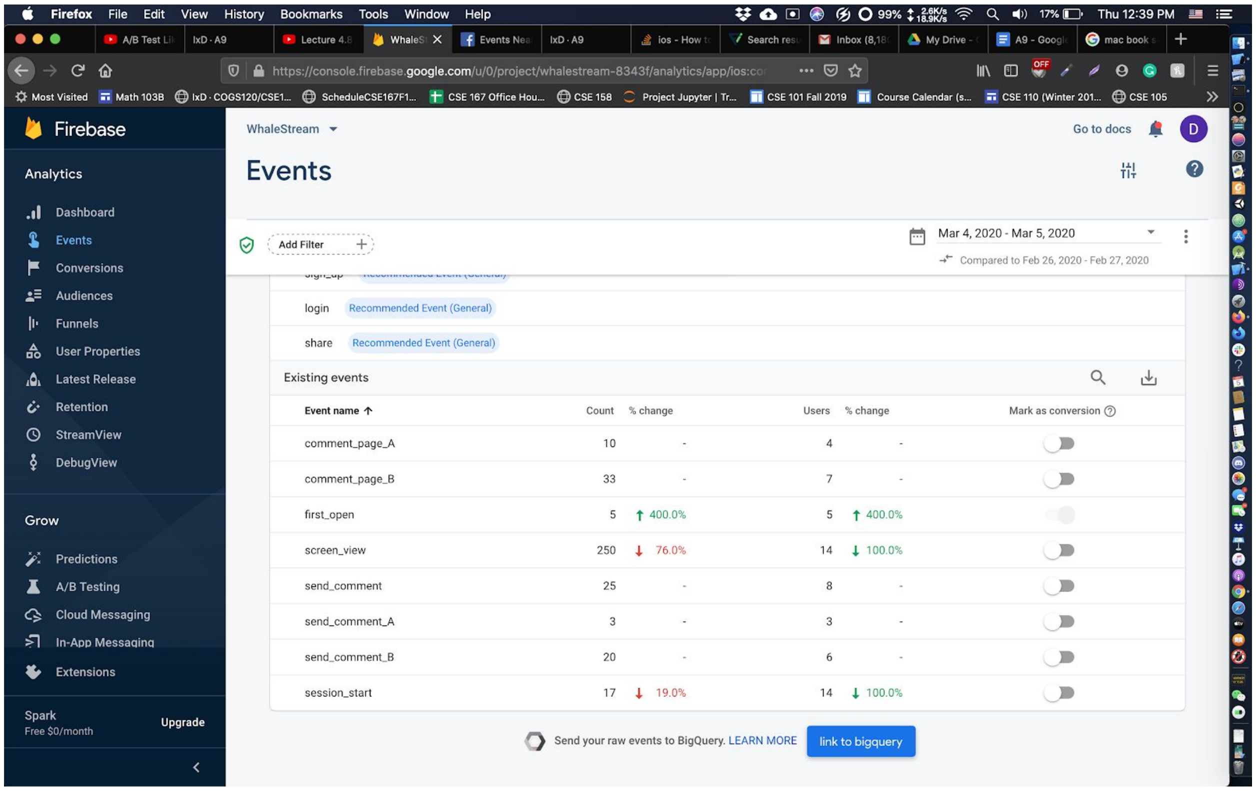
Experiment results

Percentage of users triggering send_comment event

Variant	Improvement	Probability to beat baseline	Probability to be best variant	Conversion rate (95%)	send_comment
Control group 5 users	Baseline	Baseline	––	9% – 81%	2
Variant B 8 users	Limited data	––	––	Limited data	6

From the analytics experiment above, it shows 2 out of 5 users sent comments using version A and 6 out of 8 users sent comments using version B.

The next pictures shows in detail event triggering count.



Events

Mar 4, 2020 - Mar 5, 2020
Compared to Feb 26, 2020 - Feb 27, 2020

login Recommended Event (General)
share Recommended Event (General)

Existing events

Event name	Count	% change	Users	% change	Mark as conversion
comment_page_A	10	-	4	-	
comment_page_B	33	-	7	-	
first_open	5	↑ 400.0%	5	↑ 400.0%	
screen_view	250	↓ 76.0%	14	↓ 100.0%	
send_comment	25	-	8	-	
send_comment_A	3	-	3	-	
send_comment_B	20	-	6	-	
session_start	17	↓ 19.0%	14	↓ 100.0%	

Send your raw events to BigQuery. LEARN MORE link to bigquery

Notations:

comment_page_A is how many views version A gotten
comment_page_B is how many views version B gotten
send_comment_A is number of comments sent through version A
send_comment_B is number of comments sent through version B

We will be using these datas to do our chi-squared test.

Recruitment process

We recruit our participants by randomly selecting users who have not been tested for our previous version before, so we did not recruit the participants who attended our previous user research process or who knew what our app does before. The reason was that we would like our participants to have equal knowledge and understanding of our versions to avoid the practice effect. In addition, we chose our participants all from UCSD students since our component for the test is on the students' side the main page, since students will be the target user group for the tested component. Notice that this time we chose students as our participants on purpose instead of including students who have experienced being an instructor before when we were testing both the students' and the instructors' sides.

Qualitative Data

Page A views: 10
Page B views: 33

Comments sent using page A: 3
Comments sent using page B: 20

Observed:

	Page A	Page B	Comments across both Versions
Sent	3.000	20.000	23.000
Didn’t Sent	7.000	13.000	20.000
Comment Page visited Count	10.000	33.000	43.000

Expected:

23 / 43 * 10 = 5.348

23 / 43 * 33 = 17.651

	Page A	Page B
Sent	5.348	17.651
Didn’t Sent	4.651	15.349
Comment Page visited Count	10.000	33.000

Chi-squared test:

(Observed - expected)^2 / expected

(3.000 - 5.348)^2 / 5.348 = 1.031
(7.000 - 4.651)^2 / 4.651 = 1.186
(20.000 - 17.651)^2 / 17.651 = 0.313
(13.000 - 15.349)^2 / 15.349 = 0.359
Add them up
Chi square value = 2.889
Degree of freedom = 1
So p value is between 0.1 and 0.05
Ultimately p value bigger than 0.05
Therefore the value indicates that we don't have enough data to say one of these versions perform better. Failed to reject null hypothesis.

Data Summary:

（**see above section for our quantitative data**）
Qualitative Data:

Subject Number	Randomized Version	Observation Notes and Feedback	Researcher
1	B	After exploring and switching between the feedback emoji buttons, it took some time to find the comment box below, but sent the comment very quickly twice by typing directly into the box. **Feedback:** looks like a chat window, which is hard to find but then easy to use and view history	Jing
2	B	Found the “comment” box easily. Took some moments to think about what to send. After clicking on the send button on the keyboard, he looked confused before he saw the history. **Feedback:** maybe there should be a send button so that you can view your comment after entering it in the keyboard and choose to send or edit it.	Jing
3	B	The user, after resgistering as the student successfully, found the comment box naturally on the bottom of the screen and send several comments quickly. However, she spends some time to figure out the logical in the comment history box. **Feedback**: She feels easy to use the send comment function, but she wondered why there displays other comments that she did not send.	Eryue
4	A	The user had no problem to understand the whole screen. She automatically clicked the comment box. She was then surprised by the new screen jumped out. She sent one comment successfully and was not going to sent twice. **Feedback**: it is unnatural to jump to a new screen when commenting and jump back after finished. It leaves a burden on the coginitive load.	Eryue
5	B	The user spend sometimes to figure out the place he should type comments in. After understanding it, he is happy to use the function and send several comments quickly.	Eryue

		Feedback: He wish the header could stay when typing.	
6	A	The User had no hard time using the function of the app. Directly using the comment box to give comments without telling her to do so after login. **Feedback:** instruction is clear that there is no need to think what to do next. However, I did not expect to switch to the other page when clicking the comment box.	Jiayu
7	A	The user was trying to slide to the right to go back to the previous page when on the “class list” page. Realized it does not work, she clicked the “Go Back” button. Did not use the emoji function, instead went straight to the “comment box”. **Feedback:** it is better if making every page slidable. The emoji buttons do not look clickable so I did not notice them.	Jiayu
8	A	User clicked comment box and was confused when it directed to a new screen to enter comments **Feedback**: instead a new comment screen, put a send button next to the original comment box	Damon
9	A	The user had no trouble at all sending a comment but didn’t want to send another comment. **Feedback:** the new comment page makes no sense.	Damon
10	B	User had no trouble sending comments after comments, and found it smooth as if using a developed chat messaging app. However, did mention couldn’t type a new line. **Feedback:** change the send button in keyboard to return button so users will be able to type new lines.	Damon
11	B	User finds it hard to edit comment in the comment text field because it is not multi-lined **Feedback:** dynamically resize the comment text box as texts get longer.	Damon

Interpretation

We were unable to conclude which version was better because the resulting p value failed to reject the null hypothesis. If one skim through our data, it is obvious that version B did a lot better (version B users sent comments 60% of the time, version A users sent 30% of the time), ultimately, we don't have enough users to determine which version is better. 10 users as test base is a bit too small, rendering the result inaccurate.

Internal Validity:

As we use specifically the button as the "cause" and use the number of times that users click the button, which constitutes a causal relationship between the effect we would like to test with our A/B test. The action we observe and analyze in our user testing experiment sufficiently reflects our results and conclusions.

External Validity:

our experiment uses the number of times triggering the main function to measure the users' preference on different versions, which can be **generalized** to other apps/website's tests. In this way, researchers can compare if a specific version would encourage the user action on their app's main functions more effectively than the other version.

Changes:

According to our test result, we made several changes on our comment screen. We first add the a caption above the comment history box that informs the user the functionality of the box. Second, we change the color of the emoji **as one of our users reflects the color makes no sense to her.** The most important change we made is implementing a dynamic input box to fit the actual input if there are multiple lines. Additionally, we also make the register page looks better by changing the shape of the placeholder. **In the future**, we would like to make the color of the class header match its color code on the main page. We would also adjust the space that the header occupies to make more space for the rest of the content

We have changed:

1) Add a caption above the comment history box
2) The color of the emoji between satisfied and confused should switch
3) Re-decide the color of "react to lecture"
4) Dynamic insert box for multiple line text
5) Change the shape of the instructor and student option buttons to be consistent with the container
6) Change the profile photo to be fitted in a circle rather than a square on the register page