**2015 Fall Final project rubric**

**Snap! Below the Line DeCal**

This is your chance to show off everything that you have learned this semester in a final project of your choice and design. You can design anything you want!

Picture: https://www.dosomething.org/files/styles/blog\_landscape/public/pictures/blog/200351581-001.jpg?itok=RKuBfHF8

**Fly, my friends, fly!**

Due: 12/2 (You need to present your project in class to earn credit)

Requirements:

* Work with a group of 2 to 3 people (recommended 2)
* Make a unique change to Snap! code of reasonable complexity

TImeline:

* 11/4: Project proposal -- Create a README.md on your branch to describe your project. (All good code starts with even better documentation)
* 11/18: Milestone survey -- By this time, your project should be at least halfway done
* 12/2: Presentation

Note :

* Final project should be much more complex than your mini project
* Project should add a meaningful functionality to Snap !
* Feel free to ask facilitators if you have any questions .

Submission:

You will submit your project by creating and pushing to your group’s branch on the ucb-snap organization on Github. You must give a 3 minute group presentation on the features your team added. Take this as an opportunity to practice your entrepreneurship skills and pitch in front of a panel of investors!

Grading:

This decal must be taken on a P/NP basis. Therefore, there will not be a rubric. Instead, it will be graded on effort and satisfactory completion. (We trust that everyone taking this decal is here because they want to be, and we have no intentions of assigning NP to anyone. We expect you to put in the same number of hours into this project as you normally do -- 6 hours a week including office hours and class meetings.)

How do I know if my project demonstrates reasonable effort? As long as you can successfully argue and convince a stranger that you’ve given your project reasonable effort, you will receive full credit. (Hint: we’re looking for a reason to give you full credit, as opposed to looking for a reason to not give you full credit.)

Tips for working with large code bases:

(Courtesy of Kramii @stackexchange.com)

1. Try to find out what the code is supposed to do, in business terms.
2. Read all the documentation that exists, no matter how bad it is.
3. Talk to anyone who might know something about the code.
4. Step through the code in the debugger.
5. Introduce small changes and see what breaks.
6. Make small changes to the code to make it clearer.

Where do I start?

Start with what you do know. I suggest inputs and outputs. You can often get a handle on what these are supposed to be and what they are used for. Follow data through the application and see where it goes and how it is changed.

Tips for a perfect pitch: http://www.usatoday.com/story/money/business/2014/01/12/10-tips-for-making-the-perfect-pitch-cnbc/4396281/