Carolyn: Use stripe for payments

Carolyn: What is the problem u trying to solve?

Richard: We're basically like a Kickstarter for schools

Carolyn : Are you going to pledge, or pay money to projects right away?

Roman: Pay right away

Carolyn: Is the donations public?

Richard: A lot of concepts in this project will be public.

EJ: We can see how much the project has in total, but can't see who specifically donated the money

Carolyn: Will the student post the budget?

Richard: Yes

Carolyn: How will the student receive the money? Directly in cash?

EJ: When the student gets the money, the student will get it through the school

Carolyn: It's going to be difficult, so you might want to reconsider.

Carolyn: What if the project is bad, or the students just go out wasting the donation. Is there a contract?

Richard: Updates. It let's the alums keep track of the updates

Carolyn: Think about how to make the website trusworthy to the alumni

Carolyn: For the project display page, think about how you will present different types of projects. Also, videos are a very important concept in Kickstarter. How will you benchmark it?

Roman: We're going to have Youtube videos

Carolyn: Then you should embed youtube videos in your project display page.

Roman: Should we have rich-text forms, or just plain forms with specified fields for the input for the project?

Carolyn: The plain forms will it consistent from project to project. Don't use rich text editor.

EJ: What should be the media for the updates?

Carolyn: As for the updates, think about what would make communications easiest? Think about gist from github. Maybe enable public and private advice between alums and students. It could be a full-blown forum.. up to you! It would be really good to enable private communication in some form.

EJ: How about user authentication. Should we use emails?

Carolyn : What are the options you guys considering?

Richard : Just emails

Carolyn: Yeah use the devise gem, for example. Don't spend too much time using certificates if you don't know how to do it.

Carolyn: What types of students are you letting in? There can be cross-registers.. Try to make your system is secure.

Ej: How should we handle filtering and matching?

Carolyn: People will be looking for different things. Certain people, Certain keywords,

Richard: We can enable user profiles for students.

Carolyn: Students can fill in their profiles, have a label for a fraternity or a sorrority, etc. Have the same field for every student, to keep it simple.

Carolyn: As for search, you can use tags. When you have a search box, maybe it'll suggest different tags, and it'll only search that tags.

Carolyn: Payment system will definitely be a risk. Everyone will have to learn what devise does when you want to handle sessions, so make sure Christian teaches you.

Carolyn: As for styling, just make sure the UI makes sense. Maybe follow the kickstarter model. It doesn't have to be beautiful, but it has to be usable

Carolyn: Most of today's meeting was me throwing ideas at you guys, but for the following meetings it should be the other way around.

Carolyn: For the MVP, make sure you handle privacy settings, and make sure the features support your user goals. Carefully choose what will be included in the MVP, because that will be ultimately shaping what your final deliverable will look like.

Carolyn: Data model; get rid of the views arrows. Don't use the word 'has'. It's vague. Label it with 'Donations', 'Updates', and such. 'Sendor' should be 'Poster'

Richard: Only students who posted that Project can post an update. How can we reflect that in our data model?

Caroyln: Good question. I think it's okay if you leave that as is, and mention it in the footnotes.

Carolyn: How about collaborators? Multiple students can work on the same project.

Roman: A representative student will be administrating the project page.

Carolyn: It's important that you at least make a list of collaborators, at the least

Roman: Should we have a link between project and alumn in the data model?

Carolyn: No. Same goes for advices.

Carolyn: Take out the 'views' arrow. The data model is back-end, not front-end

Carolyn: For the design doc, make it really clear what your platform does. Don't be vague. Don't just give an example situation how it can be used and expect people will understand. Be very explicit about existing solutions.

Carolyn: Maintain the same level of clarity throughout all phases. For the design doc, write it as if it's the final docs. Have a full feature list. and have something like a checkbox next to the features of whether you'll include it in the MVP phase.

Carolyn: Make sure you address the privacy issue in the MVP.

Carolyn: You never want to display the list of users who made the donation.

Roman : Why? Look at charity apps as an exampleCarolyn: Good point. Include this in your design challenge, and give a reason of whichever decision you make.

Carolyn: Feedback on the presentation - avoid having too much text in the presentations.

Carolyn: You have like 9 days until the MVP, so get working now. On your master branch in github, get a 100% running code. and everyone takes a feature to work on, and develop on the individual branch. Always have the latest working master branch locally.

Me: Should we divide the work centered around the model, rather than the feature?

Carolyn: The feature

Carolyn: For the design doc, have an author for each section.

Carolyn: Critique - look at another teams. and do what I did for you guys today. It should be something like 'please clear up this for me'. Send a pdf based out of a powerpoint.