Social Computing Capstone

Day 8: Prototyping Social Computing Systems

CSE 481p | Winter 2024

Amy X. Zhang
Instructor
Assistant Professor | University of Washington, Allen School of Computer Science & Engineering

Ruotong Wang
TA
Ph.D. Student | University of Washington, Allen School of Computer Science & Engineering

Schedule for today's class

- Lecture on prototyping (15 min)
- [pause] Deepti Doshi remote talk and Q&A (20 min)
- Continue lecture on prototyping (30 min)
- Ruotong discusses how LLMs can help with prototyping (15 min)
- Not much work time today, you'll get a lot more in class work time on Thursday.

Announcements

Thanks for your pitches on Thursday! Now that you've gotten feedback on G1, please try to revise it and then onto your project website as a blogpost by ~EOD today.

Prototyping Social Computing Systems

I have an idea. Where do I start?

Or, why “build it and launch” is a bad approach, and how to do better.

- Prototyping
- The Cold Start Problem

What we learn in CSE 440

If we’ve learned anything from human-computer interaction (HCI), it’s that it’s a really bad idea to have your idea and then Just Build It.

Just Build It is an immense waste of time, energy and resources in order to find out that your idea was terrible.

Instead of Just Building, HCI pushes for the concept of **prototyping.**

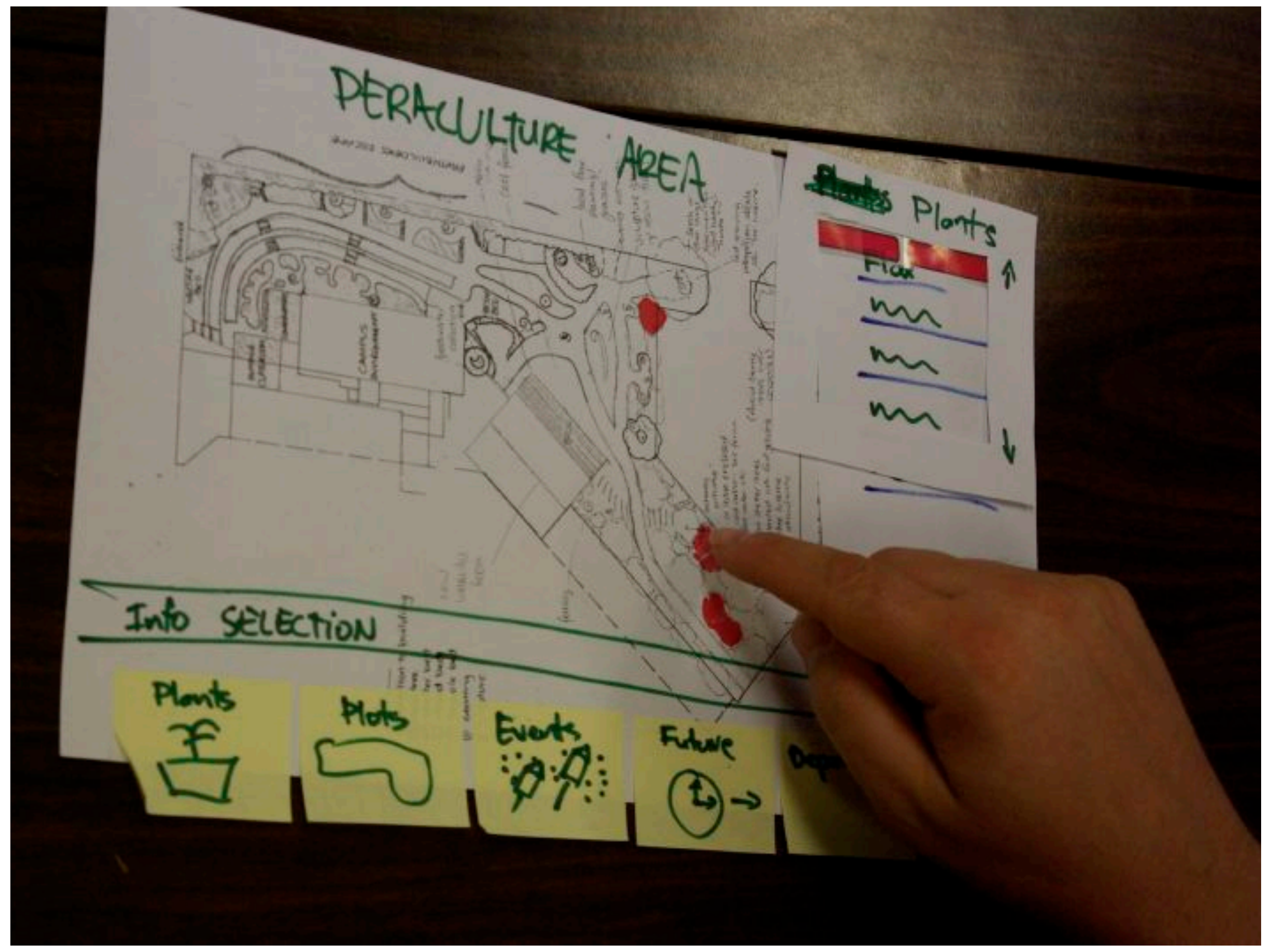
Prototyping

Progressive fidelity increases as you gain confidence in your idea

Low-fidelity paper prototypes that are rapid to create and explore

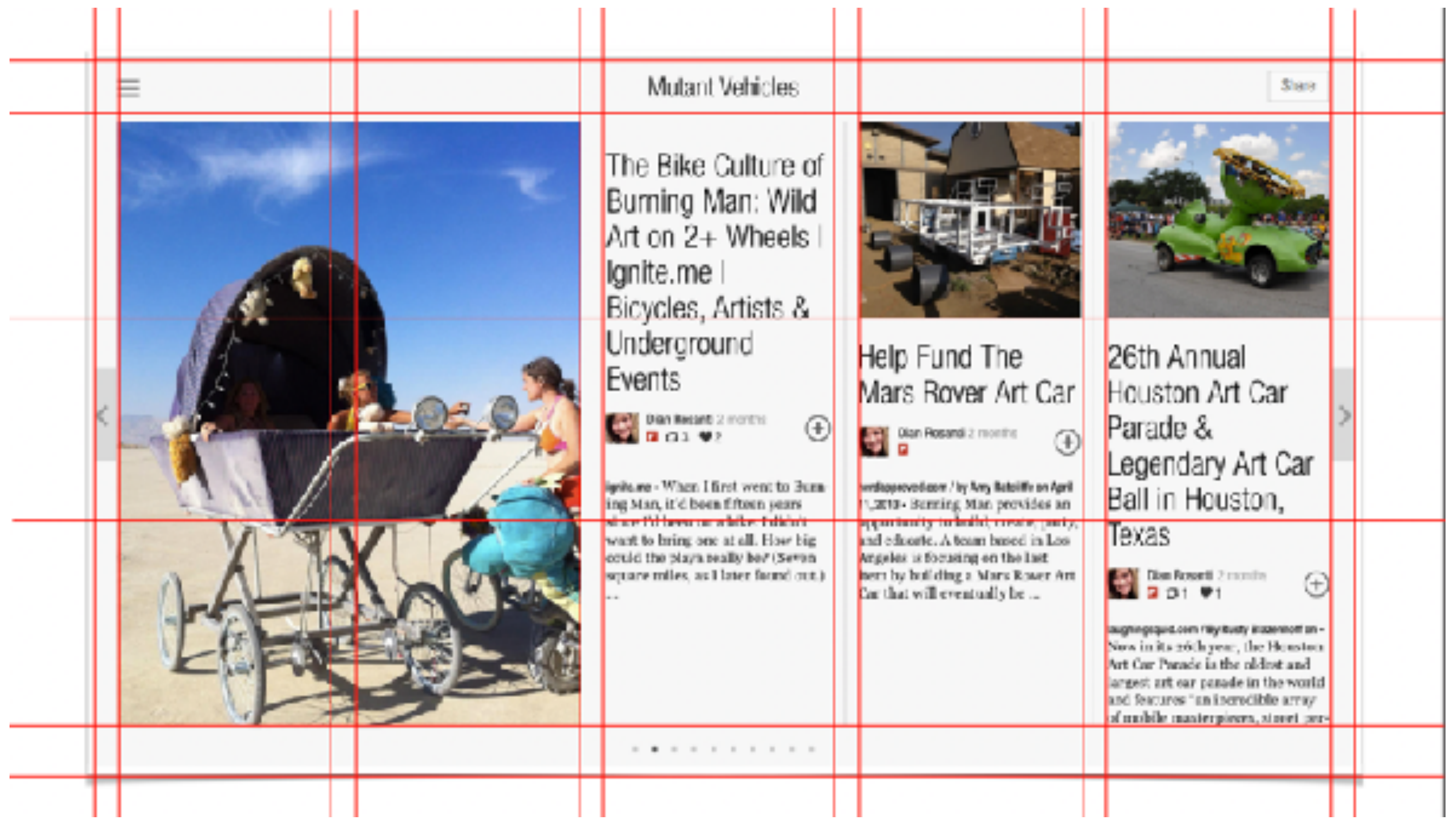
Richer instantiations that answer lingering questions

Detailed refinements of the approach



Flickr: Samuel Mann

IDEO


Mutant Vehicles
Share
The Bike Culture of Burning Man: Wild Art on 2+ Wheels | Ignite.me | Bicycles, Artists & Underground Events
Help Fund The Mars Rover Art Car
26th Annual Houston Art Car Parade & Legendary Art Car Ball in Houston, Texas

Erin Malone

But just making traditional low-fidelity prototypes isn't the best fit.

Why?

The basic interface flows aren't what need to be tested.

Example: On-demand office hours

Imagine that the Allen School had allocated a few floating TAs across classes to offer ad-hoc help/support/tutoring on demand.

Your idea is to create a social computing system where you can request help, and if any of the floating TAs had previously TA'ed the class you need help with, and are awake, they'll swing by.

Why is a paper prototype not that helpful here?

You're not really answering the question at the core of the idea.

A prototype answers a question

Prototypes shouldn’t focus on a specific modality, e.g., paper.

Instead, prototypes should focus laserlike on what’s the big risky unanswered question about the idea.

Oftentimes, that question is not whether the interface is usable. It’s **how the social dynamics will play out.**

How might you “wizard of oz” a social computing prototype?

“Wizard of oz” - just like in the book/movie, instead of building the thing, simulate it with a person pulling the strings behind the scene!

Don’t build the entire technical stack just to answer a question. Instead, piggyback on existing social computing systems that get you similar affordances. [Grevet and Gilbert 2015]

> Push messaging/notifications? Manually send texts to them instead.
> Collaboration? Use GDocs or GSlides.
> Chatbot? Manually message people in Slack or Discord.

For your eventual system, you could build a bot or extension on top of these systems. For now, you can fake it with a person.

On-demand office hours

What's the big unanswered social dynamics question about the on-demand office hours idea?

How could we wizard-of-oz a prototype to answer it in 24 hours?

When *might* you want to integrate paper prototypes into this?

Example 2: Peer emotional support

Imagine an anonymous social computing system where UW students can suggest other UW students who need support: conversation, a hug, etc.

Student volunteers fulfill the request themselves (e.g., go swing by their room), or find and recruit a friend of the requestee to do it.

No guarantees that this is a good idea. How would you find out?

Prototype simple social formulas

It's very tempting to try and design out an entire social ecosystem.

Don't.

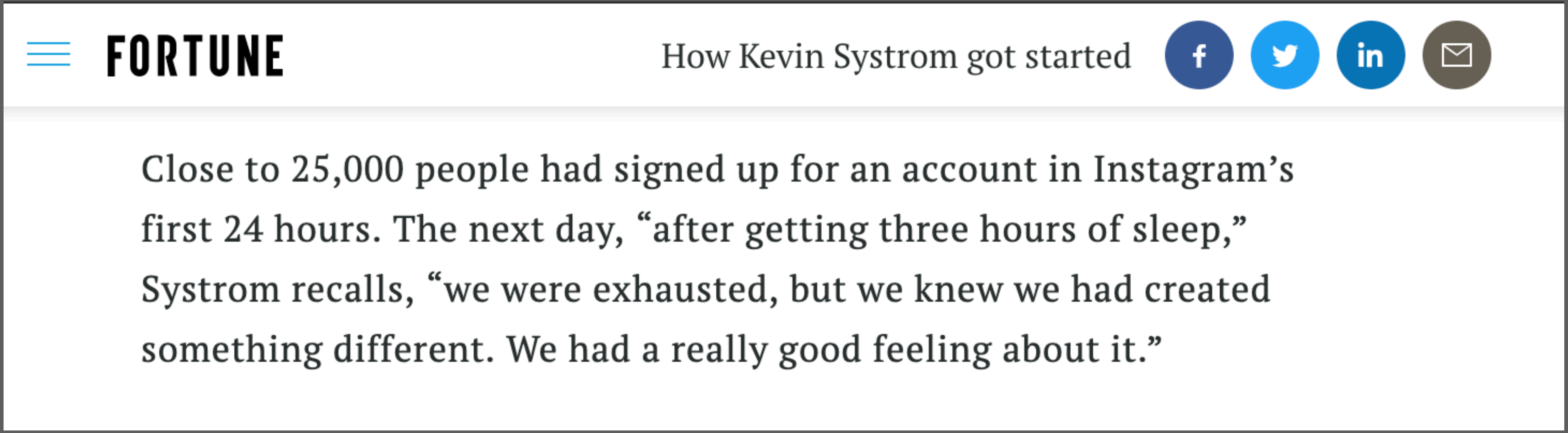
Sounds like a Dungeons and Dragons campaign

> "Oh we'll let people post photos of their dogs. If it gets more than 33.5% upvotes, it'll get a Pupper badge. But then others can sidevote the photos, and if there are enough sidevotes, it starts a chain. Chains are groups of Puppers that are displayed together and aggregate their upvote counts. If someone uploads an old Pupper, the mod (oh by the way there are mods. they're called Doggos.) can remove the sidevotes and...

Prototype simple social formulas

Instead, identify the simplest set of social interactions that is possible. This focuses the social formula you're prototyping down to its core.

The cold start problem


FORTUNE

How Kevin Systrom got started

Close to 25,000 people had signed up for an account in Instagram's first 24 hours. The next day, "after getting three hours of sleep," Systrom recalls, "we were exhausted, but we knew we had created something different. We had a really good feeling about it."

This is an outlier.

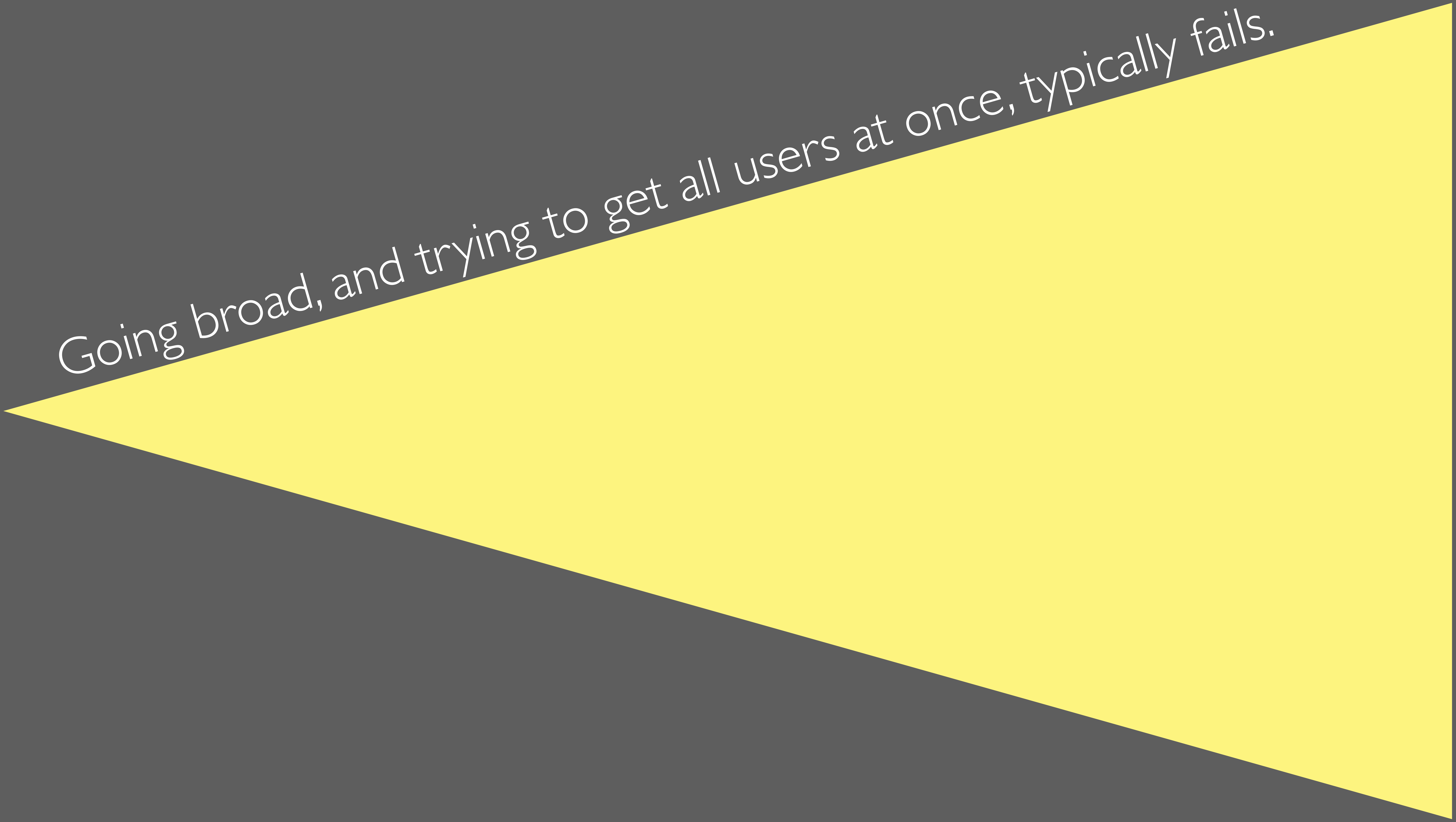
Cold start problem

The problem: the social computing system isn't really very enjoyable or useful to anybody when nobody's there yet (remember: network effects).

...but then, why would someone join and start populating it, if there's nobody there?

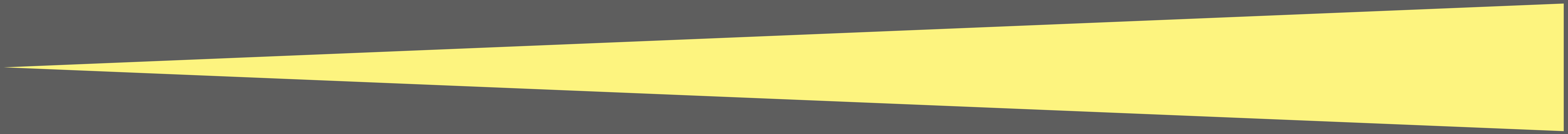
The entire effort struggles to hit **critical mass**, like how a car engine on a freezing day can't start up because it's too cold, and if it can't start up, it can't warm itself up to start. Thus, a cold start problem.

Going broad, and trying to get all users at once, typically fails.



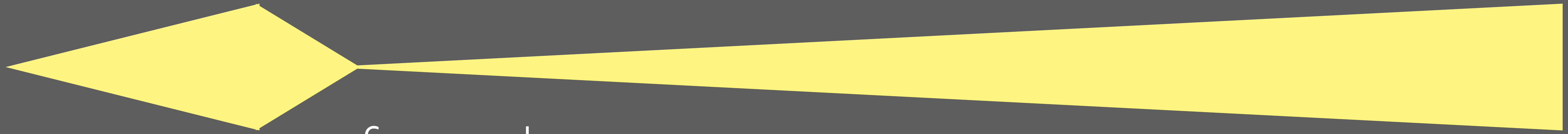
Building a focused, engaged core initially is better design.

Why?
- The designer has a clear sense of who the users are, what issues they face, and what norms they expect
- Cliques help overcome the cold start problem



Harvard undergraduates

launched as justin.tv, tried to be relevant to everybody, but failed



Kickstart by creating a social space that affirms a group's sense of identity (for example: Subtle Asian Traits)

- Look for micro-cultures, e.g., coffee snobs, surfers, people who love nail art

Don’t prototype sparsely

Remember we talked about **network effects** in a prior class.

Typically social computing systems rely on network effects, like a bunch of friends all using them together. Getting sparse networks by having a few friends from here and there won’t work.

Instead, get a small **clique** (the graph sense, not the high school sense) to join the prototype together. In other words, a small dense group >>> a larger diffuse group.

Bootstrapping content and interest

Prototypes do not magically become bustling spaces. Often they feel like ghost towns, because there's nothing there yet.

When prototyping (and deploying), you will need to **bootstrap** the bustling spaces to help set the norms and encourage contributions.

Reddit co-founders initially pushed content until the community took over. (They used many fake accounts).

If it's a two-sided system — e.g., students and huggers — be prepared to prop up one side of the system to make it useful to the other side.

Today maybe you could automatically make fake accounts (Social Simulacra paper). In both cases there are important ethical considerations about notice and consent (anyone seen the movie The Truman Show?).

Summary

Prototyping social computing systems requires a different approach than usual. Use “wizard of oz” prototyping and lean on existing social systems in order to understand the social dynamics you’re creating. Then consider paper prototypes if you have a new interface you want to test.

The cold start problem occurs when a system is too empty to attract initial usage, so it remains empty. Two solutions:

Focus on a narrow group initially, and broaden out later

Be prepared to bootstrap activity!

~3 min bathroom break

Group projects

It's time to move on to imagining solutions! Reminder that G3 (low-fidelity prototype) is due next Tuesday. The prototype needs to be done and ready to go BEFORE class so that in class, each group can be paired with another group to actually RUN the prototype testing and feedback.

We've split up G3's turn-in assignment to G3AB and G3C to reflect this. G3AB is due before class and describes your lo-fi prototype and wizard-of-oz test plan. Then you'll actually run this in class. Then G3C is due EOD describing how your test went and what you learned.

Brainstorming solutions for your group project

Here's activities you can do to spark creativity if you're in need of more solution ideas:

1) "Crazy Eights" - do this activity again but with more targeted solutions to your narrowed-down problem space.

2) "Map the Design Space" - name all the dimensions upon which your solution could vary. Then come up with as many values as possible for each dimension. From this table, roll a die to pick a random selection of values and try to turn that into a cohesive idea. Do this several times. (See slides 34-37 in Lecture 3 of CSE 440: https://courses.cs.washington.edu/courses/cse440/21au/slides/03-Ideation_Critique.pdf)

After coming up with your list of solutions, you've got to settle on one (or a combination solution) relatively soon! Can discuss or take a vote.

After that: plan your low fidelity prototype

What's the big unanswered social dynamics question about your idea? What are the riskiest unknowns at this point? <- Discuss and come to an agreement.

How could you wizard-of-oz a prototype to answer it? Do you want to just rely on existing social systems or also make a paper prototype? Do you want to have one prototype or have a few slight variations to try out?

LLMs as a tool for social computing prototyping