

Lecture 29:

# USER-CENTERED DESIGN

(and other tools for making your project go a bit more smoothly)

---

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

# Outline

- A motivating example
- User-centered design
  - What it is
  - Why do it
  - Ways to do it
- Life skill #1: paper prototypes
- Life skill #2: architecture diagrams

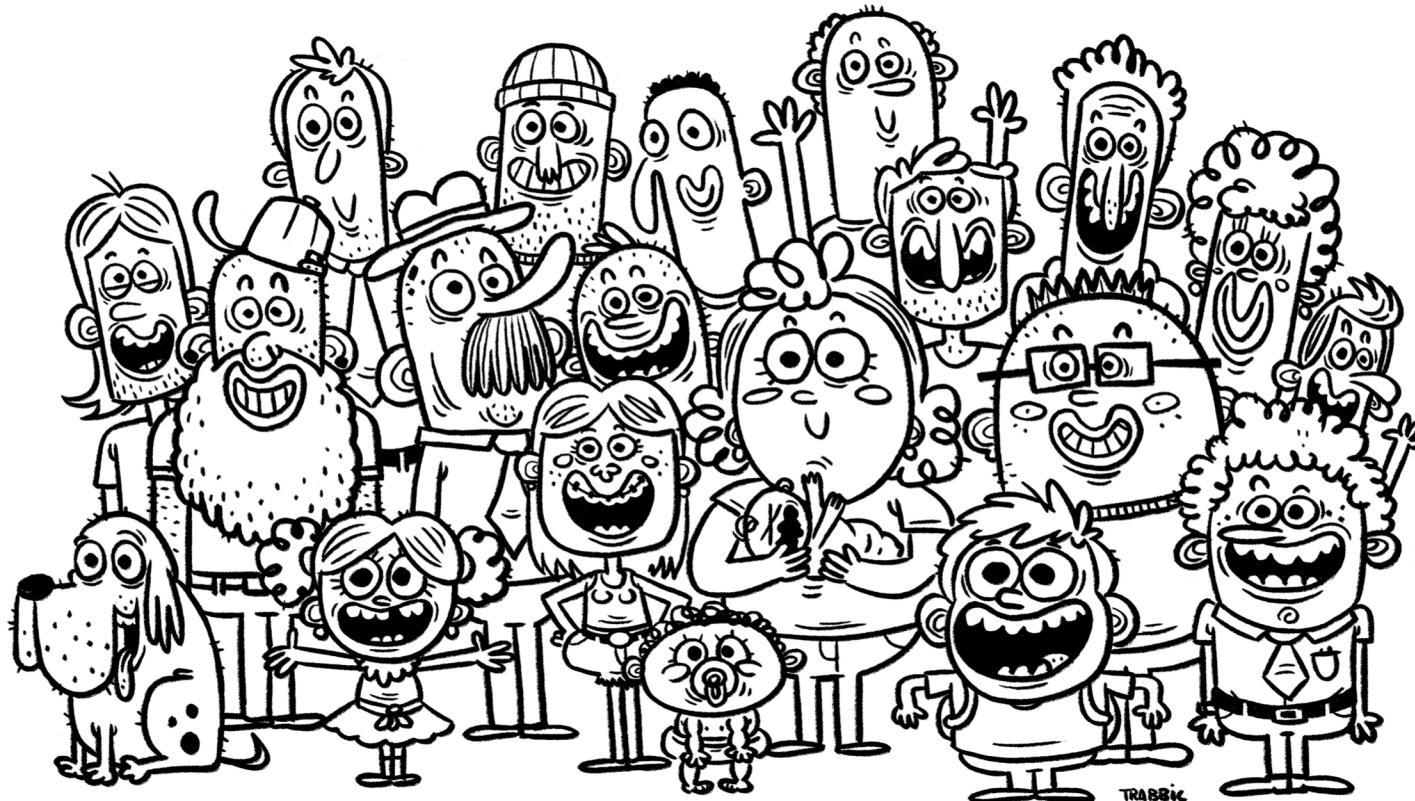
# Hypothetical example

## “Advising Assistant”

**Team:** Jordan Crouser and Dominique Thiebaut

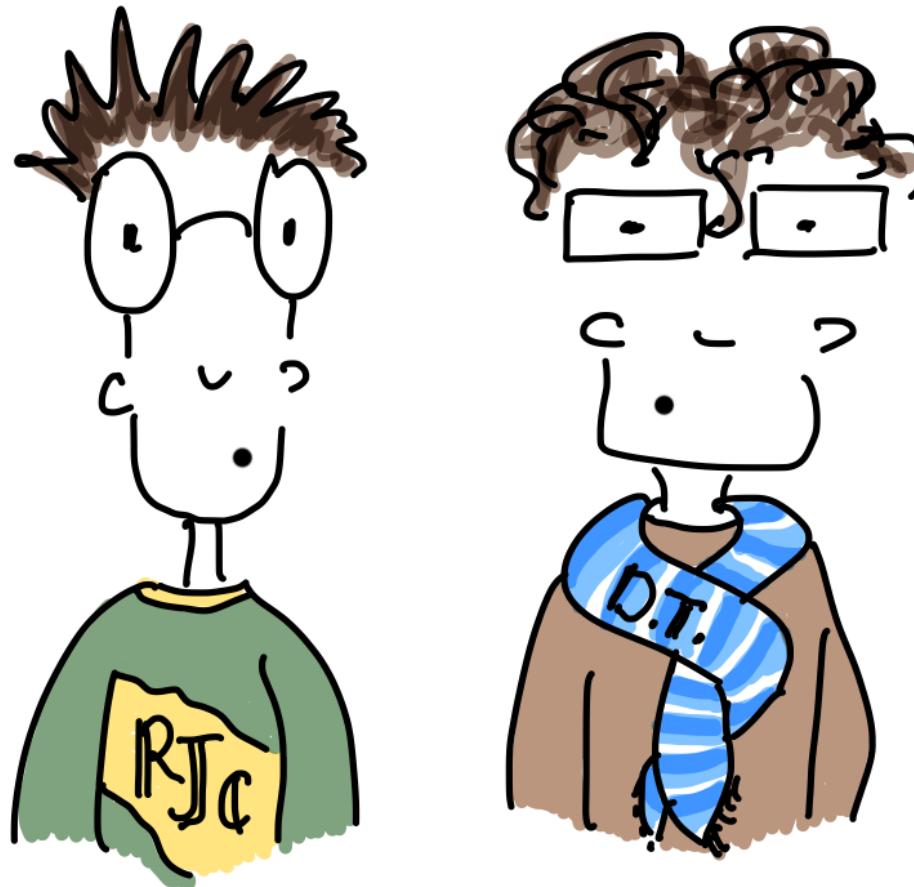
# Hypothetical example

- **Overview:** 122 majors and minors in CSC



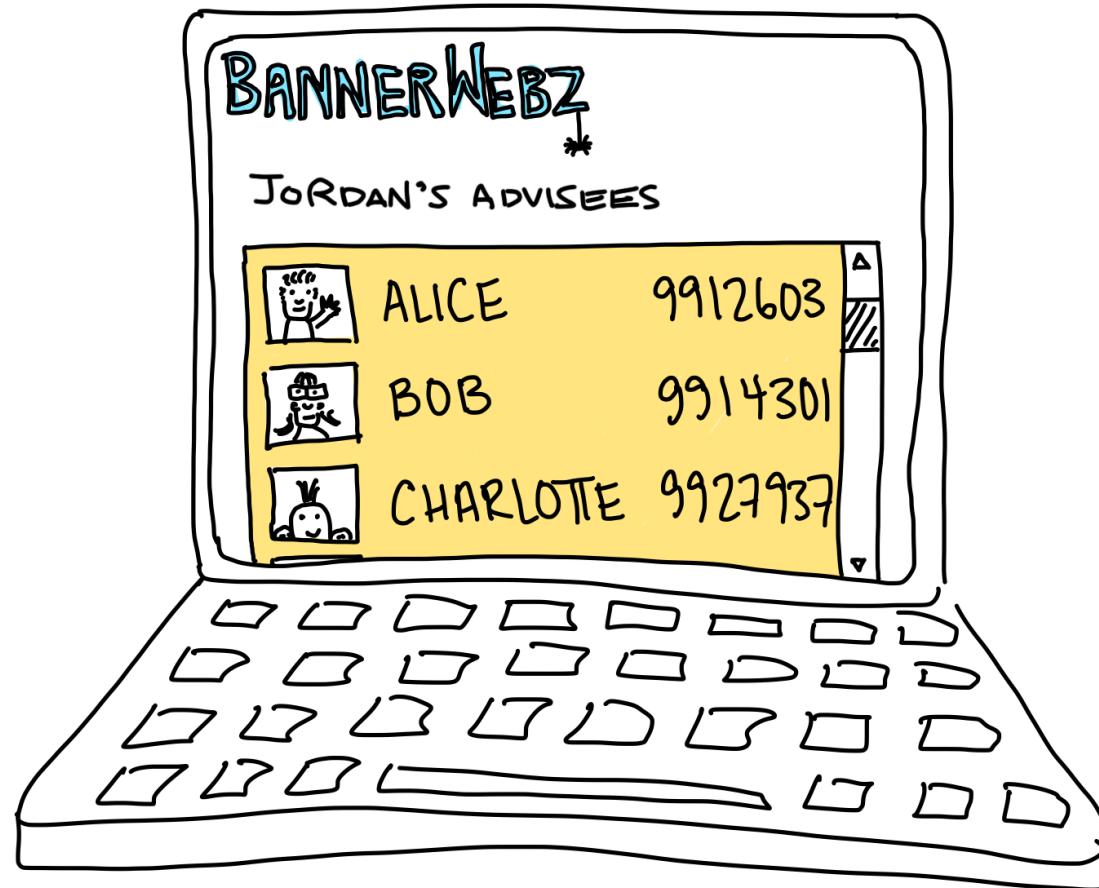
# Hypothetical example

- **Overview:** two advising CSC faculty in Spring 2019



# Hypothetical example

- Overview: BannerWeb not hugely helpful...



# Hypothetical example

- **Overview:**

current process  
is manual

## Goal 1:

detect which courses a student has taken that count toward the major

## Goal 2:

give adviser an overview of all advisees

NAME 2020 99X		SEMESTER		REQUIREMENTS		NOTE					
				CSC111							
				CSC212							
				CSC231							
				CSC250							
				MTH111+							
				MTH153							
				Theory							
				Programming							
				Systems							
				Seminar							
				Elective / +4 Intro							
Officially Declared Senior Certification Honors Thesis?											
FALL	2016	2017		2017	2018		2018	2019		2019	2020
			Outside Major?			Outside Major?			Outside Major?		Outside Major?
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
SPRING	2017	2018		2018	2019		2019	2020		2020	2021
			Outside Major?			Outside Major?			Outside Major?		Outside Major?
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
			▼			▼			▼		▼
Transfer (inside major) 0 ▾ Total Credits: 0											
Transfer (outside major) 0 ▾ Outside Major: 0											

# Hypothetical example

- **Ideal:** we would like to build a plugin that sits on top of Banner and does everything for us, but that won't work because of FERPA concerns (can't alter Banner)
- **Actual strategy:**
  - Define **mapping** from course numbers to major/minor designation (core, T|S|P, seminar, etc.)
  - **Export** CSV of all advisees from Banner
  - Build a **parser** that extracts data from unofficial transcript (i.e. courses taken) and joins with mapping, majors/minors
  - Build **authenticated frontend** for adviser to track student progress, as well as (maybe) send messages / schedule appointments

# Hypothetical example

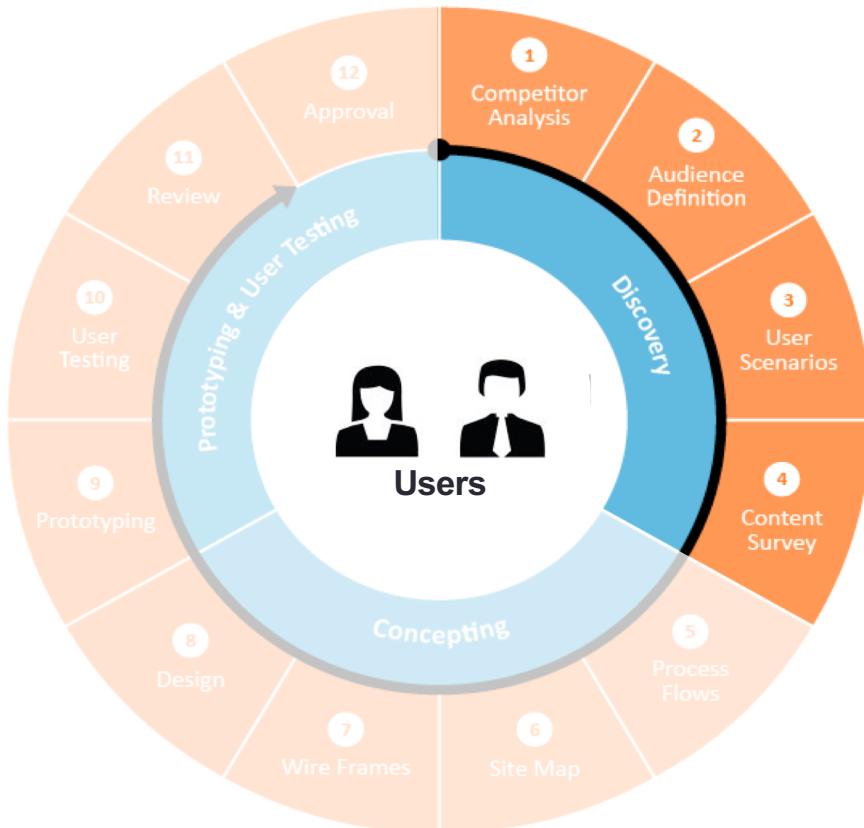
- **Ideal:** we would like to build a plugin that sits on top of Banner and does everything for us, but that won't work because of FERPA concerns (can't alter Banner)
- **Actual strategy:**
  - Define **mapping** from course numbers to major/minor designation (core, T|S|P, seminar, etc.)
  - **Export** CSV of all advisees from Banner
  - Build a **parser** that extracts data from unofficial transcript (i.e. courses taken) and joins with mapping, majors/minors
  - **Build authenticated frontend for adviser to track student progress, as well as (maybe) send messages / schedule appointments**

# Discussion

Let's say you wanted to actually implement this;  
**where do you start?**



# User-centered design framework



## 1) Discovery

- Learning about your users
- Modeling your users
- Analyzing your users' tasks
- Eliciting and defining clear product requirements

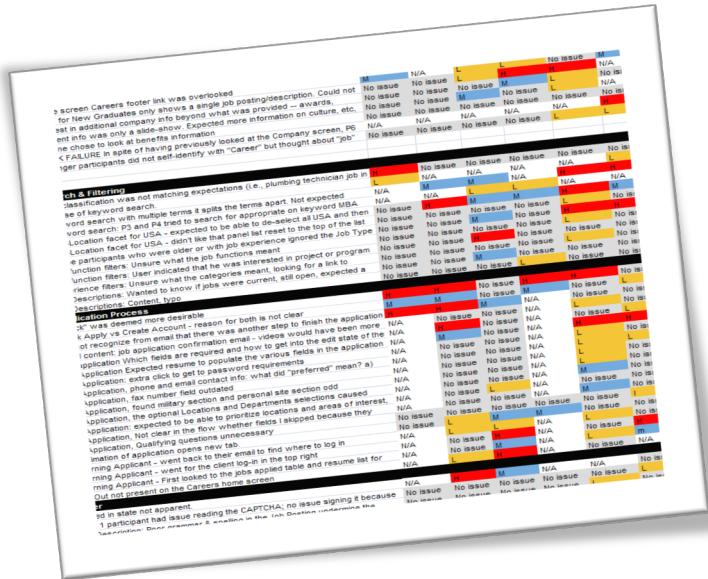
## 2) Concepting Phase

- Developing conceptual models
- Solving design problems through ideation
- Detailed design activities

## 3) Prototyping + User Testing

- Delivery of a high-quality product that meets users' needs and is easy to learn and use

# Competitive review



## • Why?

- If you look at what already exists, you might be able to identify potential issues in advance
- Also helps establish your unique contribution

## • How?

- Literature or product review
- Analysis
  - What are the existing tools?
  - What is their purpose?
  - **What audience are they aiming for?**
  - What kinds of strategies are they using?
  - What functionality do they contain?
  - What are their strengths and shortcomings?
- Identify opportunities and design constraints

# Defining your audience

- Learning about their problem
  - Semi-structured interview
- Analyzing their tasks
  - Hierarchical task analysis
- Modeling users
  - Personas

# Semi-structured interviews

- **Why?**

- gather qualitative data about users to understand the problem
  - can help identify key differences between designer and target user

- **How?**

- ask open-ended questions
  - bring along a “cheat sheet” to ensure that you gather all the information you need

- **Some tips:**

- establish trust at the beginning
  - participant engagement will vary
  - be flexible, but make sure you get what you came for
  - consider recording or note-taking to help with recall



# Defining your audience

- Learning about their problem
  - Semi-structured interview
- Analyzing their tasks
  - Hierarchical task analysis
- Modeling users
  - Personas

# Hierarchical task analysis

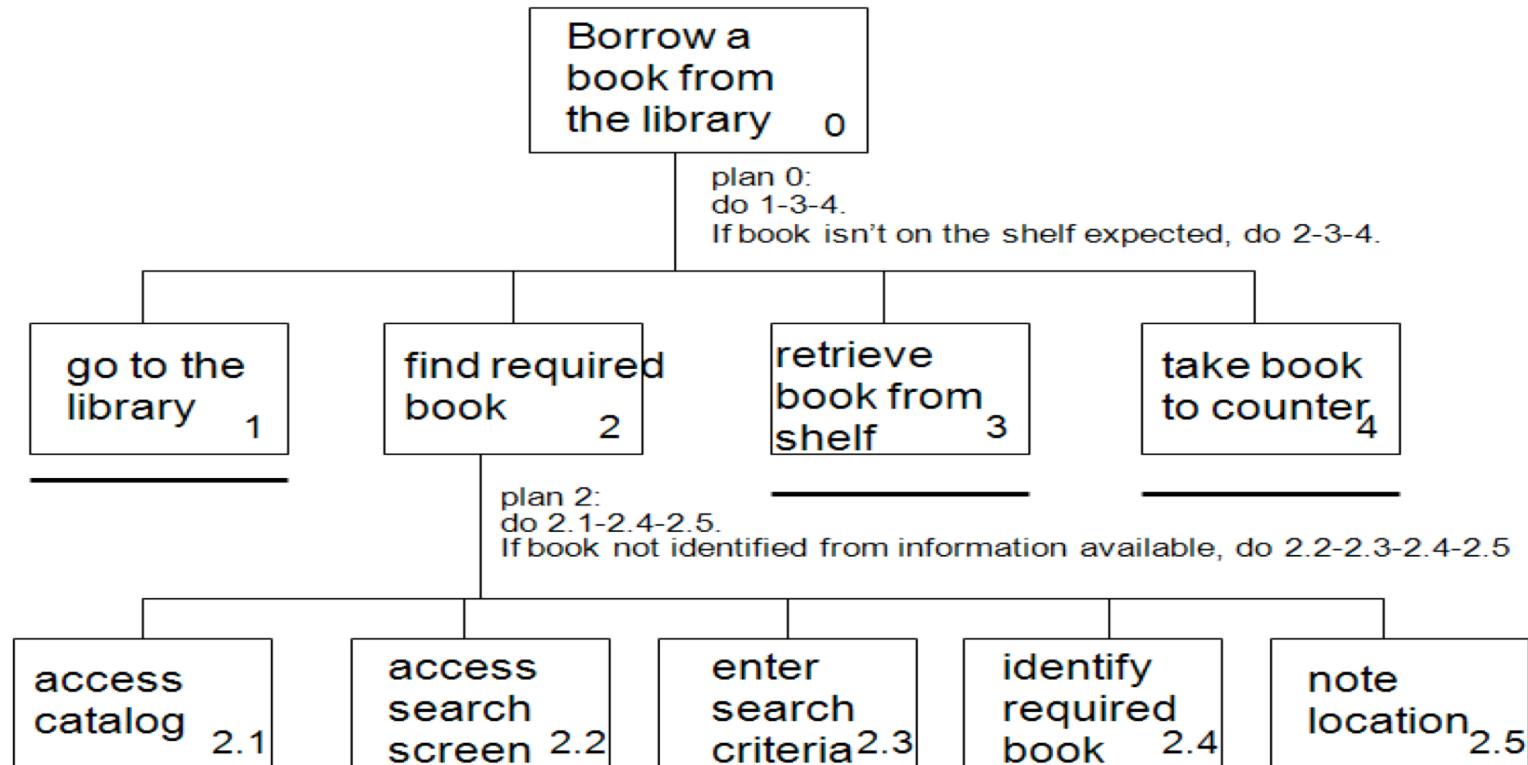
- **Why?**

- Understand user workflow
  - Identify pain points and areas for optimization

- **How?**

- Decompose tasks into 4-8 sequential steps
  - Identify patterns, sequences and skips in the tasks
  - An example:

# Task analysis example

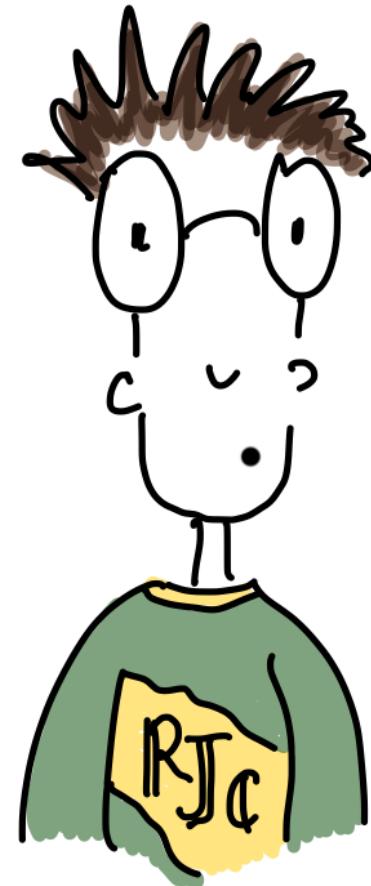


# Defining your audience

- Learning about your users
  - Semi-structured interview
  - Contextual inquiry
- Analyzing users' tasks
  - Hierarchical task analysis
- Modeling users
  - Personas

# Personas

- **Why?**
  - mechanism for reasoning about user needs
  - model behavioral characteristics of target users
  - doesn't require access to ACTUAL users
- **How?**
  - fictionalization
  - narrative, goals, needs, “pain points”
  - attributes specific to the problem space
  - data-driven method\* using info from interviews
  - mapping persona to software features



# Activity: personas

**Goal:** come up with **3 personas** that characterize people who might be interested in your project



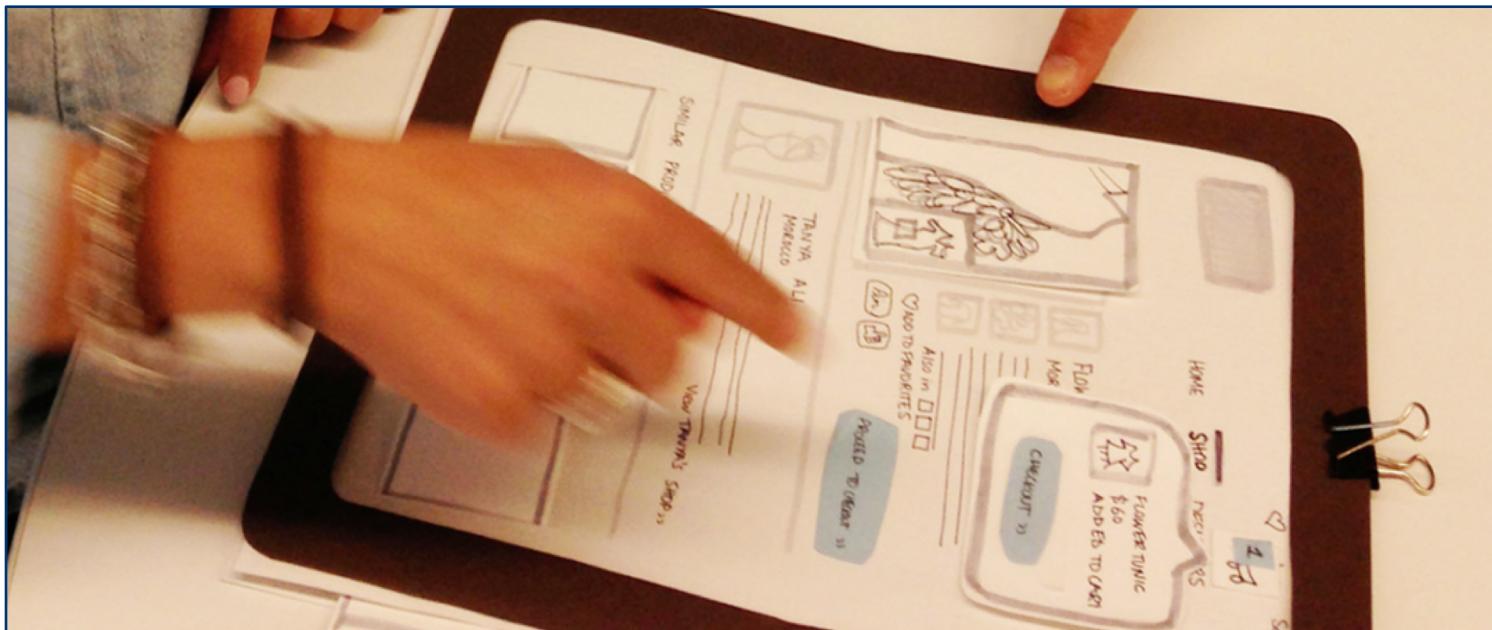
# Discussion

Now that we've got some end users in mind,  
what would a **prototype** look like?



# CS Life Skill #1: “Paper prototyping”

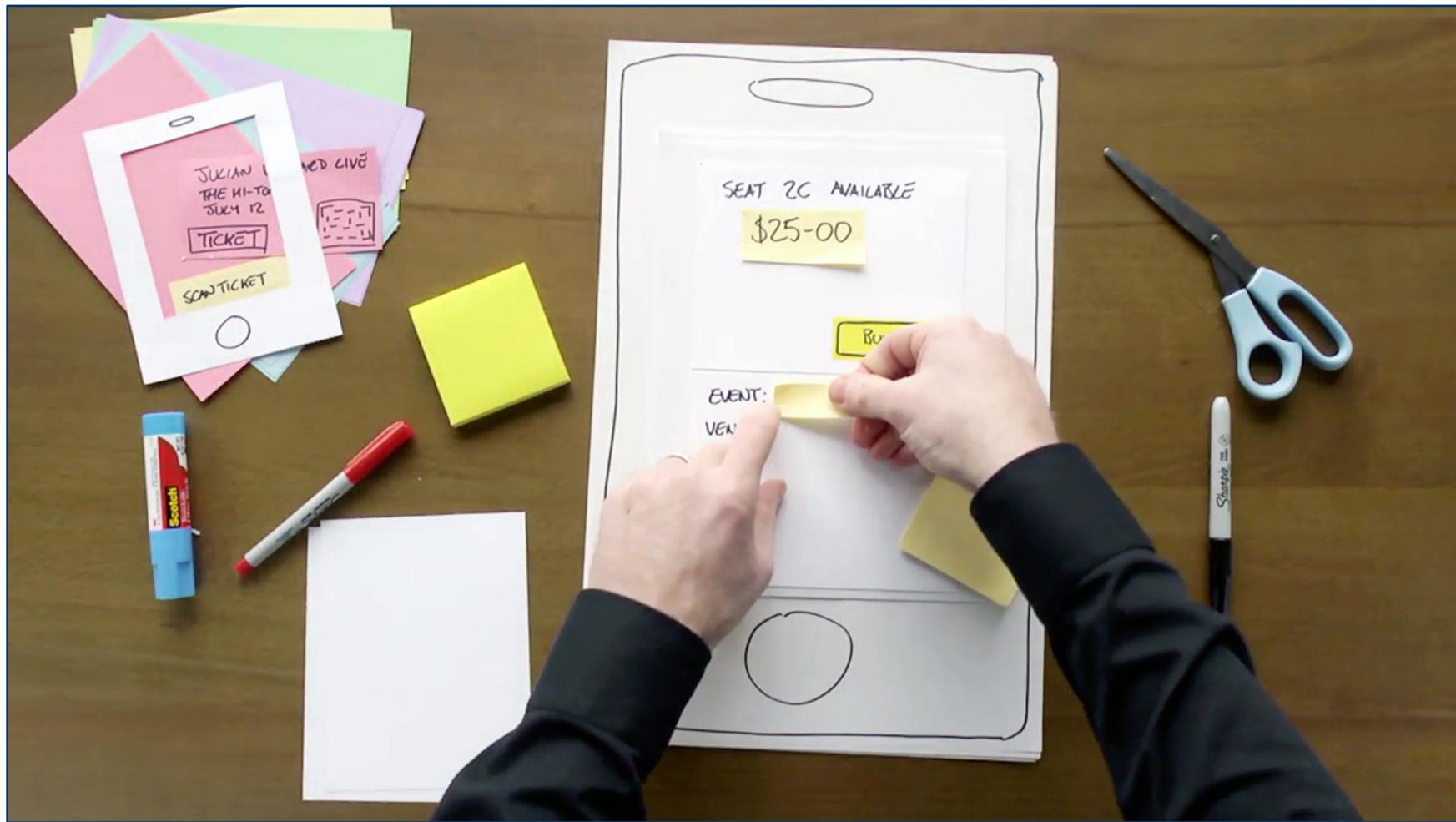
- **Big idea:**
  - Not sure yet whether or not an **idea** will work?
  - Making a **paper version** of an interface is a lot faster and easier than coding a working prototype – start there!



# “Paper prototyping” goals

- Generate **lots of ideas**
- Engage **other people** in the design process
- Identify **potential problems** before you waste time coding
- Get **feedback** quickly, from lots of different people
- Some tips:
  - Focus on the **big picture**, don't worry about the details
  - **Think about what you want it to do**, rather than what you know how to implement (we'll worry about that later)
  - Not so into arts and crafts? It doesn't have to be **actual paper**... Whiteboard / PowerPoint / Keynote will also do the trick!

# Your turn!



# Discussion

How did it go?

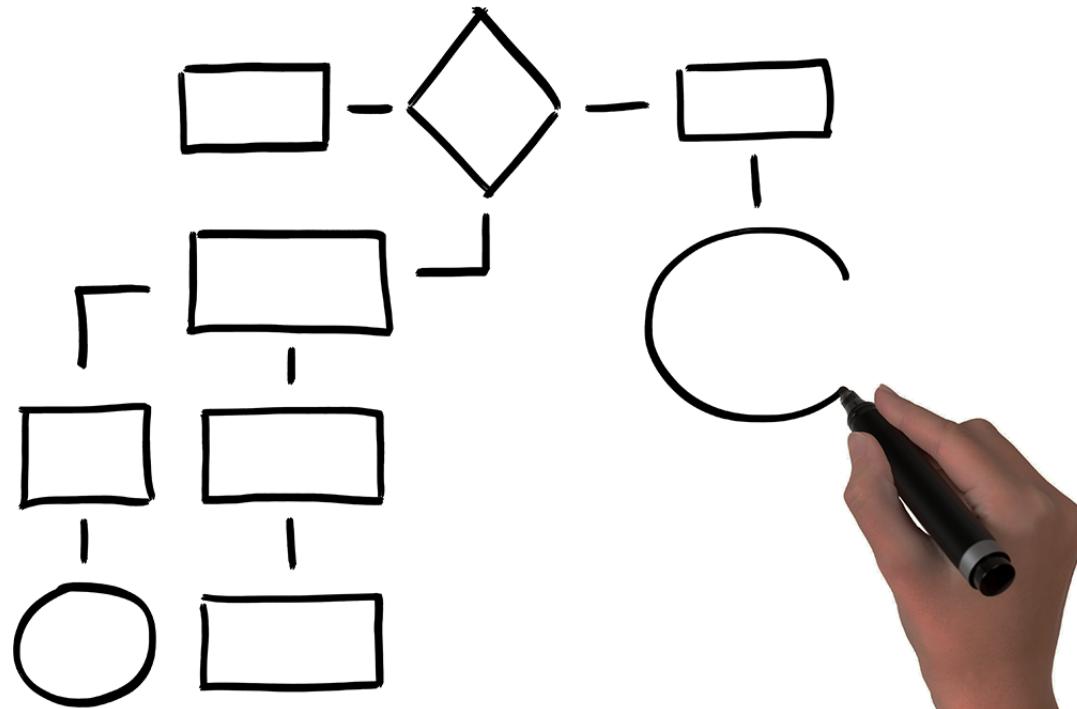
What did you **notice**?



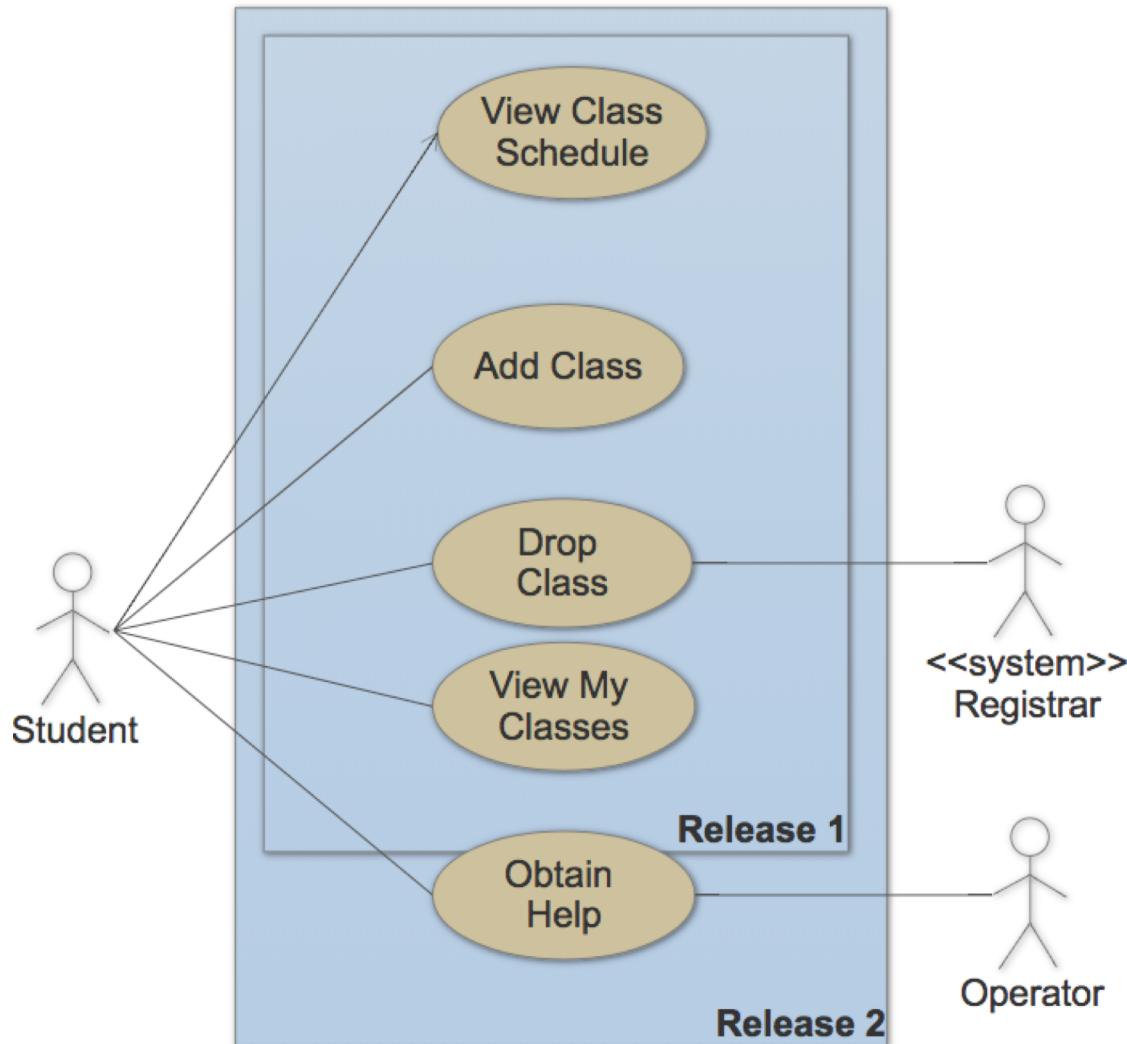
# CS Life Skill #2: “Architecture diagrams”

- **Big idea:**

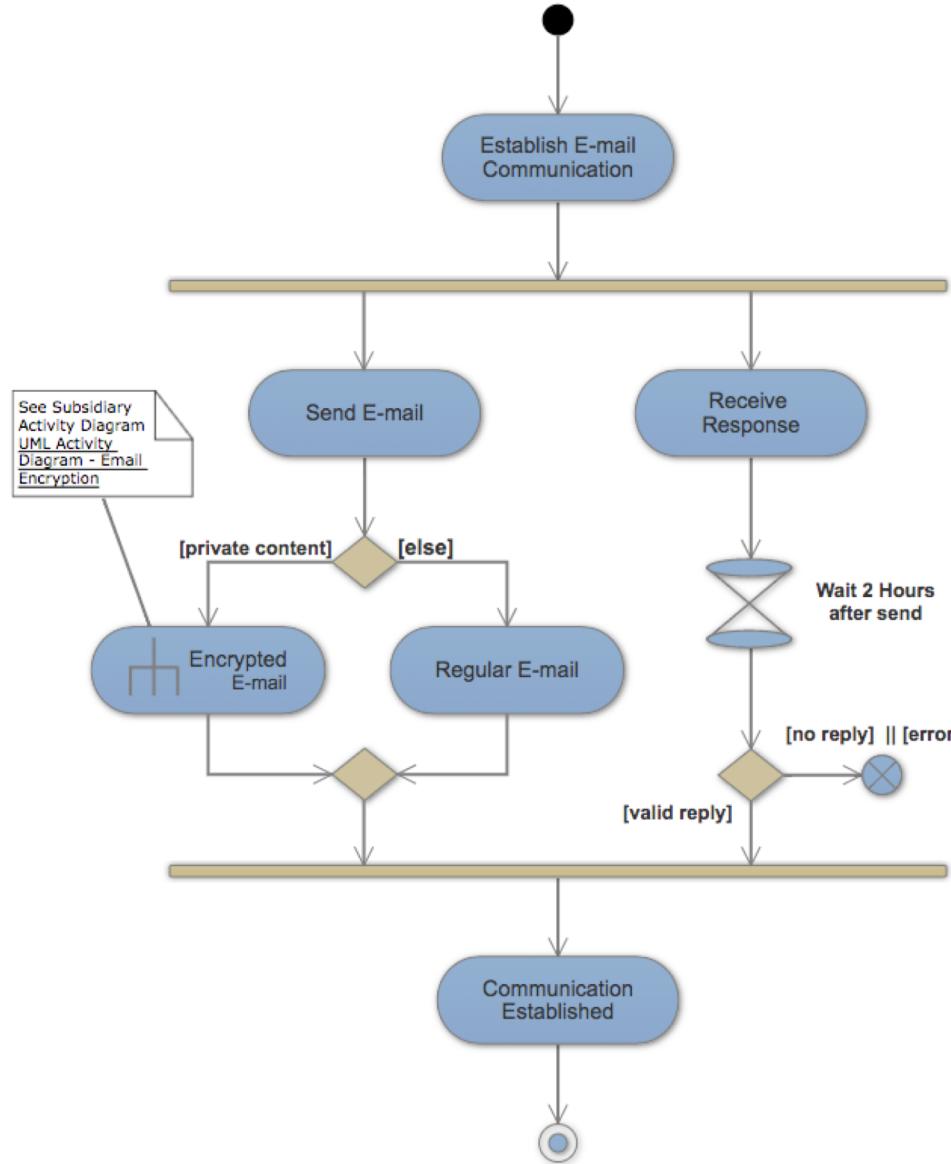
- Now that you’ve got an idea of what your interface might look like, break that down into **manageable pieces** so you can get started
- This can happen at **several levels of detail**



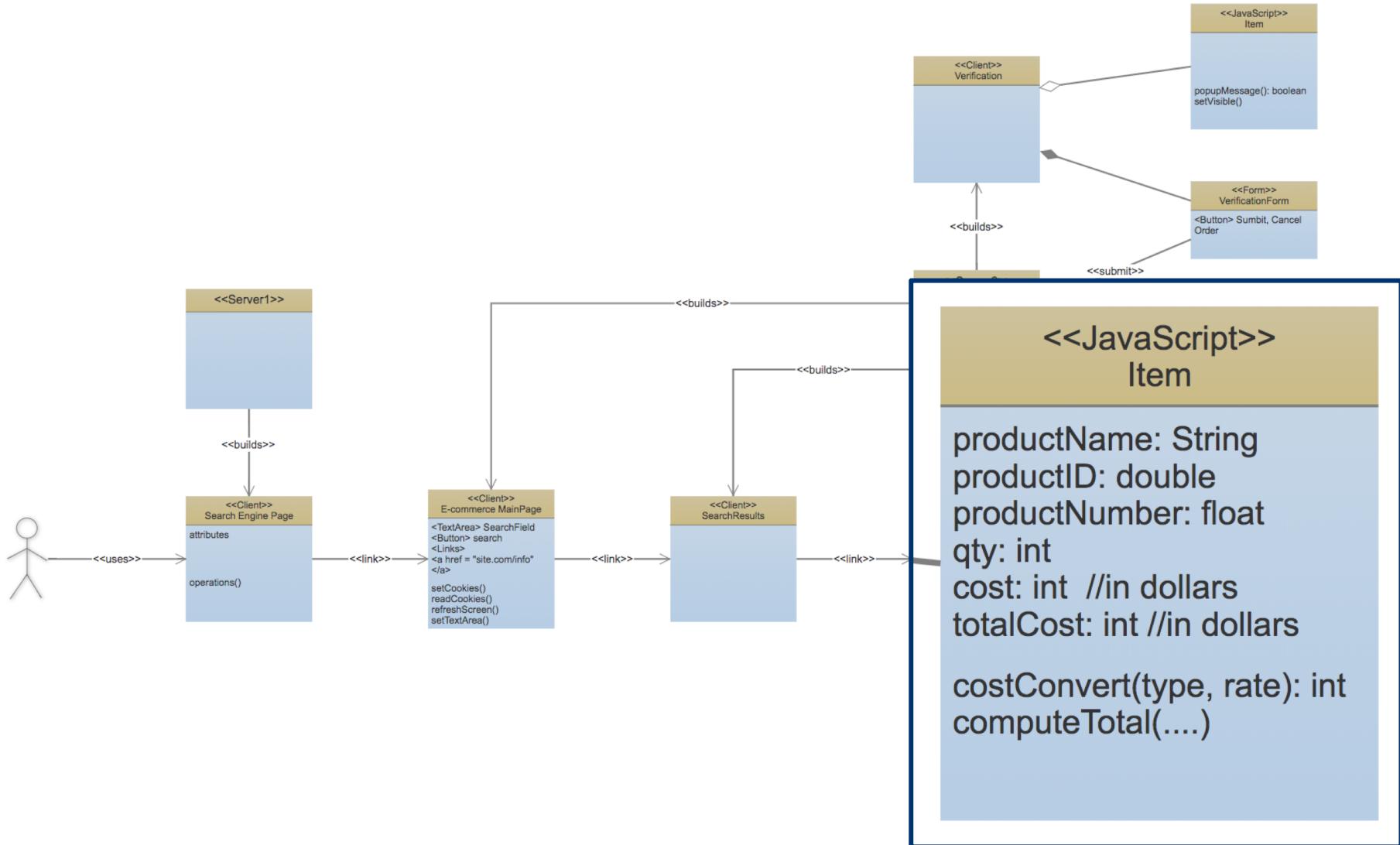
# Example: use case diagram (high level)



# Example: activity diagram (mid level)



# Example: class diagram (low level)



# Takeaways

- Thinking about your end user early → you're more likely to **build something that actually solves the problem**
- “**Low-fidelity**” prototyping saves time and energy by helping identify problems before you commit to code
- **Architecture diagrams** help you plan out your implementation so you don't run out of time
- Also, the process is **kinda fun...**

# Coming up

- No labs, assignments this week (Thanksgiving break)
- When we get back - **FP2: Prototype I:**
  - a persona
  - a paper prototype of your project
  - an annotated architecture diagram
- Monday 11/26: Introduction to Algorithms
- Wednesday 11/28: Working with Files
- Lab: Sorting
- Friday 11/30: Handling **Exceptions**