

Intro to Coding with Python— Prototyping

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (<https://jcrouser.github.io/>)

Plan for Today

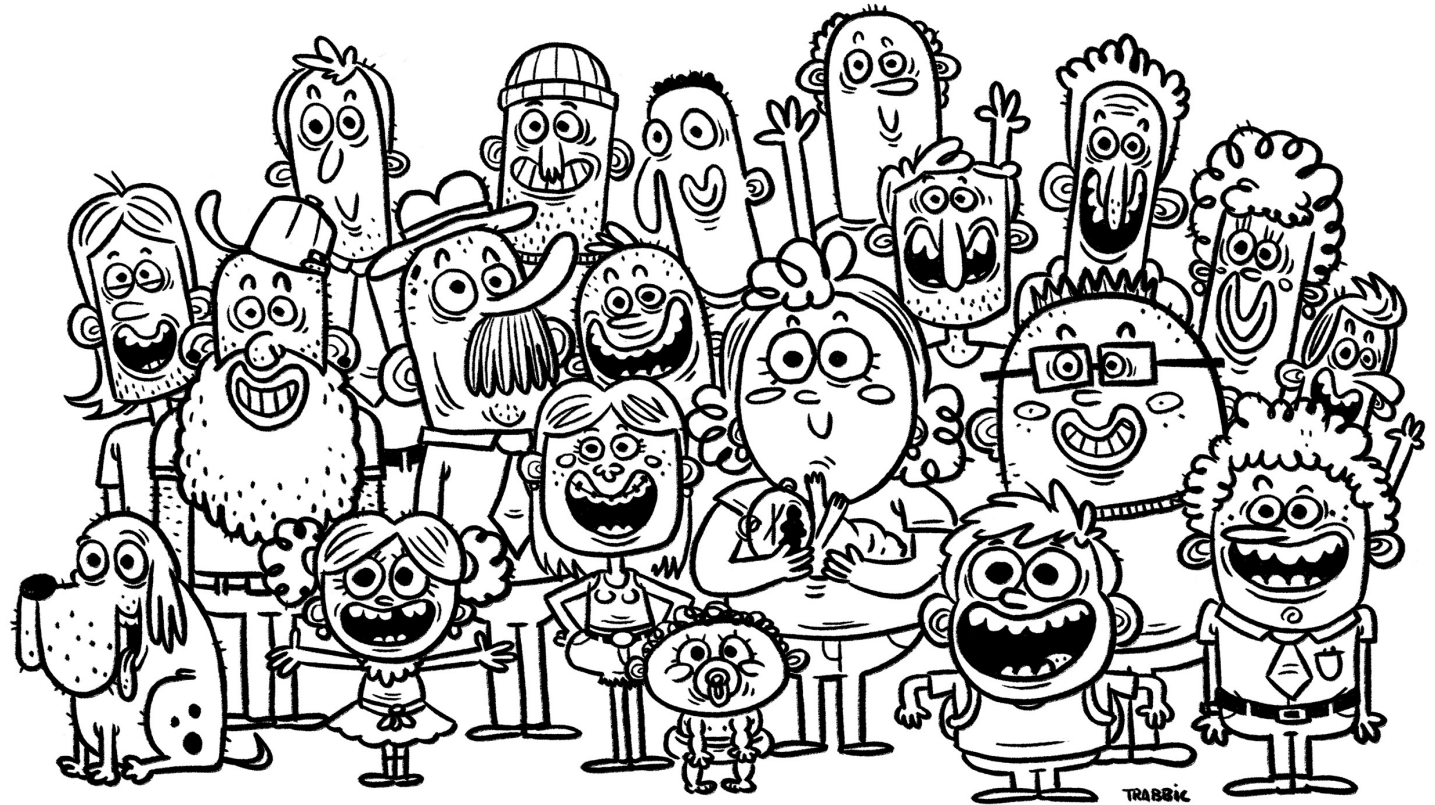
- User-centered design
 - What it is
 - Why do it
 - Ways to do it
- Life skill #1: paper prototypes
- Life skill #2: architecture diagrams

Hypothetical
example

“Advising Assistant”

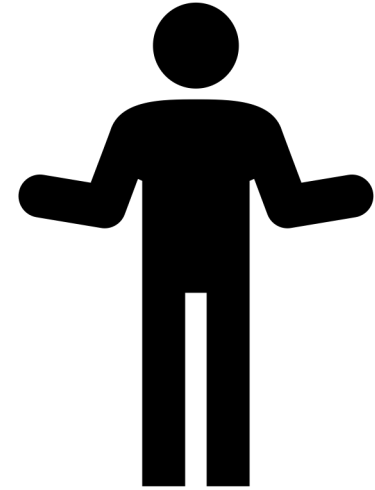
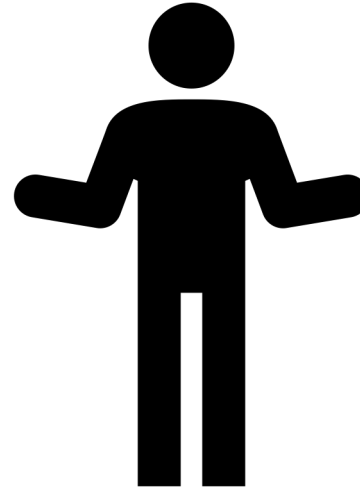
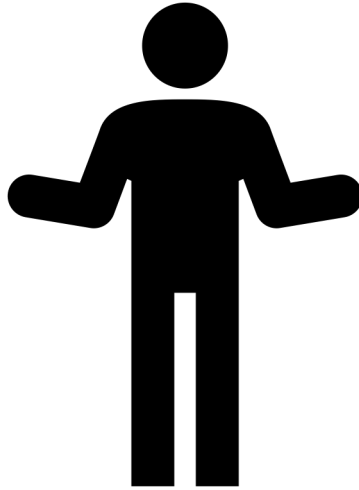
Hypothetical example

- **Overview:** over 100 majors and minors in DCIS



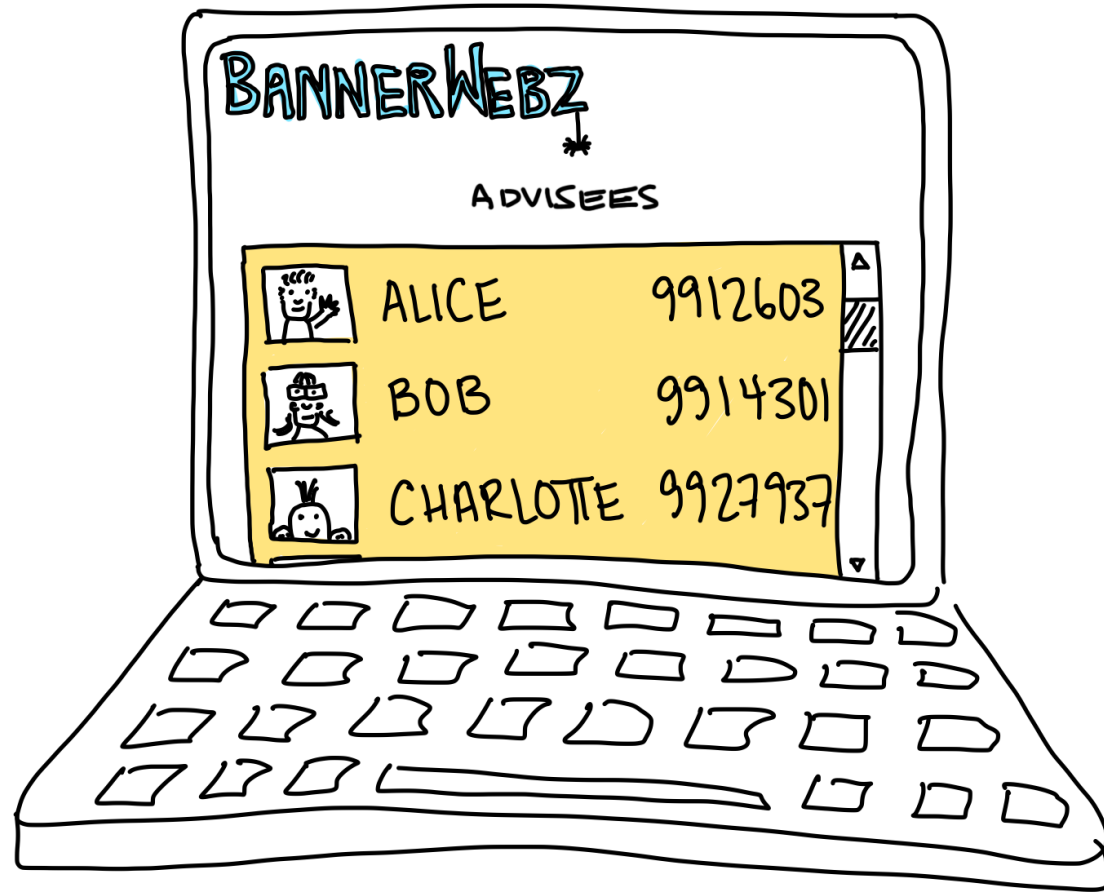
Hypothetical example

- **Overview:** three advising faculty



Hypothetical example

- **Overview:** BannerWeb not hugely helpful...

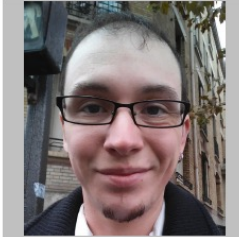


Hypothetical example

- **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		SEMESTER	REQUIREMENTS	NOTE
2020 99X			CSC111	
			CSC212	
			CSC231	
			CSC250	
			MTH111+	
			MTH153	
			Theory	
			Programming	
			Systems	
			Seminar	
			Elective / +4 Intro	

Officially Declared
Senior Certification
Honors Thesis?

☐
☐
☐

	2016	2017		2017	2018		2018	2019		2019	2020
FALL		Outside Major?			Outside Major?			Outside Major?			Outside Major?
		▼			▼			▼			▼
		▼			▼			▼			▼
		▼			▼			▼			▼
		▼			▼			▼			▼
SPRING		Outside Major?			Outside Major?			Outside Major?			Outside Major?
		▼			▼			▼			▼
		▼			▼			▼			▼
		▼			▼			▼			▼
		▼			▼			▼			▼

Transfer (inside major) 0 ▼

Transfer (outside major) 0 ▼

Total Credits: 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
 - **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

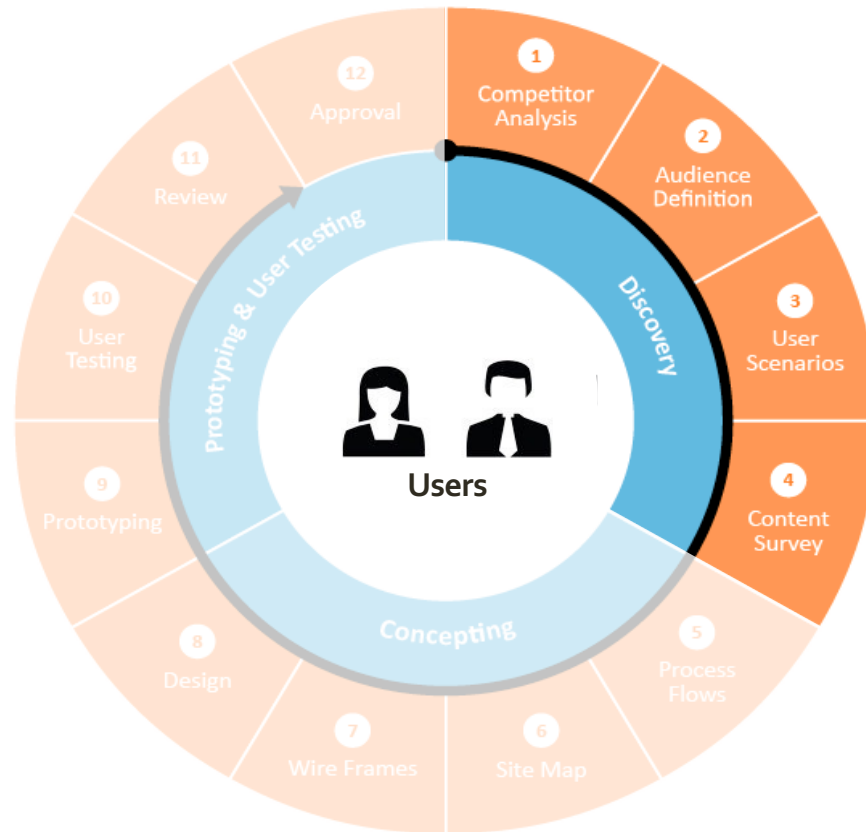
Hypothetical example

- **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

The table contains the following text (partially obscured by a diagonal line):

- screen Careers footer link was overlooked
- for New Graduates only shows a single job posting/description. Could not
- set in additional company info beyond what was provided - awards.
- ent info was only a side-show. Expected more information on culture, etc.
- ne chose to look at benefits information
- FAILURE in spite of having previously looked at the Company screen, PG
- ger participants did not self-identify with "Career" but brought about "job"
- ch & Filtering
- qualification was not matching expectations (i.e., planning technician job in
- se of keyword search
- word search with multiple terms it splits the terms apart. Not expected
- Location facet for USA - expected to be able to de-select all USA and then
- Location facet for USA - didn't like that panel list reset to the top of the list
- e participants who were older or with job experience ignored the Job Type
- function filters. Unsure what the job functions meant
- rence filters. User indicated that he was interested in project or program
- Descriptions: Wanted to know if jobs were current, and when, expected a
- Descriptions: Content type
- Location Process
- Apply vs Create Account - reason for both is not clear
- ot recognize from email that there was another step to finish the application
- context: job application confirmation email - videos would have been more
- application Which fields are required and how to get into the edit state of the
- application Expected resume to populate the various fields in the application
- application extra click to get to password requirements
- application phone and email contact info: what did "preferred" mean? a)
- application: fax number field outdated
- application: found military section and personal site selections caused
- application: the optional Locations and Departments sections caused
- application: expected to be able to prioritize locations and areas of interest
- application: not clear in the flow whether fields I skipped because they
- application: Qualifying questions unnecessary
- mination of application opens new tab
- ming Applicant - went back to their email to find where to log in
- ming Applicant - went for the client log-in in the top right
- ming Applicant - First looked to the jobs applied table and resume list for
- Out hits present on the Careers home screen
- Not in state not apparent
- 1 participant had issue reading the CAPTCHA; no issue signing it because

• 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



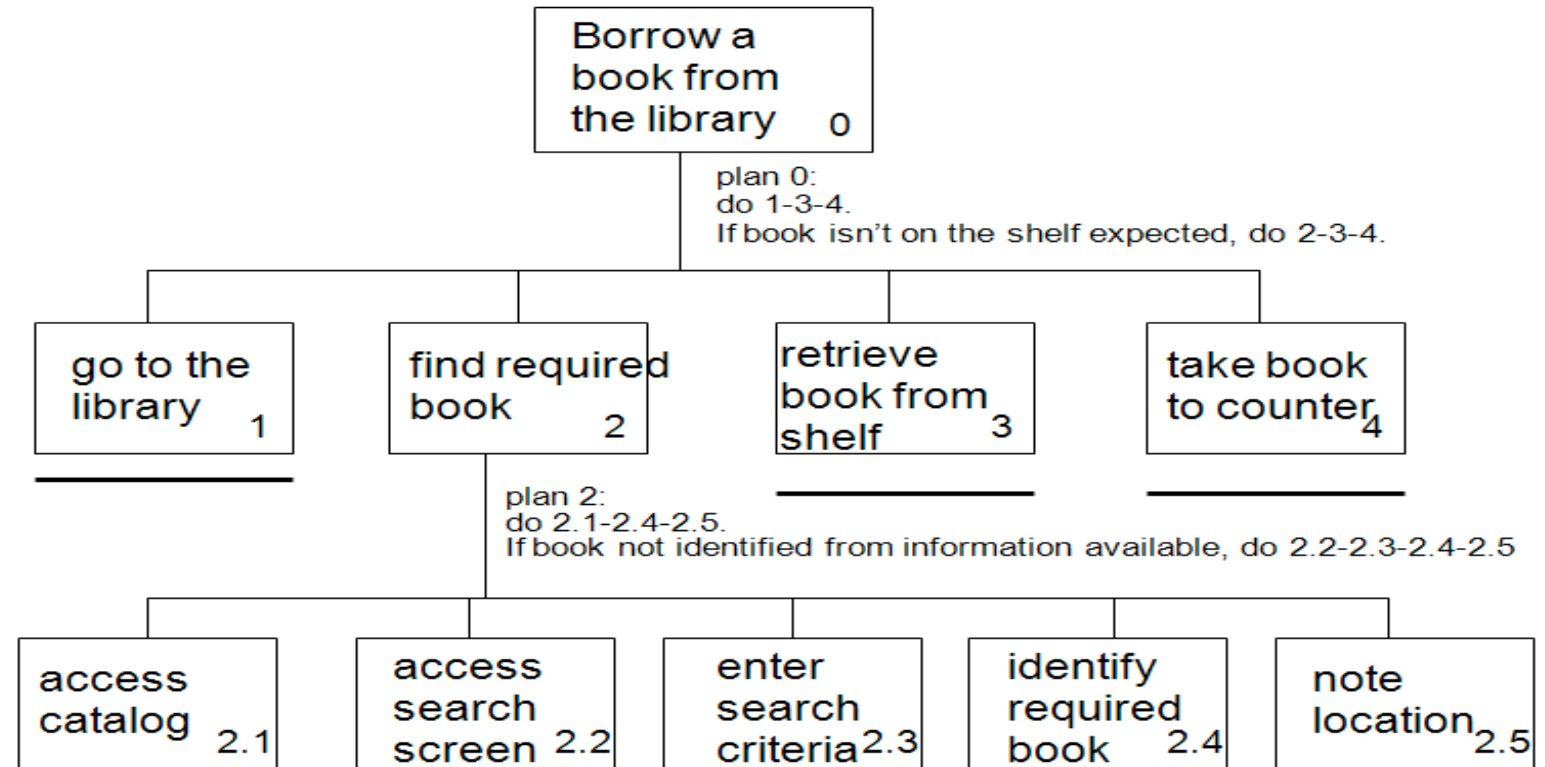
Defining your audience

- 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

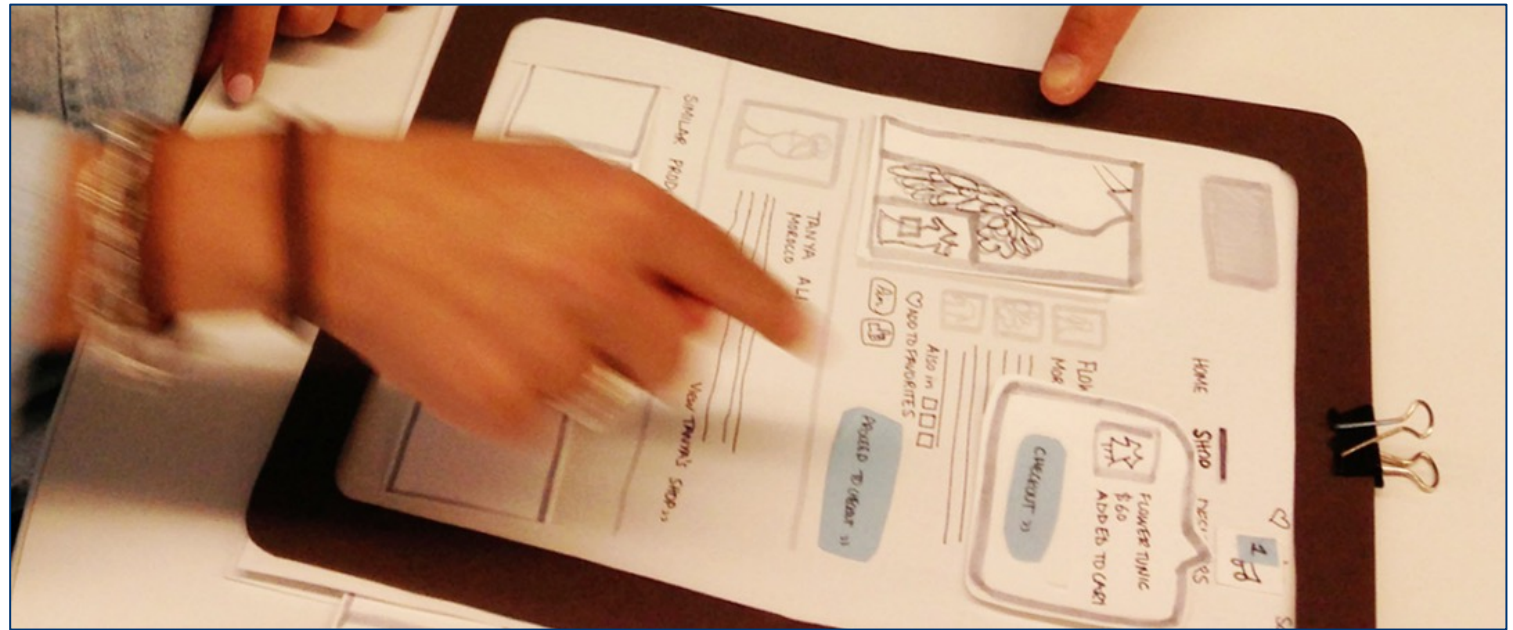
Goal: come up with **3 personas** that characterize people who might be interested in a public transit app



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!

Discussion

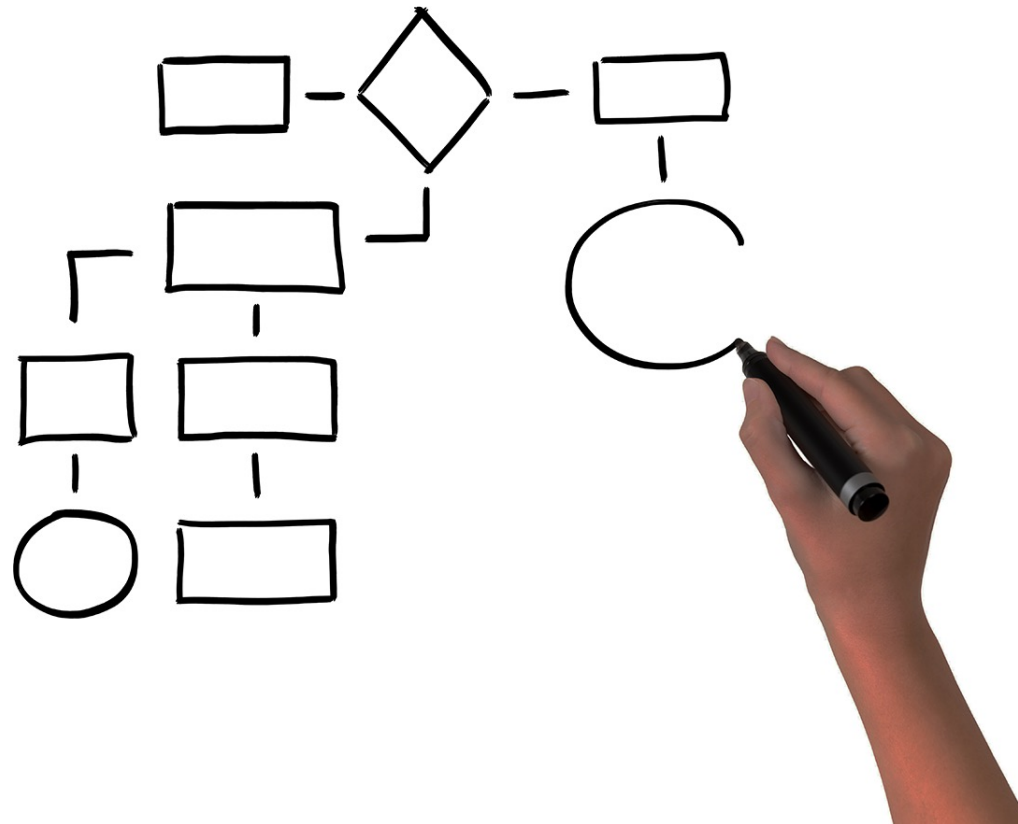
How did it go?
What did you **notice**?

CS Life Skill

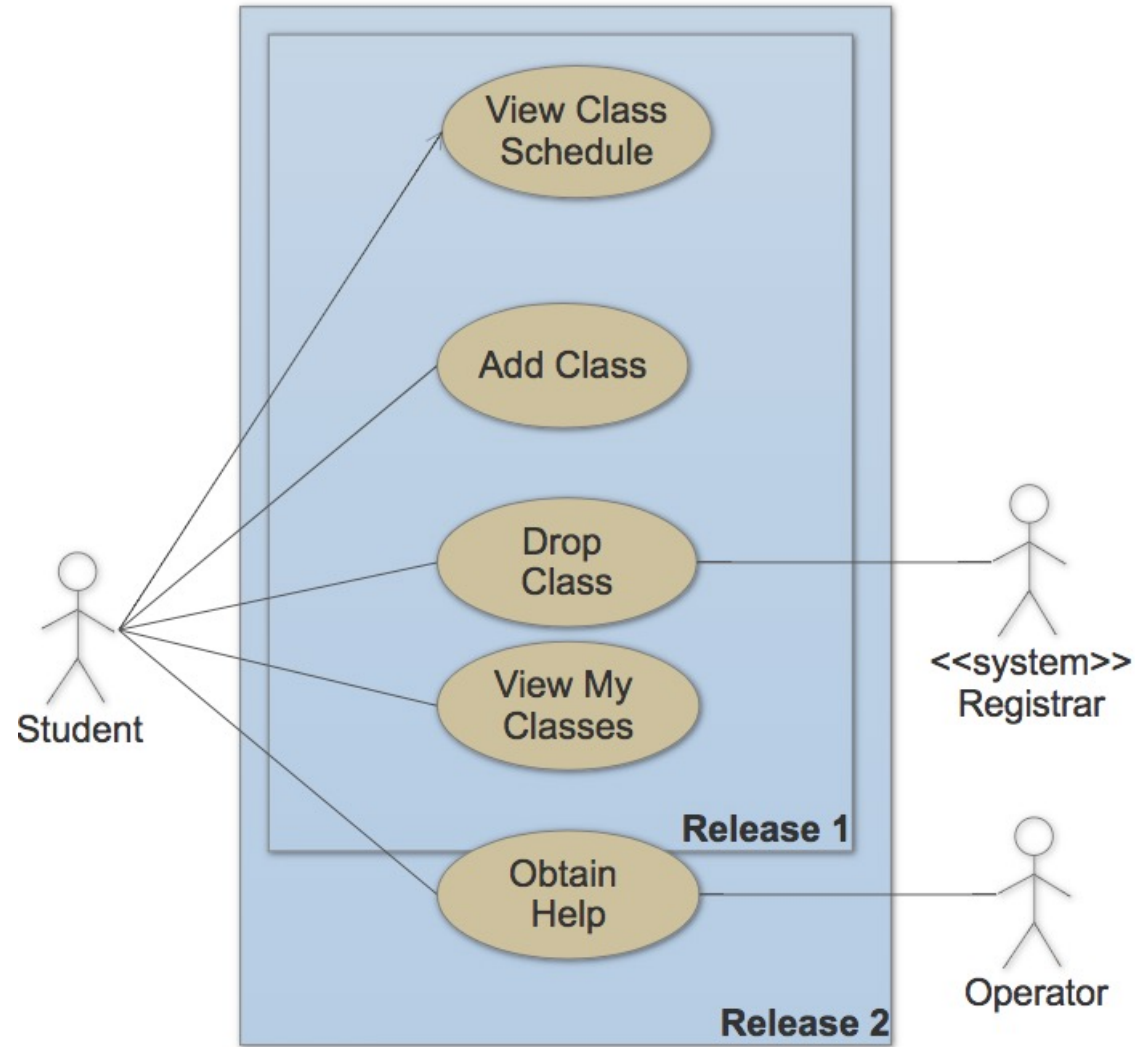
#2: "Architecture diagrams"

- **Big idea:**

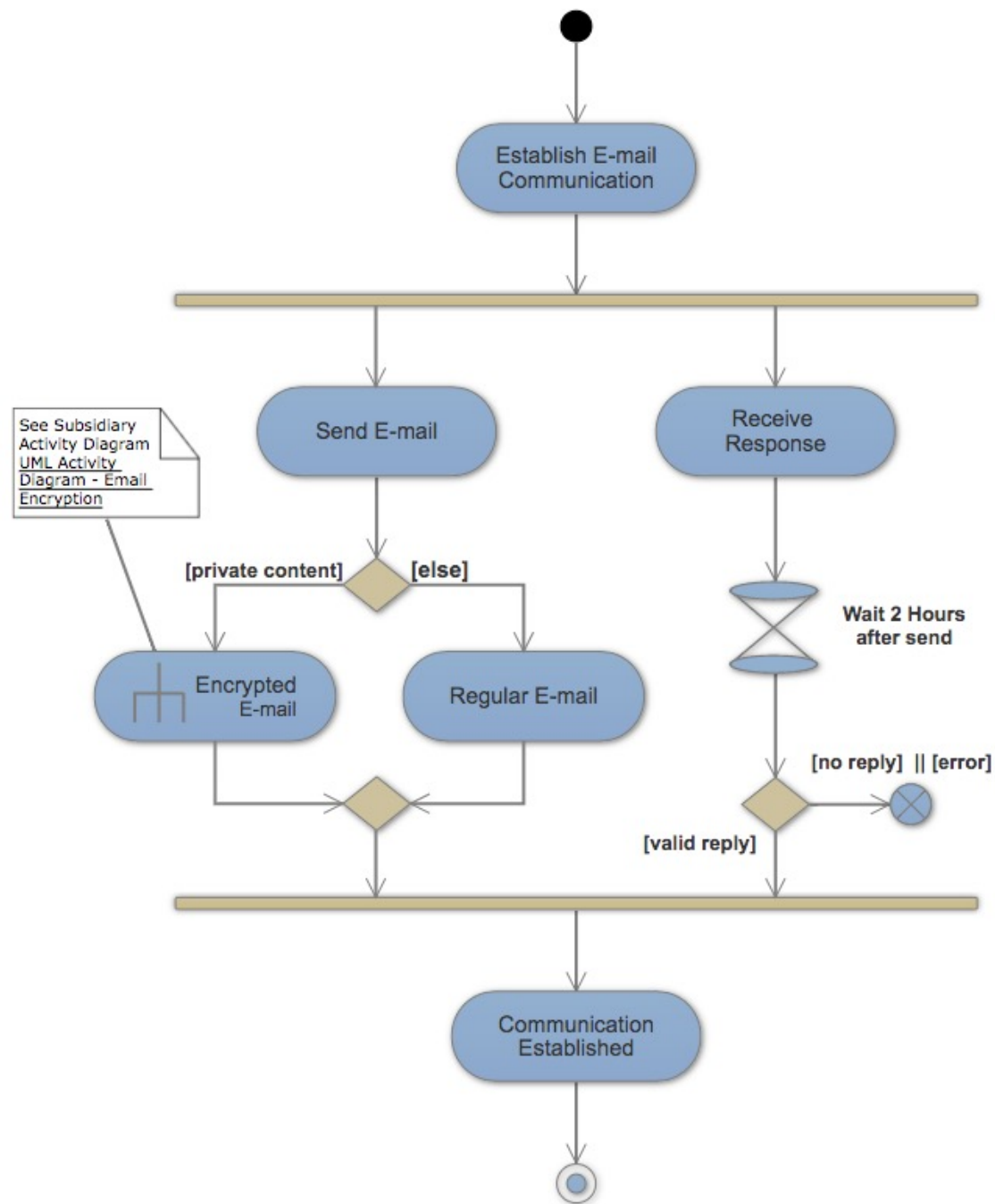
- Now that you've got an idea of what your interface might look like, break that down into **manageable pieces** so you can get started
- This can happen at **several levels of detail**



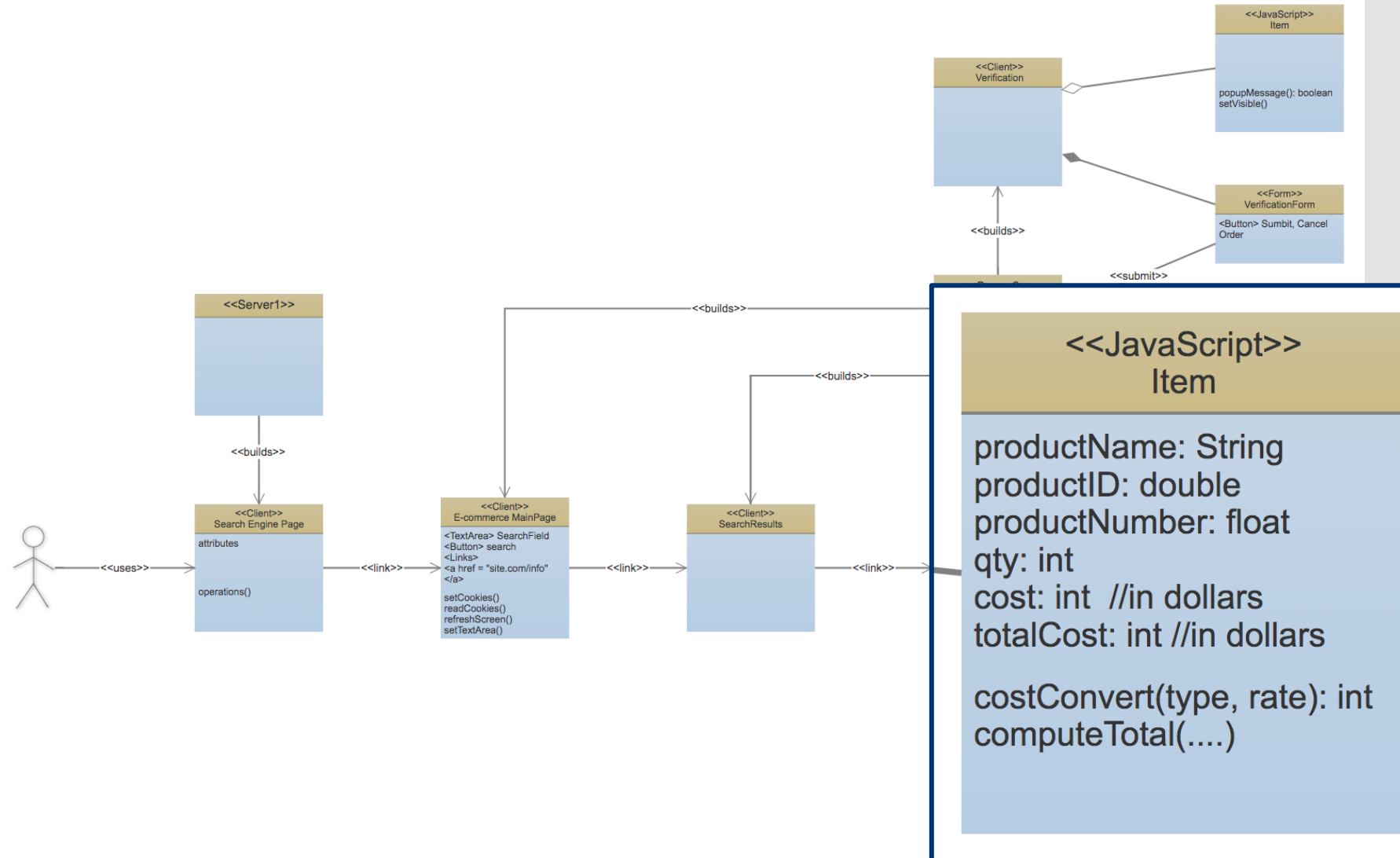
Example: use
case diagram
(high level)



Example: activity diagram (mid level)



Example: class diagram (low level)



Your turn!



Work with a partner to create a paper prototype for a transit app. Take a picture of your prototype to turn in on PLATO.

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...**