# Visual Analytics– Prototyping

Dr. Ab Mosca (they/them)

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

# Reminders

- No Class Tuesday (11/12) because it is Cromwell Day

- Final Project released today!

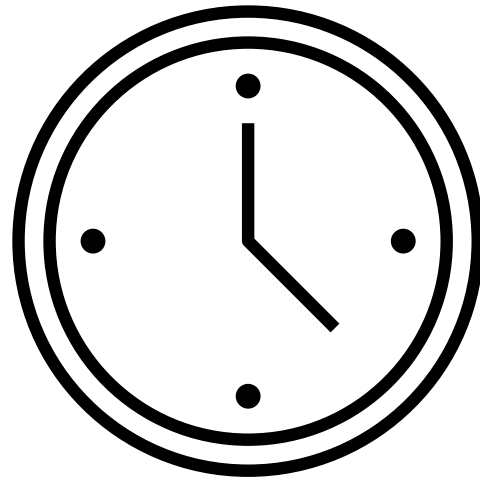- Be prepared to perform prototype testing for your project *in class on 11/14*

# Plan for Today

- User-centered design
  - What it is
  - Why do it
  - Ways to do it
- Paper prototypes
- Architecture diagrams

# Hypothetical Example
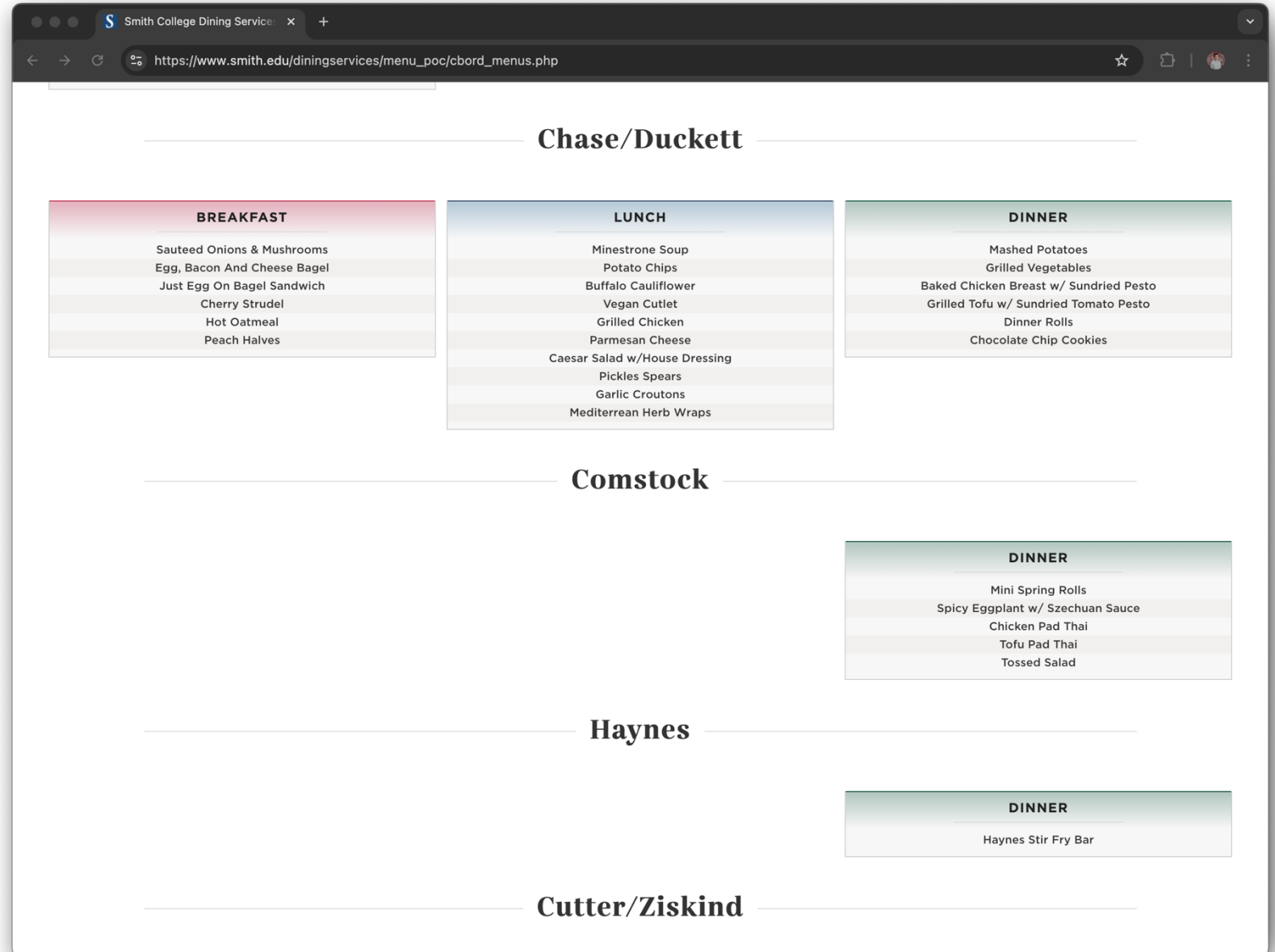
# "Where should I get lunch?"

**Overview:** ~2500 students to feed, 75 min lunch block, 10 dining halls


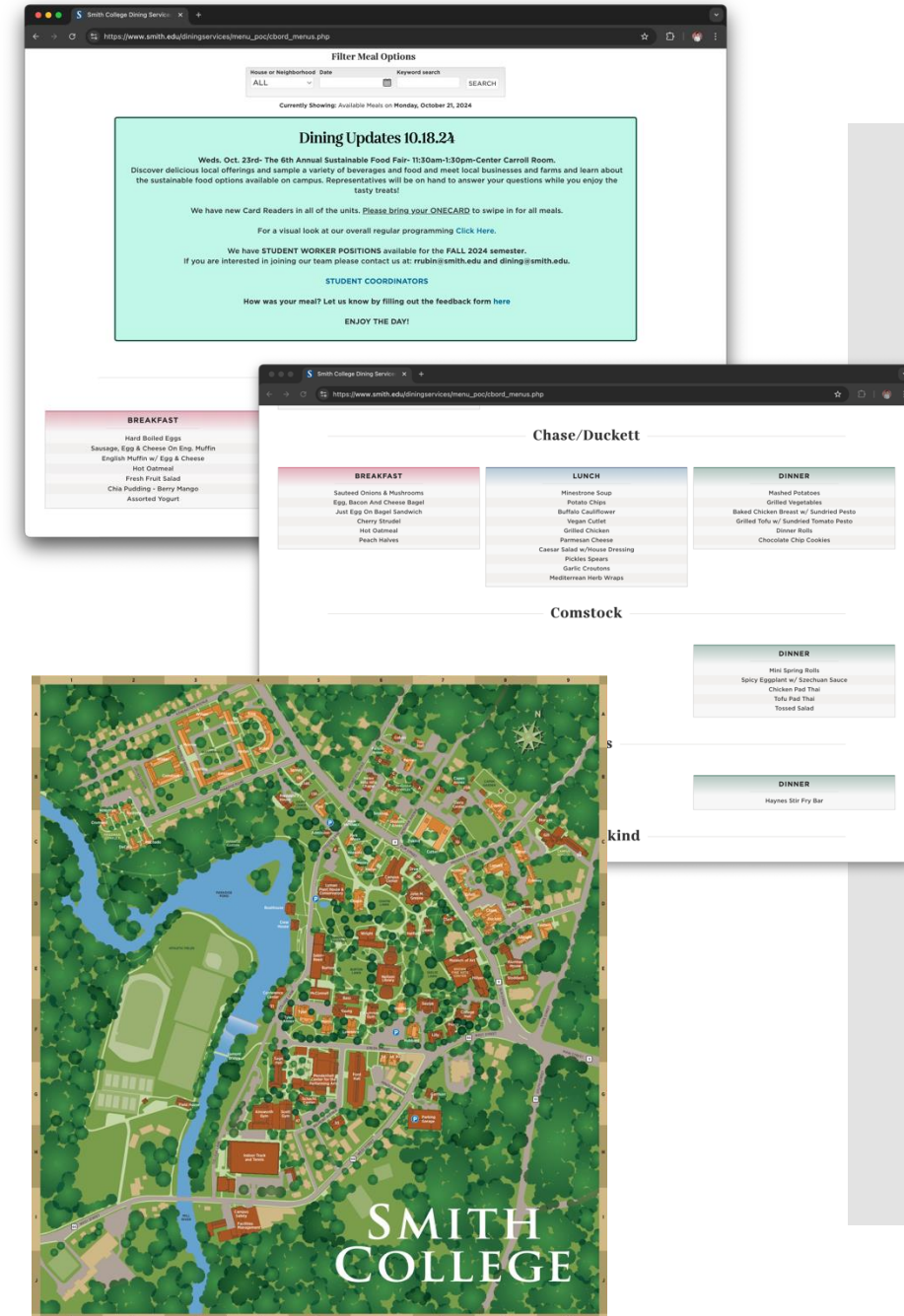
# Hypothetical example

# Hypothetical example

- **Overview**: Menu page is not very helpful...

# Hypothetical example

- **Overview**: current process is manual

- **Goal 1:** Automatically show only open dining halls

- **Goal 2:** Give an overview of locations and repeat menus

- **Goal 3:** Show how busy open dining halls are

# Hypothetical example

- **Ideal:** we would like to build an application that pulls data from the menu page, campus map, and location data to show open dining halls, menus, and business

- **Issue:** Big brother vibes, privacy concerns, disparate data sources

- **Strategy:**
  - Scrape data from menu page
  - Find shapefiles for campus map
  - Combine data sources
  - Build authenticated front-end for current Smithies

# Hypothetical example

- **Ideal:** we would like to build an application that pulls data from the menu page, campus map, and location data to show open dining halls, menus, and business

- **Issue:** Big brother vibes, privacy concerns, disparate data sources

- **Strategy:**
  - Scrape data from menu page
  - Find shapefiles for campus map
  - Combine data sources
  - Build authenticated front-end for current Smithies

# Discussion

Let's say you want to implement this front-end; **where do you start**?

# User-centered design framework
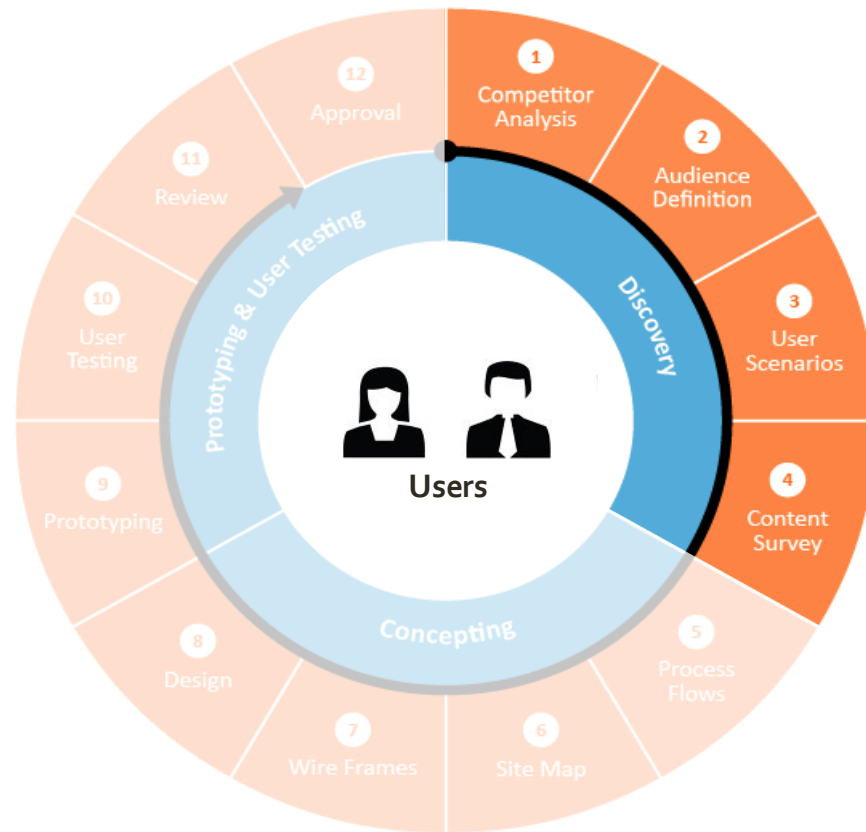


**1) Discovery**
- Learning about your users
- Modeling your users
- Analyzing your users' tasks
- Eliciting and defining clear product requirements

**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

http://www.uxmatters.com/mt/archives/2010/07/design-is-a-process-not-a-methodology.php

# Discovery: Competitor Analysis

- **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

# Discovery: Competitor Analysis

- W
- Ho

Identify opportunities and design constraints
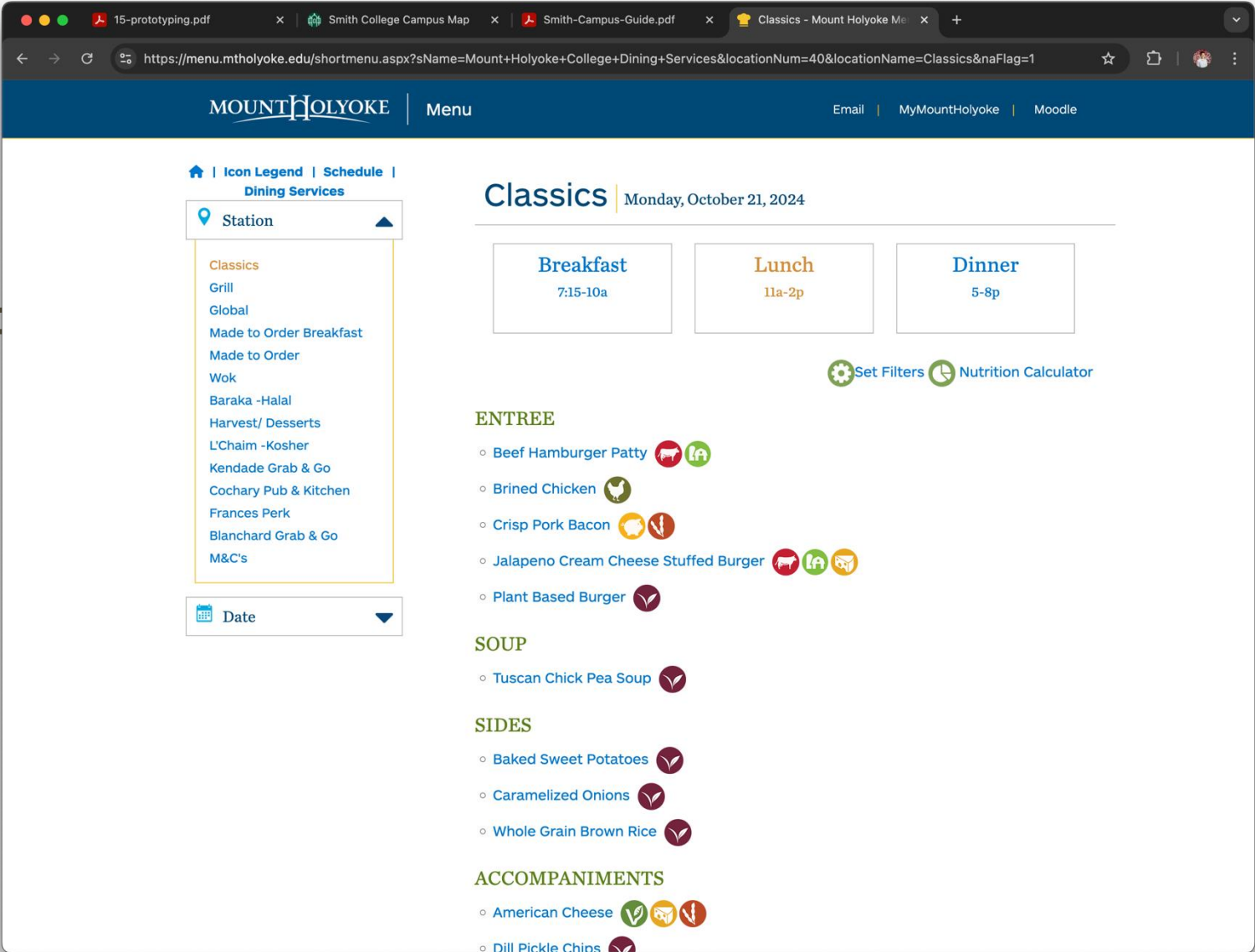
# Discovery: Audience Definition

- Learning about their problem
  - Semi-structured interview

- Analyzing their tasks
  - Hierarchical task analysis

- Modeling users
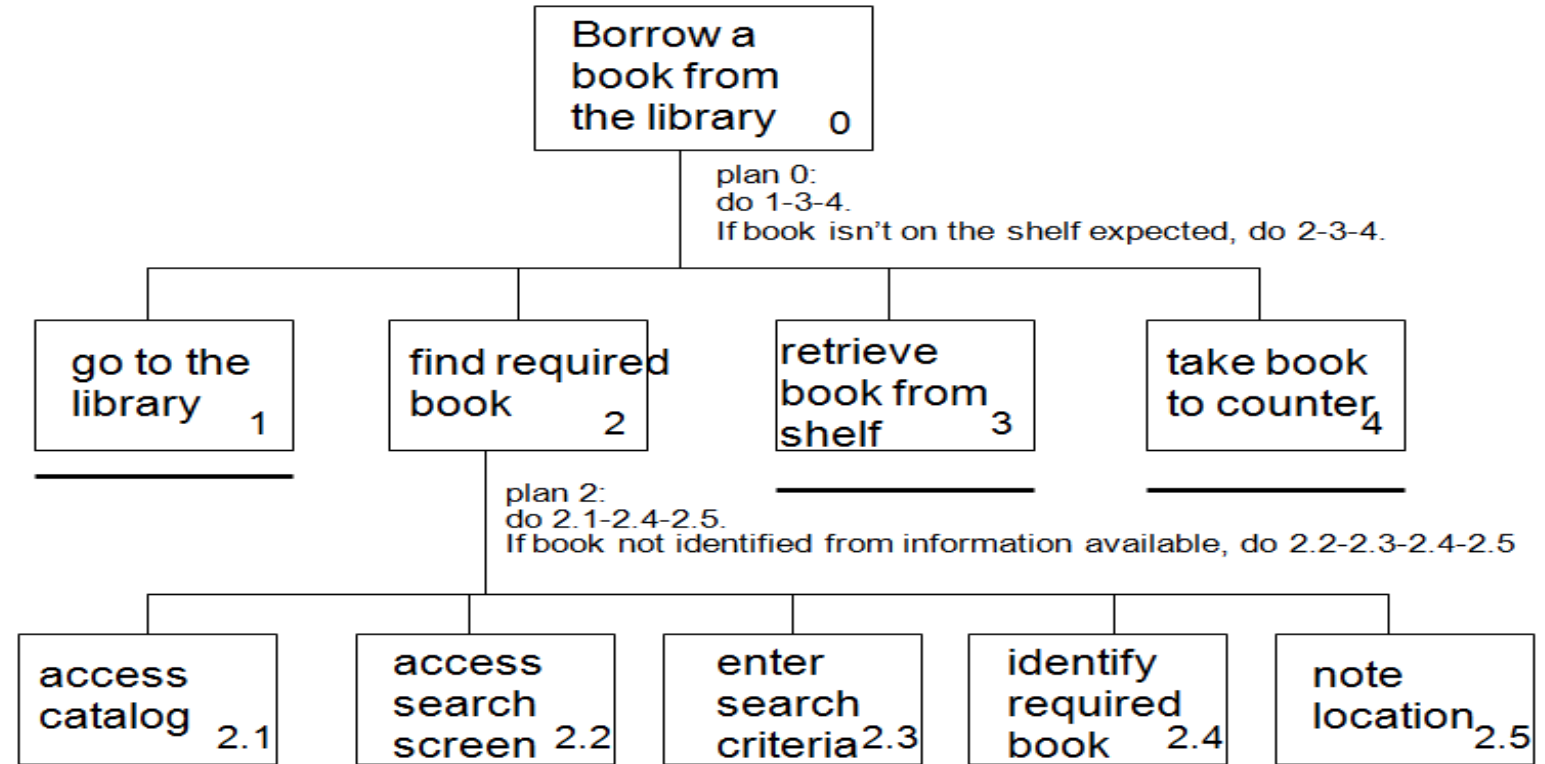  - Personas

# 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



Wood, L. E. (1997). Semi-structured interviewing for user-centered design. *interactions*, 4(2), 48-61.

# 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:

Hackos, J. & Redish, J. (1998). User and Task Analysis for *Interface* Design. Chichester: Wiley.

# Task analysis example



Borrow a book from the library — 0

plan 0:
do 1-3-4.
If book isn't on the shelf expected, do 2-3-4.

go to the library — 1

find required book — 2

retrieve book from shelf — 3

take book to counter — 4

plan 2:
do 2.1-2.4-2.5.
If book not identified from information available, do 2.2-2.3-2.4-2.5

access catalog — 2.1

access search screen — 2.2

enter search criteria — 2.3

identify required book — 2.4

note location — 2.5

# 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

# Personas

Example for the dining app

- Ellis is a first year Smithie who lives in in Wilson. They do cross country and have 6am practice before their 9:25am art history class in the art museum on MWF. On TR, they have engineering 101 in Ford Hall until 12:05, and their work study job at the campus school that starts at 12:45pm sharp. Ellis is a "live to eat" kind of person, and likes to optimize the "yum" factor in their meals. In addition, they are vegan. They are comfortable using apps on their phone like GoogleMaps, but are struggling to find the time to survey dining hall menus and get to the appropriate dining hall for the meal they want between their various other activities.

Pain Point: Limited time

Goal/Need: Eating what they like

Pain Point: Limited time

Pain Point: Distilling current menu list

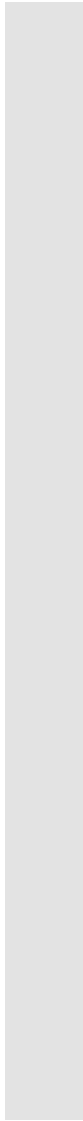Technical skills: Comfortable with basic apps

Pain Point: Walking far & fast

Goal/Need: Minimize walk distance
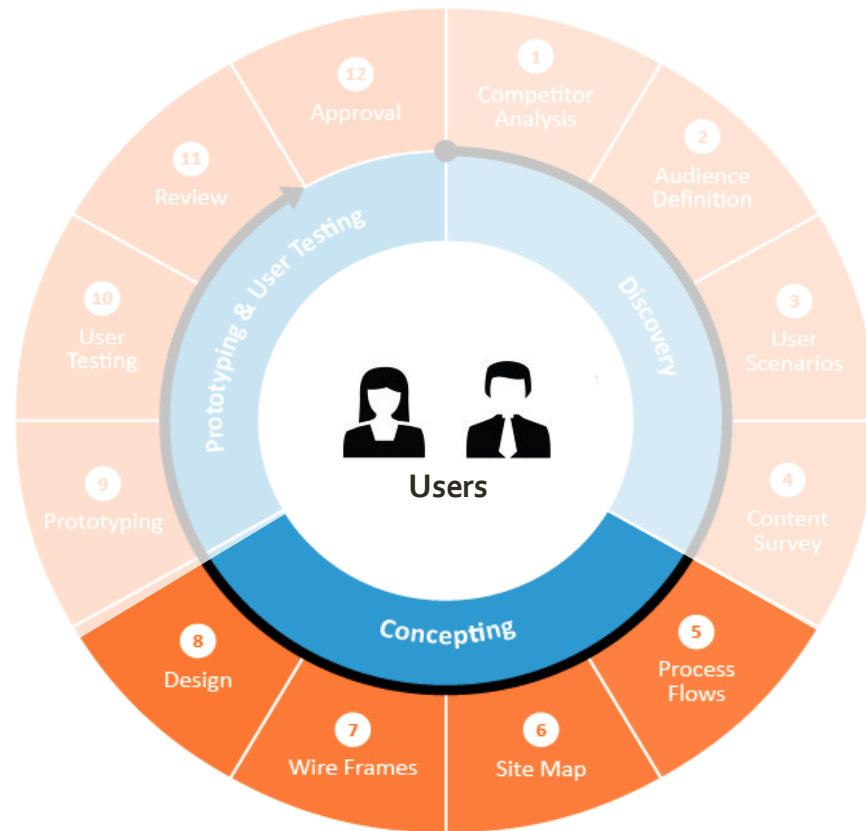
# Activity: personas

**Goal**: come up with **a persona** that characterizes a user of a visual analytic system for exploring energy usage on campus.

Now that we've got some end users in mind, what would the system look like?

# 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
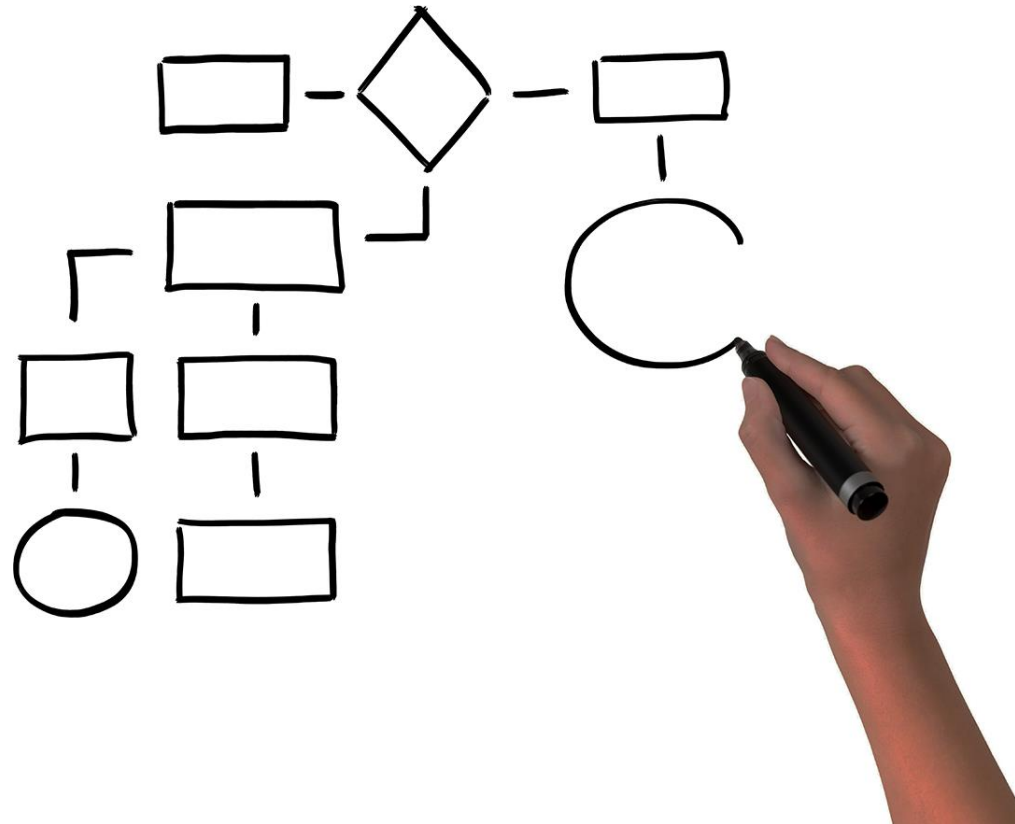- Solving design problems through ideation
- Detailed design activities

**3) Prototyping + User Testing**
- Delivery of a high-quality product that meets users' needs and is easy to learn and use

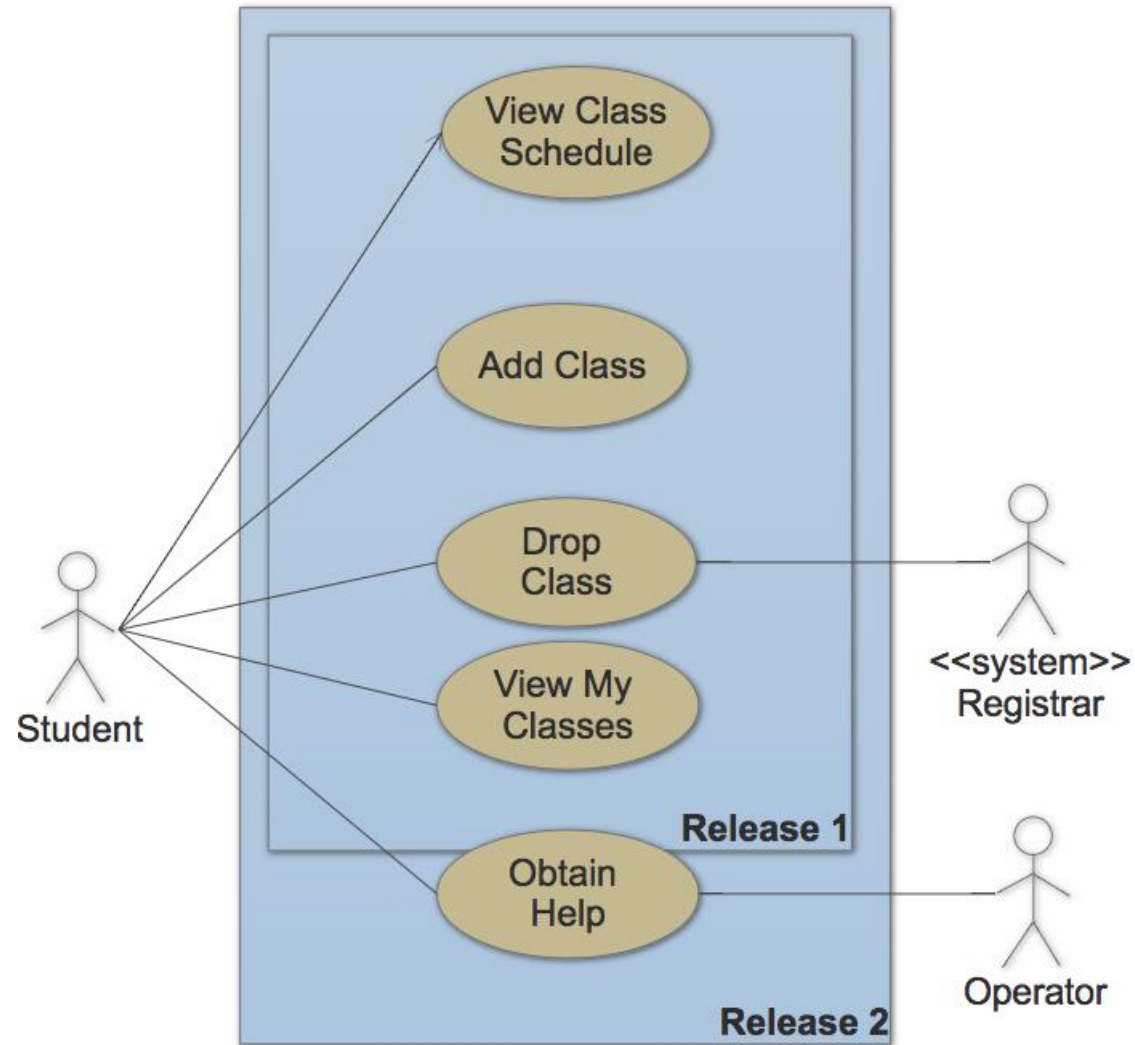http://www.uxmatters.com/mt/archives/2010/07/design-is-a-process-not-a-methodology.php
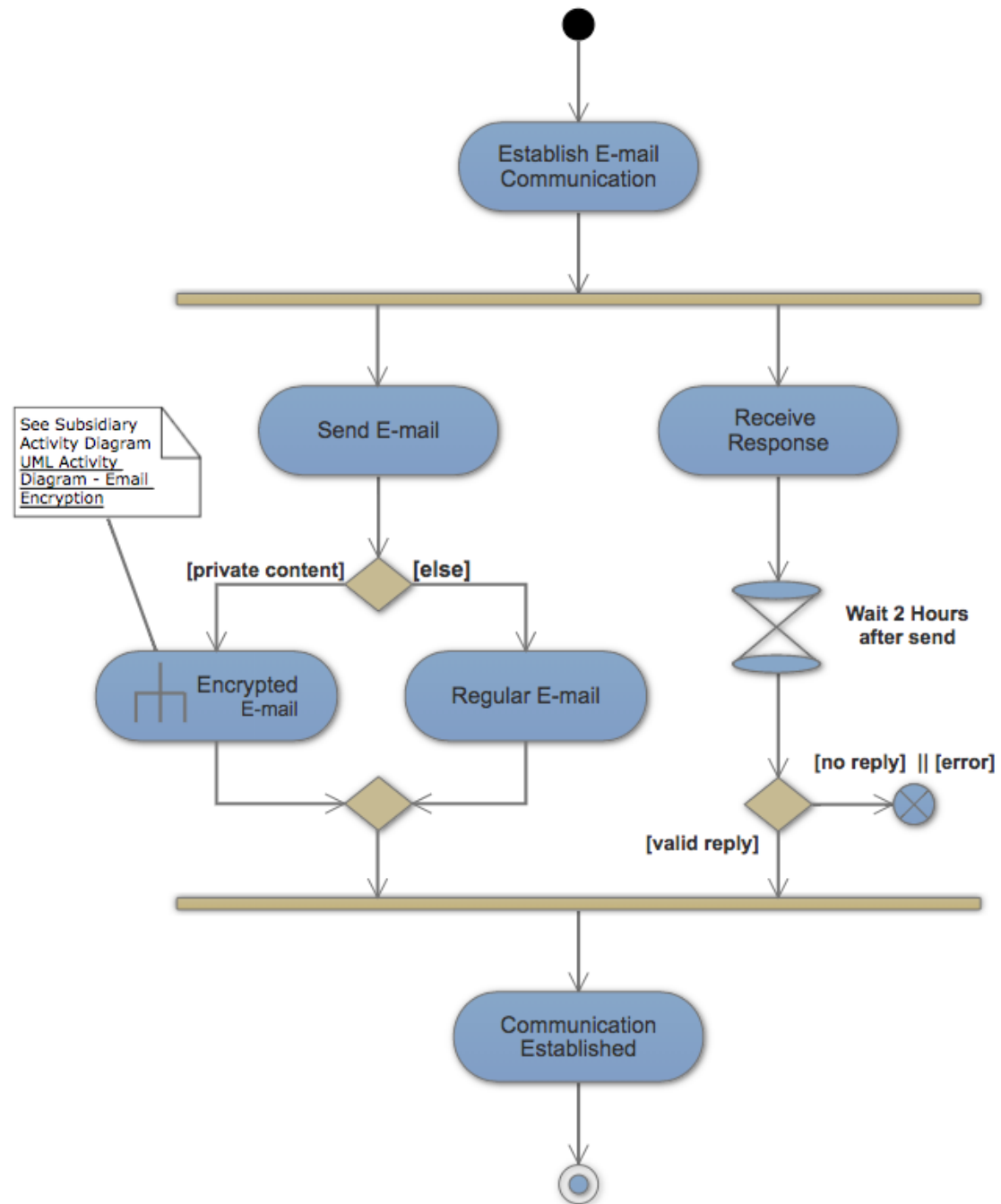
# Concepting: Architecture diagrams

- **Big idea:**
  - Now that you've got an idea of what your too should do, 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)
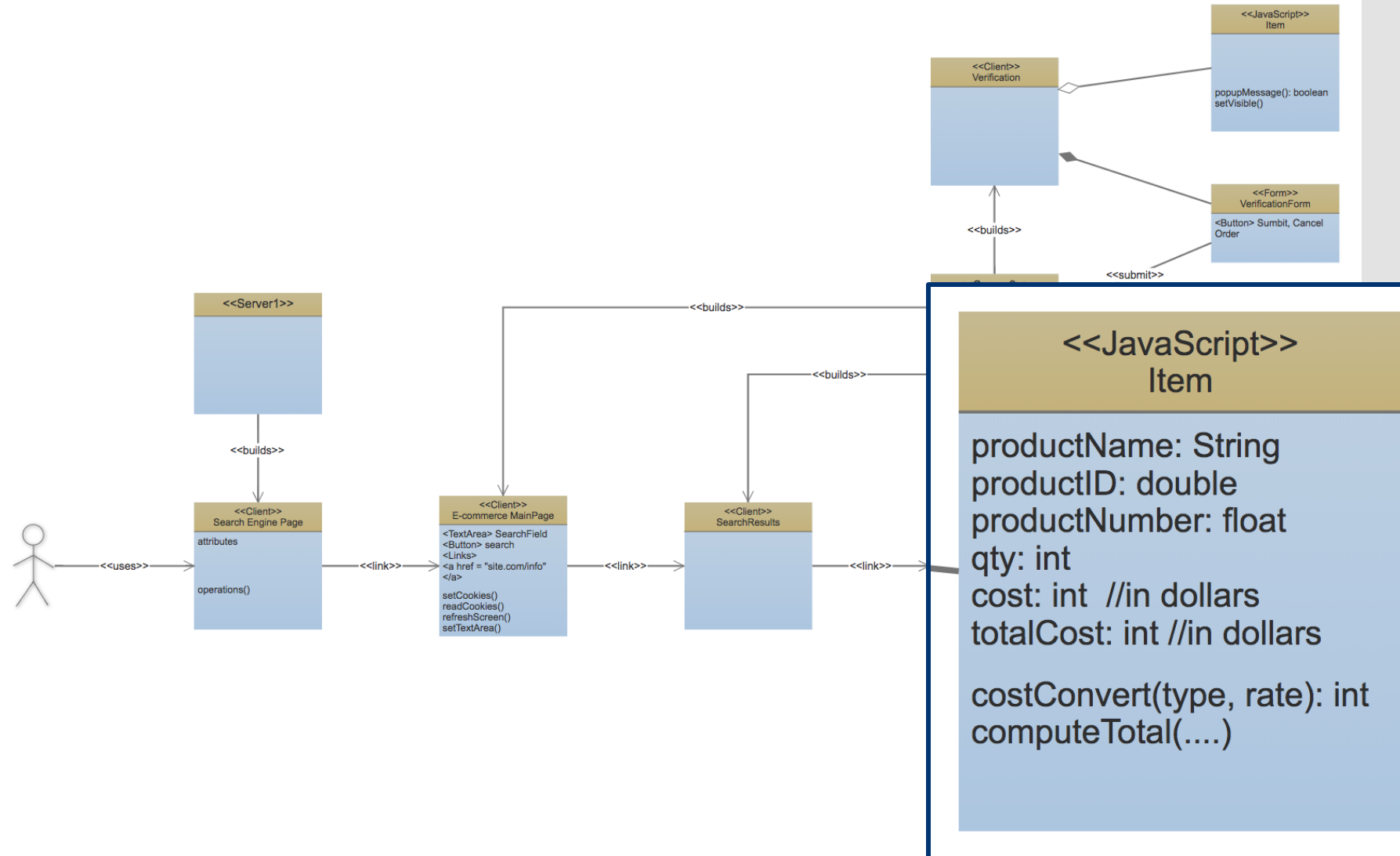
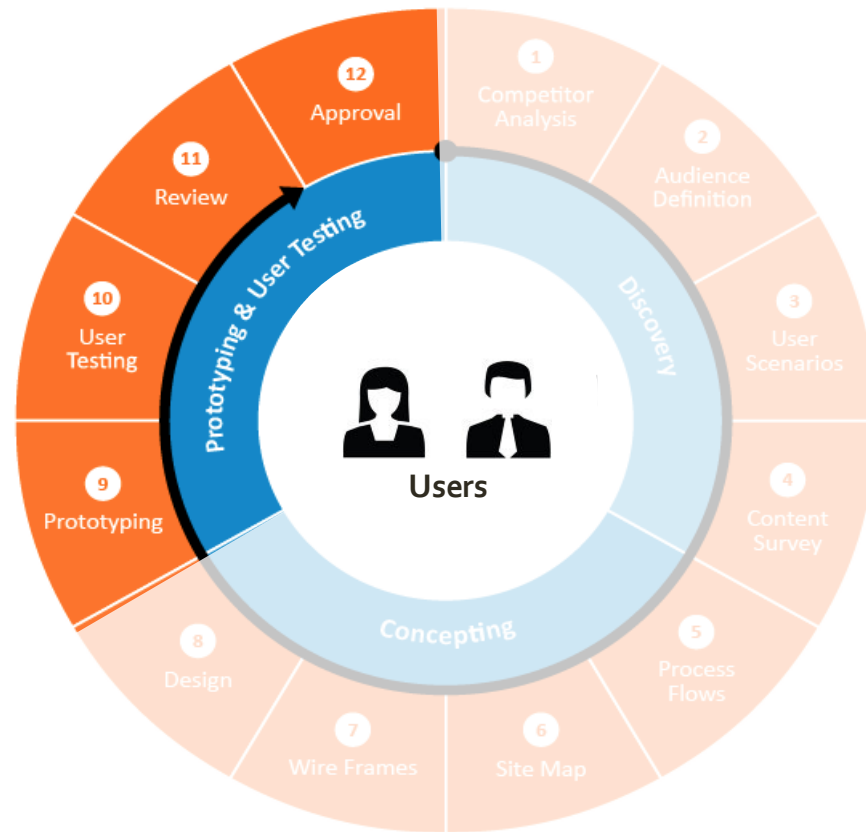# Example: class diagram (low level)

# Diagram Practice

- What would the case diagram look like for the dining hall app?

- What about an activity diagram?

# 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

http://www.uxmatters.com/mt/archives/2010/07/design-is-a-process-not-a-methodology.php

# Prototyping and Testing: Low-fidelity 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!
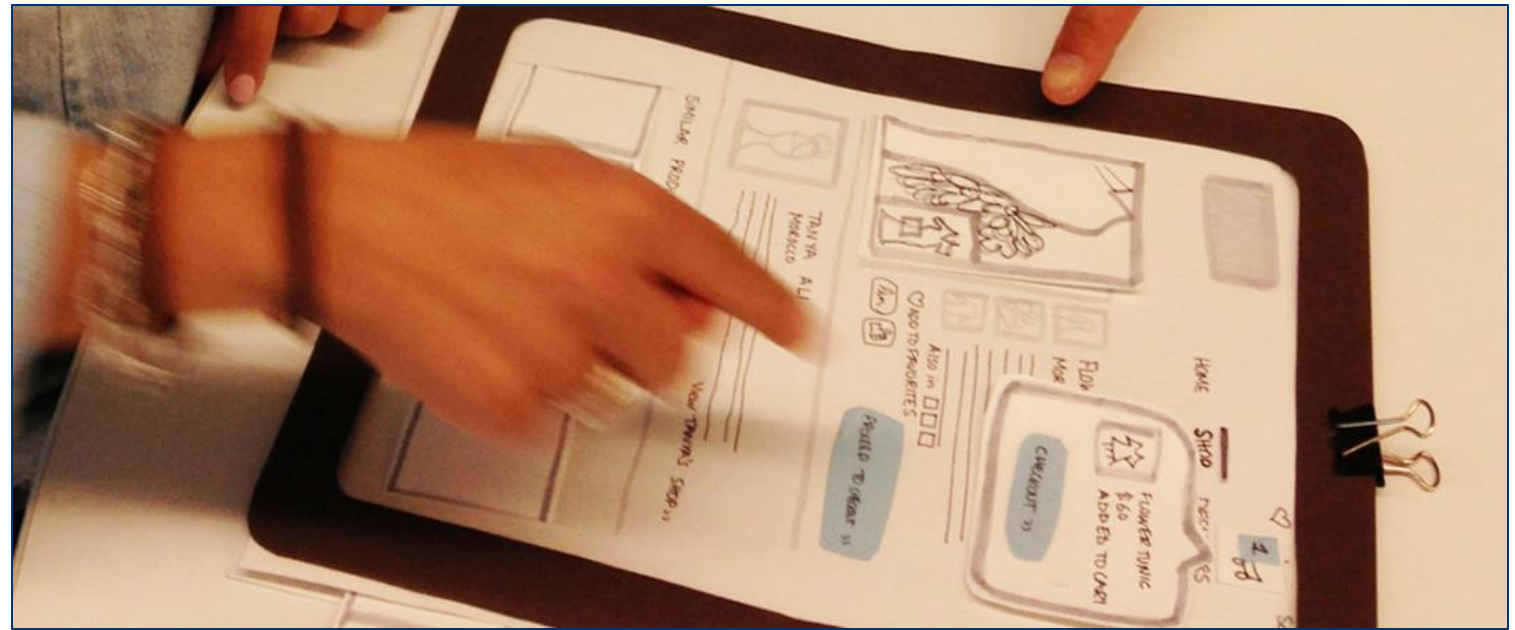
# Prototyping and Testing: Low-fidelity paper prototyping

- 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!

# Prototyping and Testing: Low-fidelity paper prototyping

Examples:

- https://www.youtube.com/watch?v=nA9QP9lkl2o
- https://www.youtube.com/watch?v=y20E3qBmHpg
- https://www.youtube.com/watch?v=yafaGNFu8Eg

# Prototyping and Testing: Soliciting feedback

- One purpose of a prototype is to get feedback on your design idea. We do this through **user testing**:
  - Choose specific tasks your end user should be able to do with your app
  - Ask someone to perform those tasks with your prototype
    - Do not give clues or help while they perform each task
    - Ask the tester to "think out loud" i.e. narrate what they are doing / why
    - Observe where they get stuck and what they like
  - Modify your design based on testing

- What tasks could we ask a user to perform to test the dining hall app design?

# Prototyping and Testing: Medium & High Fidelity

- **Medium-fidelity prototype**
  - Typically made once design is more solidified
  - Usually does not involve coding
  - Could be made with:
    - Powerpoint
    - Canva
    - Figma
    - Etc..
  - Same idea -- build, test, adapt
- **High-fidelity prototype**
  - Once you have a solid idea of the design, code your app
  - Test and adapt
    - You might implement one feature at a time

# Your turn!
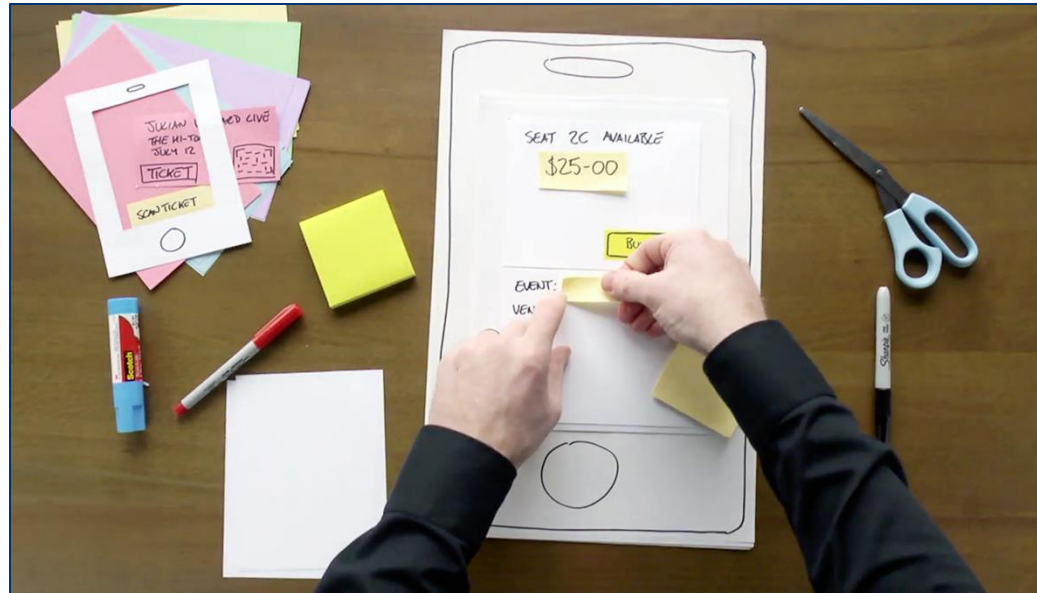


- Work with the group you developed a persona with
- Sketch out the visual analytic system you'd make based on your persona
- Build a paper prototype of the system
- Choose 1 task to have another group perform with your prototype
  - Pay attention to where they run into trouble

# 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**…