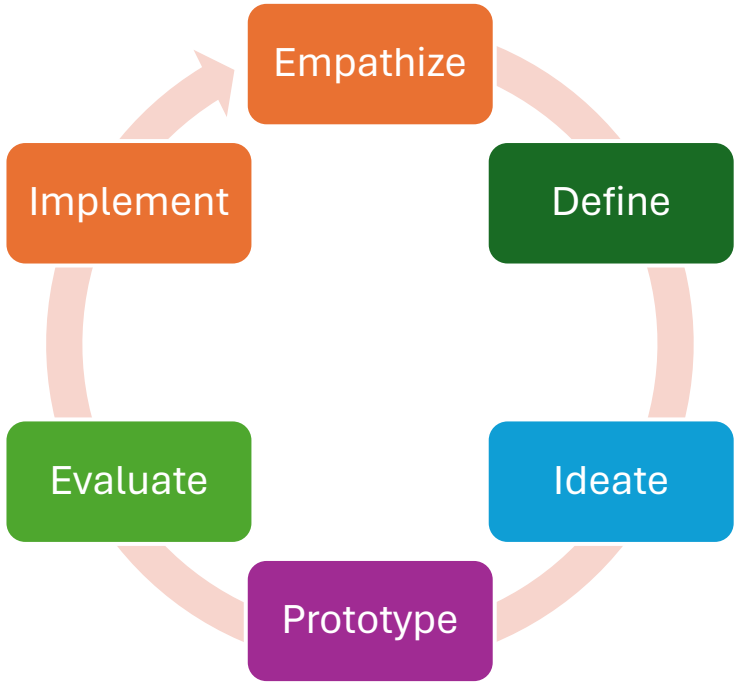
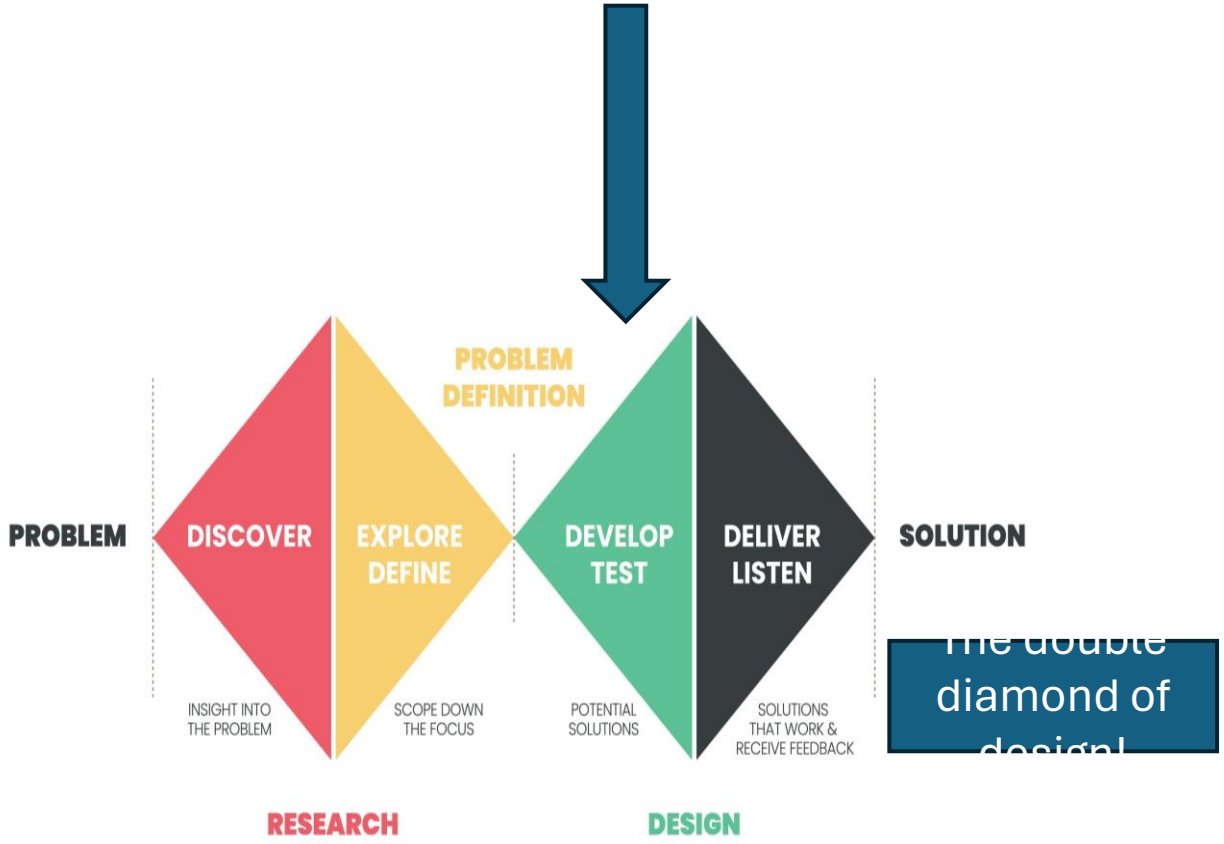
Week 11: Prototyping

CS798H: Sem-II 2023-24

Prototyping





No matter which design process we choose, prototyping shows up!
It is central to the “design” of anything (products, UI/UX, buildings, etc.)

What is a prototype?

- “Model” of a system (or its most salient aspects relevant to the problem at hand)
- Intended mostly to get feedback (or throw if not good!), so it needs to be quick and dirty

- You prototype to..
  - See if a solution idea is feasible
  - Get early feedback on a solution idea
  - Compare multiple potential solutions
  - Think through details

Characteristics of a prototype

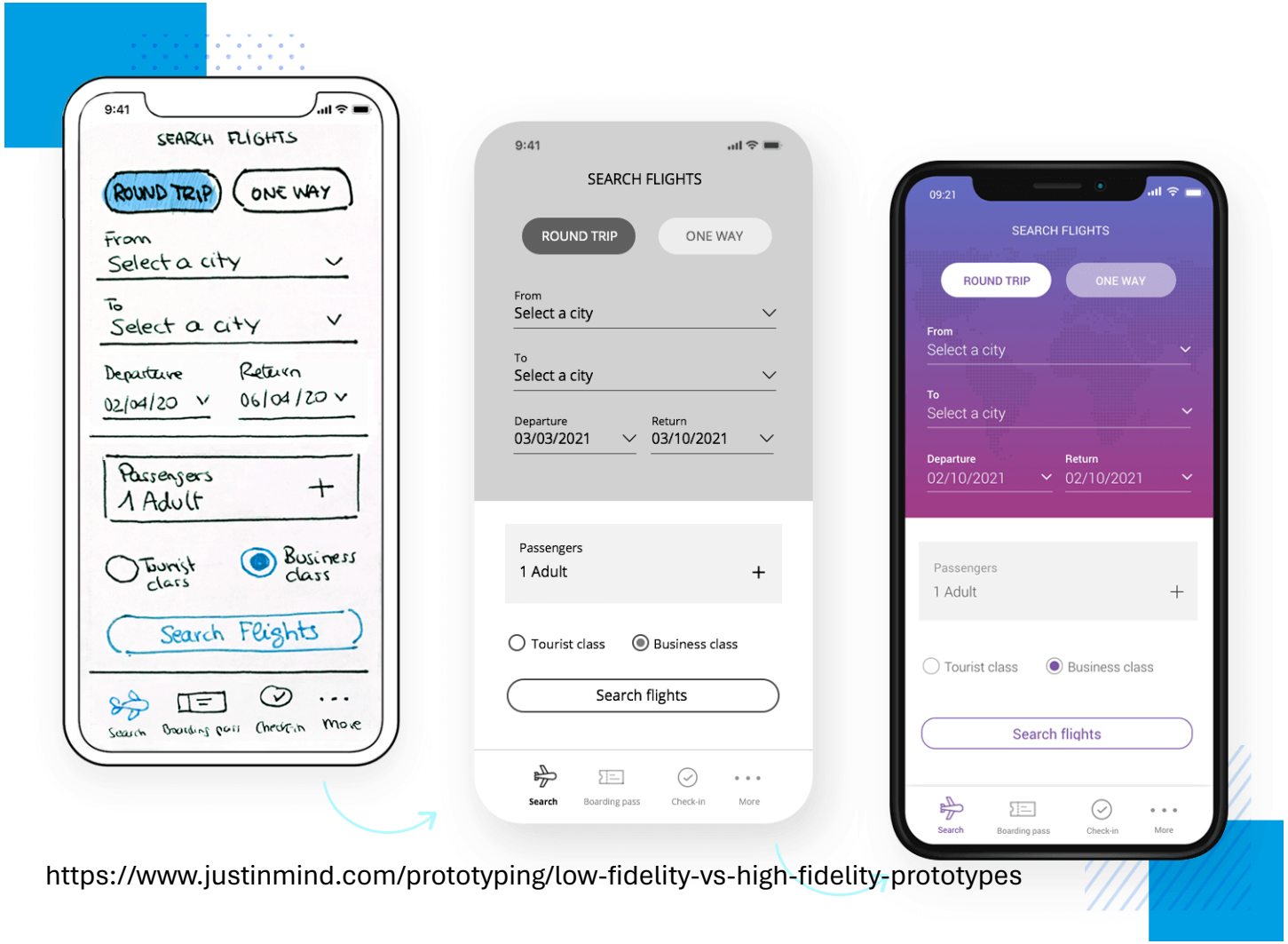
- Almost always incomplete (otherwise, it is a full-fledged solution implementation and not a prototype)
- Cheap, quick and dirty
  - Quick, so it is easy to try out a lot of ideas to move fast
  - Cheap, so you don’t feel too stuck to an idea and are okay to throw away bad ones and try a lot of potential ideas
  - Dirty, again essential to keep it quick and cheap, but also to focus on essential details and leave the rest “dirty” (incomplete, put place holders like “lorem ipsum”, etc.)

Types of prototypes in UI/UX

Fidelity = exactness; completeness; amount of fine or precise details

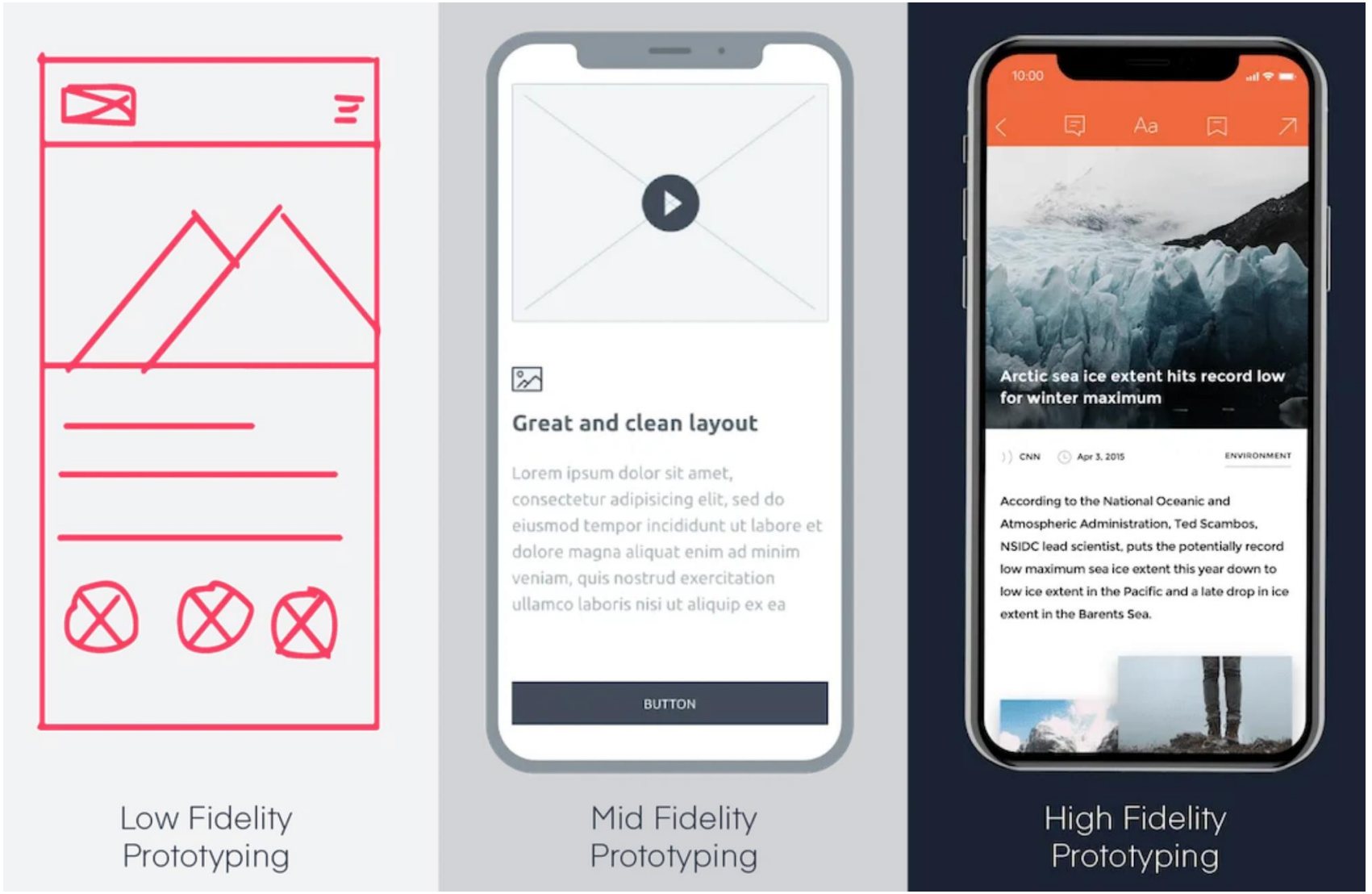
Low-fidelity prototype	Medium-fidelity prototype	High-fidelity prototype
Incomplete, less detailed	Has some essential details and behaviors; not all	Has all details, including look and feel (some behaviors
Content: mostly blocks of areas Behavior: Crude, manual transitions, or just arrows Colors, look and feel crude	Content: Some text, including some realistic details Behavior: Some animations, but stubbed, not all. Colors, look and feel OK	Content: Realistic Behavior: Mostly yes, but stubbed (clicking form submit says “submitted”, but not actually so) Colors, look & feel close to actual
Examples: Paper prototypes, wireframes/mockups, wizard-of-oz, “jugaad” models	Examples: Powerpoint, colored sketches,	Example: Figma, HTML/CSS/JS, Chatbot with nice interface but answered by humans
Advantage: quick, dirty	Compromise between hi-fi and low-fi prototype (e.g., some colors where expected, but rest all just plain black)	Advantage: Realistic for users, easy to reuse parts for real system
Disadvantage: User often stuck on appearance details		Disadvantage: Takes time

Examples (from internet)



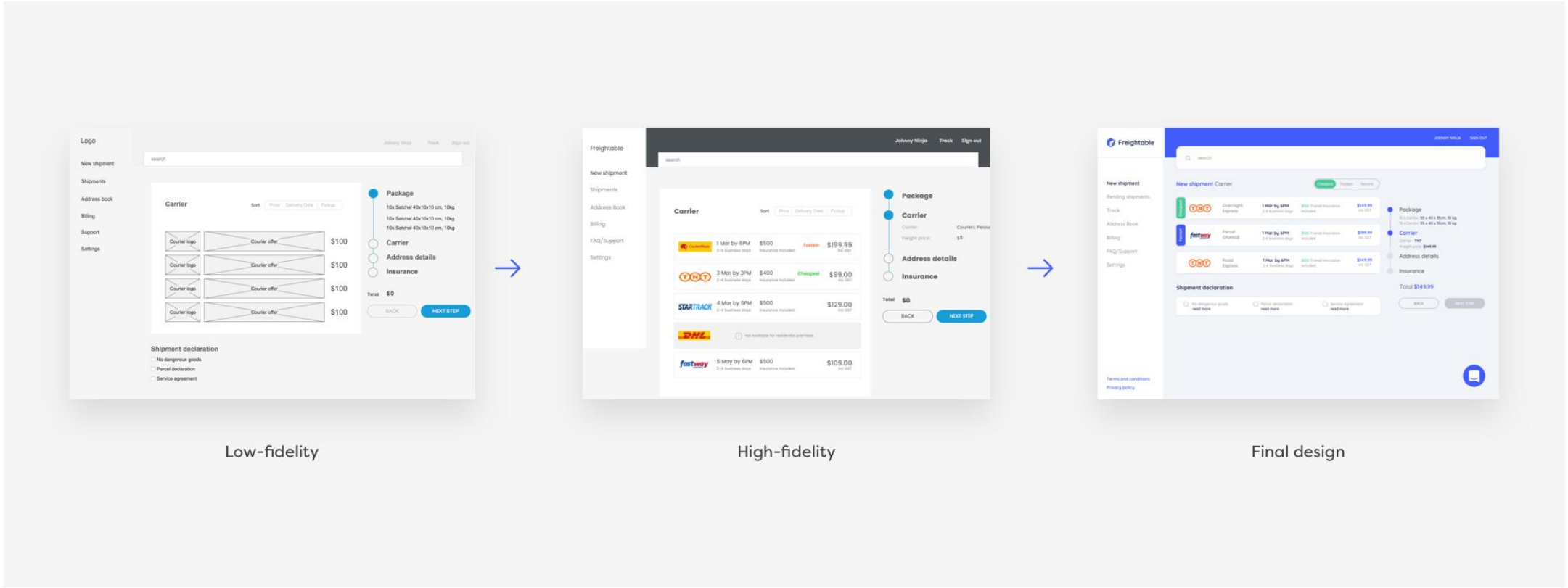
https://www.justinmind.com/prototyping/low-fidelity-vs-high-fidelity-prototypes

Examples (from internet)



https://aloa.co/blog/what-is-a-prototype-popular-tools-in-2021

Examples (from internet)



https://medium.com/7ninjas/low-fidelity-vs-high-fidelity-prototypes-903a7befaa5a

Paper prototypes

- Interactive paper prototypes
  - https://www.youtube.com/watch?v=y20E3qBmHpg
  - https://www.youtube.com/watch?v=JMjozqJS44M
  - https://www.youtube.com/watch?v=85muhAaySps

- Evaluating paper prototypes
  - https://www.youtube.com/watch?v=OlbdIXLunt4

  See Chapter 3 on this video for how to make various controls in paper prototypes

Other tools / options

- Powerpoint with animations for transitions / interactivity
- Moqups, Visio and many other wireframing tools
- Figma
- Wizard of oz (human in place of AI)
- Human in place of voice-based responses/commands

You can also get very creative with them!

- https://www.youtube.com/watch?v=d5_h1VuwD6g
- https://www.youtube.com/watch?v=-SOeMA3DUEs
- https://www.youtube.com/watch?v=IwL3yXdupv0&t=3120s
- https://www.youtube.com/watch?v=KhwifJtBxTk

When prototyping…

- Keep it simple, stupid and cheap
- Helps you throw away what is bad, and you don’t get too attached to ideas

Evaluating prototypes

- Evaluate a prototype on its own – fidelity depends on nature of evaluation
  - For usefulness
  - For usability

- Evaluate a prototype against another
  - Prototype A vs Prototype B
    - Both need to be same fidelity
  - Old system vs. New prototype
    - Old system is high fidelity, so new one needs something similar
    - Alternatively, compile a bare bones version of old system

Two kinds of evaluations

- Evaluation with users
  - Controlled experiments (compare A and B, by controlling everything else such as tasks, machine, ever user characteristics)
  - Usability tests (users do a task and we see how well they do it)
  - User surveys (show two elements such as icons and see what people like)
- Evaluation without users
  - Within development team: Cognitive walkthrough where you pretend to be a user (or think like a user of interest)
  - With experts: who evaluate your interfaces against a set of heuristics

Controlled experiments

- Study the effect of one variable (e.g., interface) on another (e.g., time)
  - E.g., Does the presence of search history improve search time? If so, by how much?

- General setup:
  - Get participants, make them do the same/different task with and without search history, and then compare search times.
  - Looks simple; doing this right is HARD!

Experiment design: Identifying variables

1. Independent variables (IV) : What researchers change (e.g., interface with vs. without search results) and is independent of other experiment variables

2. Dependent variables (DV) : Depend on independent variables (e.g., search times, no. of search queries, etc.; they might change based on the independent variable (yes/no search history))

   Independent variable → Dependent variable

3. Confound variables (CV): Anything else (other than independent) that might alter the dependent variable in the experiment; ideally, we should not allow confounds to alter results. Atleast, we must reduce their effect on results.

Ensure variables are concretely measurable (e.g., time taken for task completion, and not “productivity”!)

Some examples

- Gas laws in Physics:
  - At Constant Pressure, Volume proportional to Temperature ($V \propto T$)
  - In experiments, we change temperature, and measure volume each time
  - Temperature → independent (experimenter changes)
  - Volume → dependent (on change in temperature)
  - Volume can also depend on pressure, but we don't want it to mess up readings (we are only interested in volume and temperature) → Confound
  - We therefore keep pressure constant (so there is no extra changing effect of pressure across readings)

Some examples

- Using slides in class reduces attention and lowers grades
- Experiment: Half lectures with slides and half without; hand out survey after each lecture on how interesting, did you make notes, were you surprised, did you fall asleep, quiz questions with scores, etc.
  - IV = with/without slides (0 or 1 categorical variable)
  - DV = survey results (quiz scores, interest scores, etc.)
  - Confound? (Due to social desirability bias, power relation between teacher and student, each lecture has different content, different students show up to class each time, etc.)

Experiment design: Study location

- In-vitro=in the lab
  - In the lab; unrealistic, but offers better control (e.g., no distractions)
- In- vivo = in the field
  - Realistic conditions, so higher external validity
  - But, can introduce confounds (e.g., interruptions mess up time measurements, as well as focus and attention)

Experiment design: Participants

- Recruit from user population
- Hard to decide whether to recruit diverse / narrow
- Diverse means controlling for expertise, backgrounds, etc = they might introduce confounds, so we need to ensure the same kinds of people use both systems/prototypes

- How many?
  - Atleast 30 comparisons between the two systems being compared!
  - 30, because many statistical tests operate by comparing distributions of data, and distributions plot smooth at about that size

Study design: Task assignment

- Within-subject
  - Take 2 comparable tasks; Get N participants
  - Each participant does two tasks: one with System1 and one with System2
  - Balance: N/2 participants do Task 1 first, and N/2 do task 2 first.
    N/2 participants do Task1 with System1, and N/2 do Task1 with system2.
  - Compare the difference in DV for both System1 and System2.
  - Question: Why do we need this balancing?
- Between-subject
  - Take one task(s); Get N participants
  - All N participants do same task(s), N/2 with System1 & N/2 with System2
  - Compare average/median/SD between the two groups
  - Confound: Different participants have different skills, motivations, backgrounds, etc.
  - Question: how to deal with confound?

Conclusions from experiment design

- Measurements subject to rigorous statistical tests
- Tests aimed at rejecting a true/false hypothesis
  - There is no difference in mean search times between prototype A and B
  - Tests provide a “p-value” which is basically probability whether any difference observed is “by chance” or is real difference (lower p = lower chance of “by chance”, and higher significant differences). Typically, $p<0.05$.
- They work by comparing distributions of data (e.g., frequency distribution of task time in A vs. B).
- Different tests also make some assumptions about data (sample size, normality of distribution, equal variances, etc.)

- Don’t fret much about specific tests, but this is the general idea!

Study design

- Read Lazar, Chapter 2 carefully.

Usability tests

- Also done with users (N=5 or more)
- What we do?
  - Give the prototype to user
  - Give a task
  - Observe user do the task (with optional think aloud)
  - Optionally, provide a survey on what characteristics are good about the interface, how they rate it, etc.
    - NASA's Task Load Index, System Usability Survey (SUS), Microsoft desirability toolkit, etc.
- Outcome is a list of usability issues
- Also helps see usefulness issues if you see when users are confused, or if you ask if this is what they'd normally do

Cognitive walkthroughs

- Done within the team, when access to users is hard
- Even otherwise, do this as a first cut evaluation
- Pick a prototype and task; list the task steps
- Create a user “persona” (and list down their key characteristics)
  - Often, you need more than persona, then use ones at extremes
- For each step along the task, answer the following 4 questions
  - Will “User” want to do this? [Ideally, use persona name instead of “User”]
  - Assume “User” wants to do this, will s/he know what to do?
  - Assume “User” knows what to do, will s/he actually do it?
  - Assume “User” did it, will s/he know they did the right thing?
- Write down yes/no/maybe, along with reasons. Every no/maybe, is a usability issue to be fixed. The reasons often provide hints for what the fix is.
- Seems tedious, but can be done in an afternoon for atleast most common/least common paths as needed.

A note on personas

- Good personas are data driven, and come from user research (in the empathize phase)
- Example for how to do it, if you care:
  - https://uxpressia.com/blog/how-to-create-persona-guide-examples

- There are a lot of personas out there for use (people with disability, specific problem solving aspects, etc.)

- The definitive guide on the topic is:
  - “The persona lifecycle” by Tamara Adlin and John Pruitt.

Heuristic evaluation

- Heuristics = rules of thumb (for how to build interfaces)
- We evaluate interfaces against heuristics and look for violations
- Result is a list of violations to be fixed
- Who does it?
  - Ideally, someone that can interpret UI/UX heuristics, and catch violations (so a UI/UX expert/professional)
  - How many? Ideally, 5 or more.

- Provide interface (or screens)
- Provide a set of heuristics (or ask experts to pick their favorite)
- Evaluate interface against heuristics, and get a list of violations and possible fixes from the expert
- Challenging, but works great in large organizations with lots of designers, UX folks
- Also great for designers to evaluate their own designs systematically, as a first step

Example heuristics

- https://www.nngroup.com/articles/ten-usability-heuristics/
- Ben Shneiderman's eight golden rules of interface design
- Microsoft's guidelines for human-AI interaction design
- UI tenets and traps
- Rules from Steve Krug's "Don't make me think" for web usability
- There's a lot more, go look for them!