APRESENTAÇÃO
DE APOIO

DESIGN DE UI

Joe Natoli – Aula 01

PUCRS online uol edtech

Professores

JOE NATOLI

Professor Convidado

Tem orientado e praticando a experiência de usuário (UX) para organizações da “Fortune 100, 500” e outras instituições governamentais por quase três décadas. Esse trabalho inclui líderes da indústria comercial, como “Google Ventures”, “Kroll / Duff + Phelps”, “Broadridge”, “Conde Nast”, “Johns Hopkins”, “Mettler-Toledo”, “PHH Arval”, “SC Johnson” e “Wolters Kluwer”, bem como agências governamentais como a “National Science Foundation”, “National Institutes da Saúde” e do “Departamento de Segurança Interna”. Tem falado publicamente sobre os tópicos de Experiência do Usuário (UX) e Design em todo o mundo por 24 anos, em conferências globais como HOW Design Live, SDD, DevWeek e IxDA South America. Seu livro mais recente sobre Estratégia de UX e Melhoria de Produto, Think First, foi publicado mundialmente em 5 de outubro de 2015. Escreveu vários artigos sobre os papeis que o design e a experiência do usuário desempenham no sucesso dos negócios para publicações, como “Fast Company”, “SmartCEO”, “The AIGA Journal”, “The Baltimore Business Journal” e “Business Monthly”.

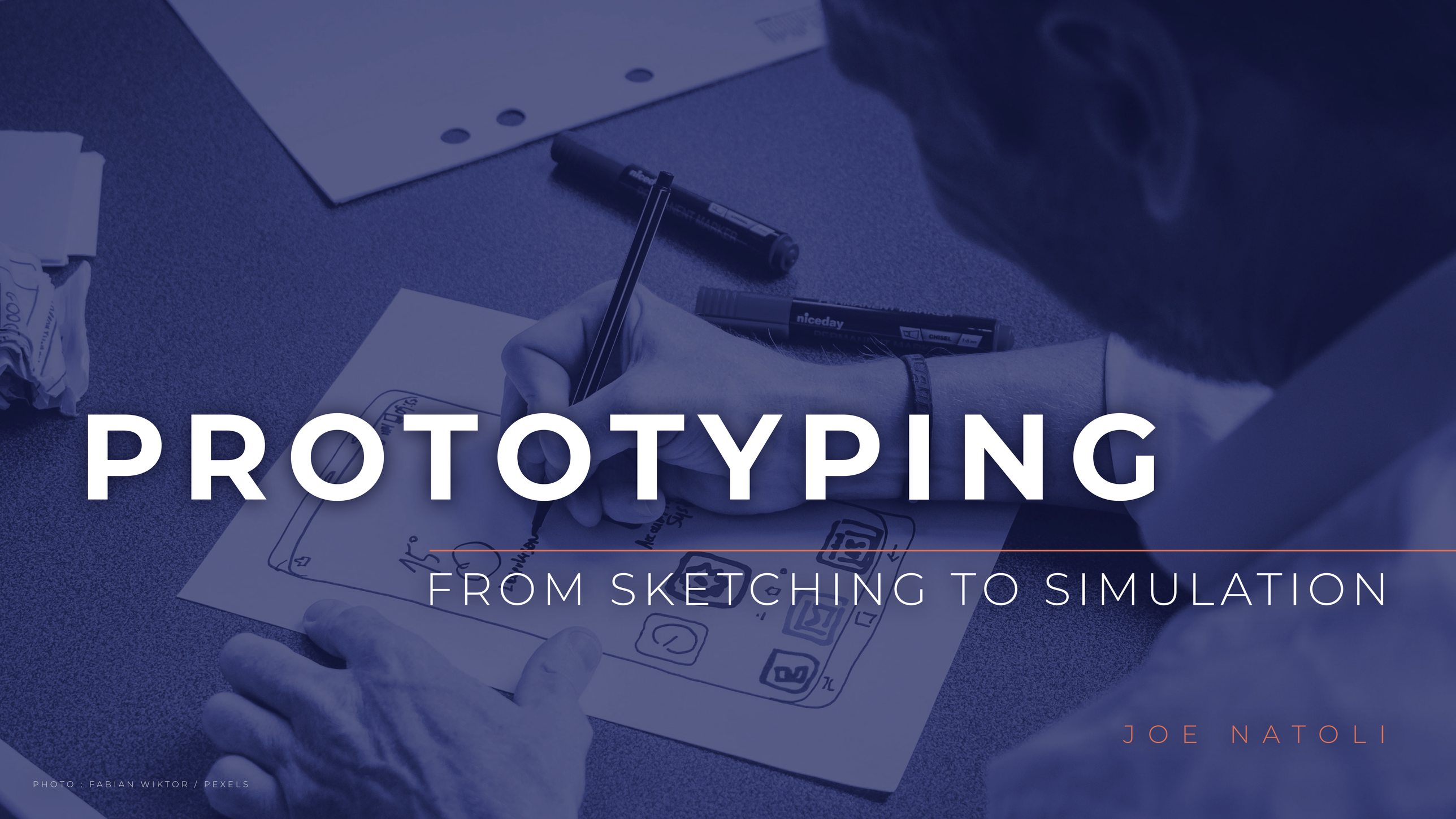
LUANA MÜLLER

Professora PUCRS

Bacharel em Sistemas de Informação - ESUCRI. Mestre em Ciência da Computação (ênfase em Sistemas de Recomendação)- PUCRS. Doutora em Ciência da Computação (ênfase em HCC) - PUCRS. Coordenadora de Engenharia de Software – Pravaler.Pesquisadora colaboradora SERG (Semiotic Engineering Research Group) e DaVInt (Data Visualization and Interaction Lab)

Ementa da disciplina

Caracterização de Interface com Usuário (UI). Introdução ao Design Visual e arquitetura de Informações. Compreensão dos princípios de redação para UX. Definição de prototipação e seus principais objetivos. Apresentação de ferramentas de apoio à prototipação e ao design de interfaces.



PROTOTYPING

FROM SKETCHING TO SIMULATION

JOE NATOLI

PHOTO : FABIAN WIKTOR / PEXELS

what exactly *is* a **prototype**?

a prototype is a **simulation** of a finished product.

- represents how **interaction** between the user + the interface would work.
- serves as an imitation of how the real-world **processes** underneath the UI would work.
- can represent a **single interaction** or an **entire app experience** from login to logout.

why do we prototype? to **test**.

USABILITY

will this work for users in a way they can easily understand and act upon?

does it surface action and information where, when and how they need it?

FEASIBILITY

is it possible to actually *build* this?

do we have the time, budget, resources, knowledge and access to technology?

VIABILITY

how will we support + improve this?

prototyping ensures we get it *right*.

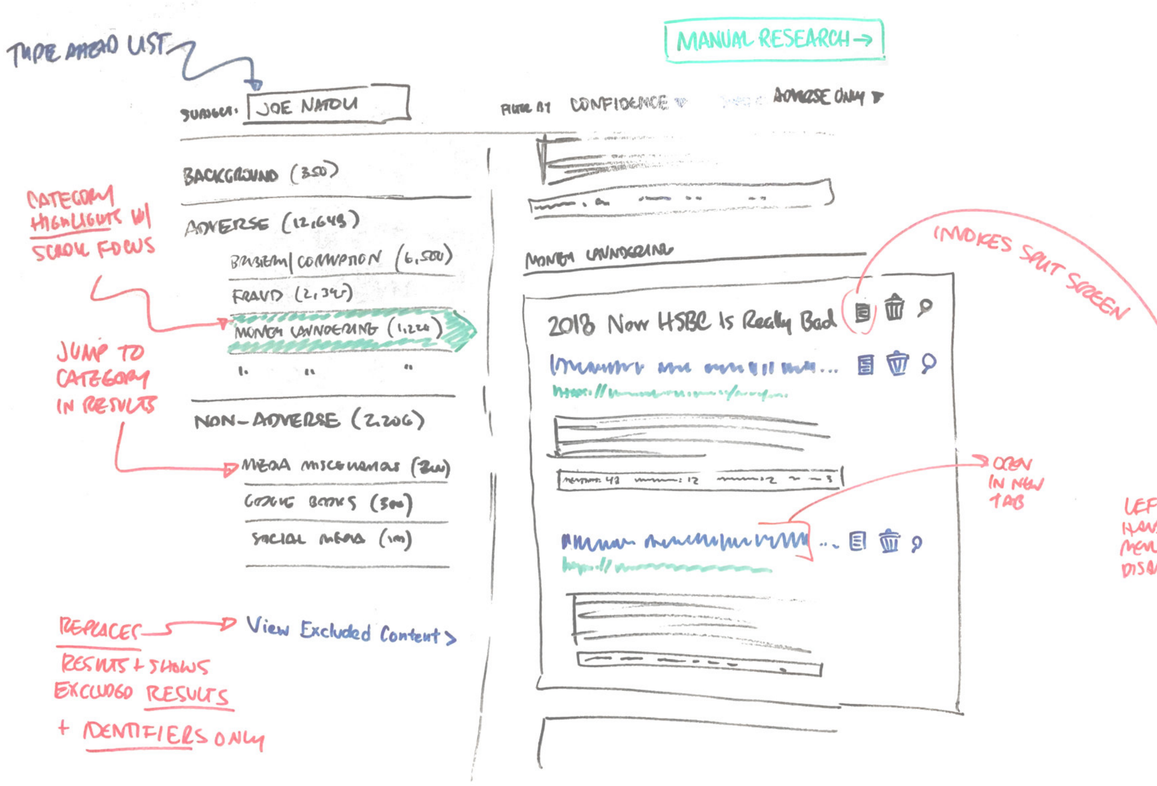
we test ideas with real users, so we can see how they expect + *want* to use the product. prototypes tell us:

- if what they **expect** is what they **get**
- what's easy to use + understand
- where, how + why they get stuck or tripped up or confused or frustrated

testing reveals the parts that need **improvement**.

we're finding + resolving usability issues well *before* launch.

this allows us to adjust the design, test again + keep doing so until we get it right.



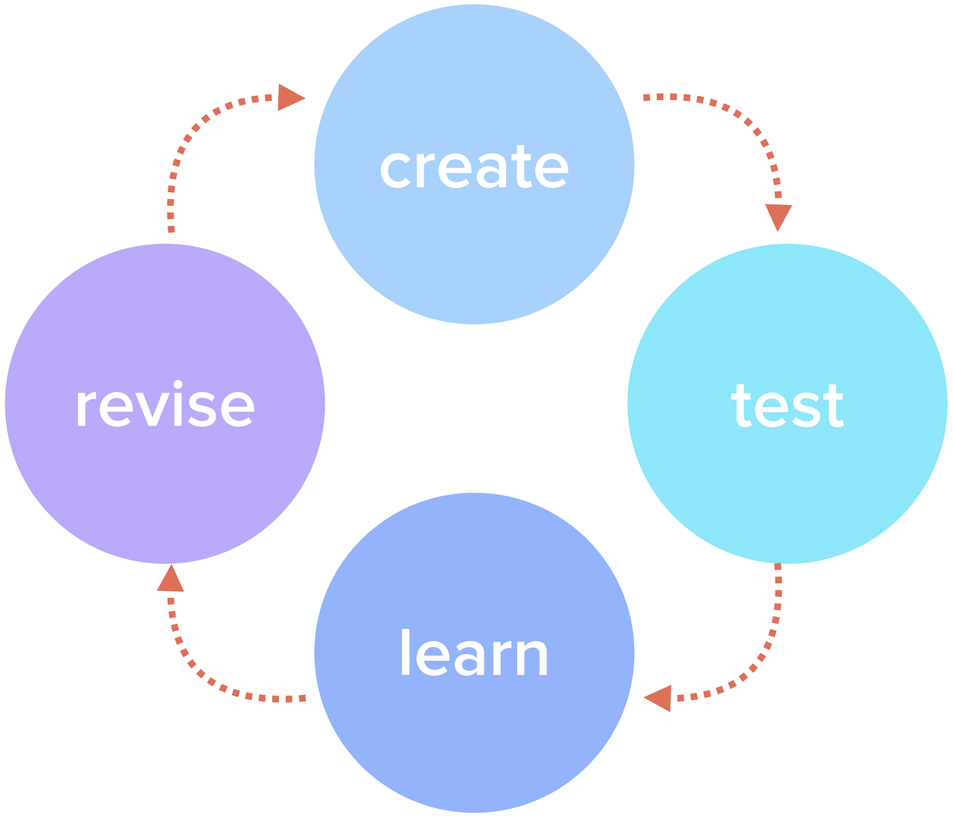
create, test, learn, revise, repeat.

create the solution, exploring content, interaction, UI + UX.

test the prototype with users, to **learn** how well it meets needs + expectations.

revise areas that need to be refined, defined, improved or clarified.

repeat ;-)



start small.

you start with only a few **key screens** or sequences.

as you iterate + test, the prototype grows in **breadth** and **depth**; more screens, content + interaction.

along the way you make **changes to smaller details,** in real-time + across iteration cycles of days or weeks.

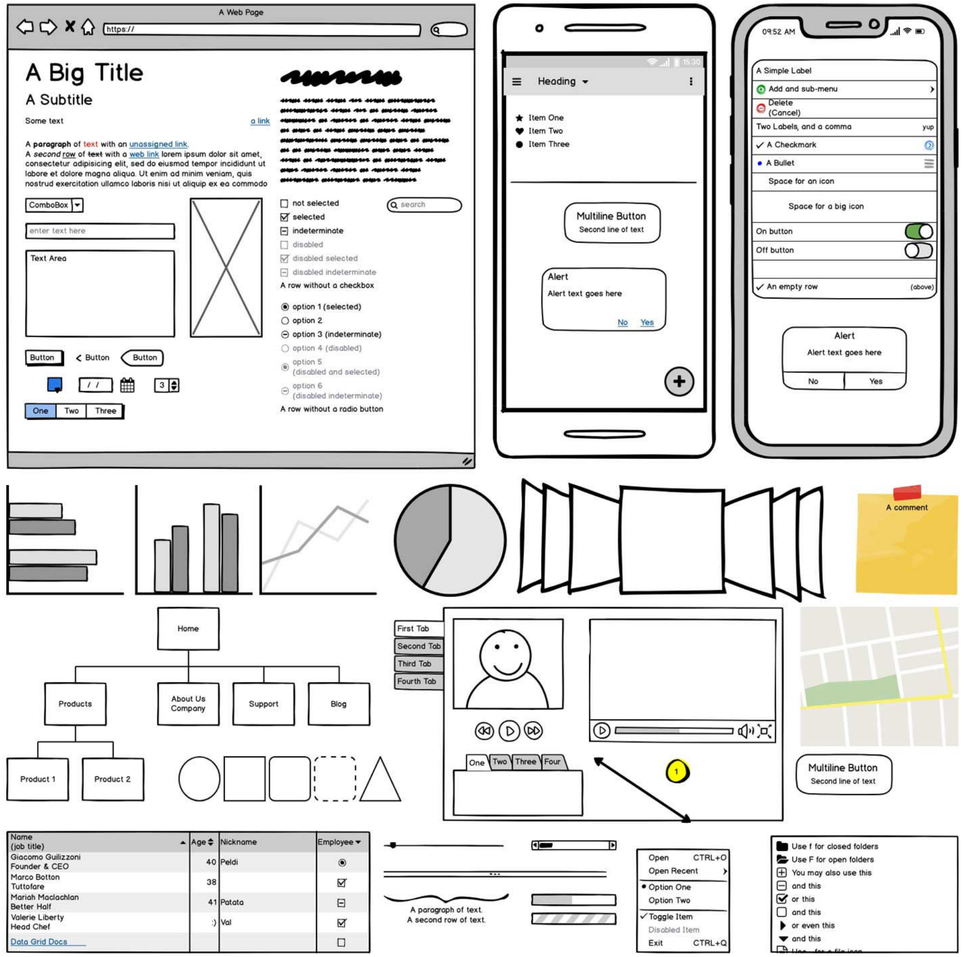


IMAGE SOURCE: Balsamiq; *balsamiq.com*

how much is enough?

usually it's **20% of the functionality** that will be used 80% of the time.

this is the stuff you have to get right in order for people to **recognize** that the product meets their needs.

it's the content + interactions + visual presentation that ensure they'll be **able + willing** to use it.

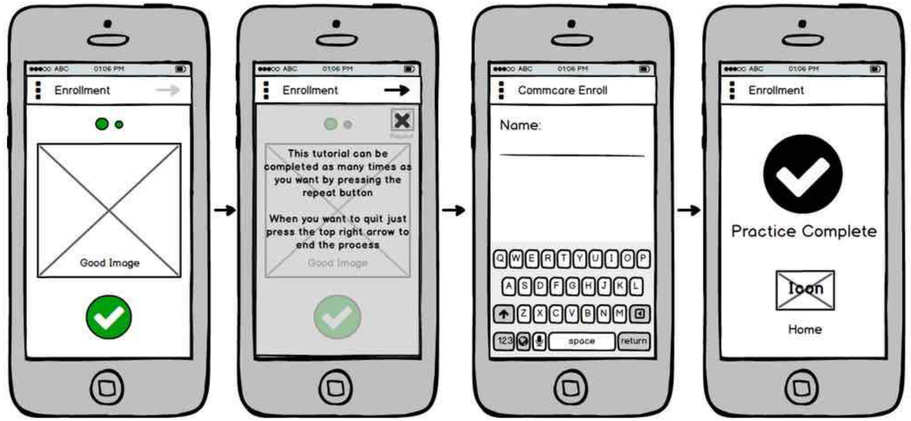


IMAGE SOURCE: Balsamiq; *balsamiq.com*

sequences, not screens.

you *never* prototype a single screen.

you prototype a sequence, a **journey**, from start to finish.

e.g. *"Joe logs in to the banking app, checks his balance and transfers money from checking to savings."*

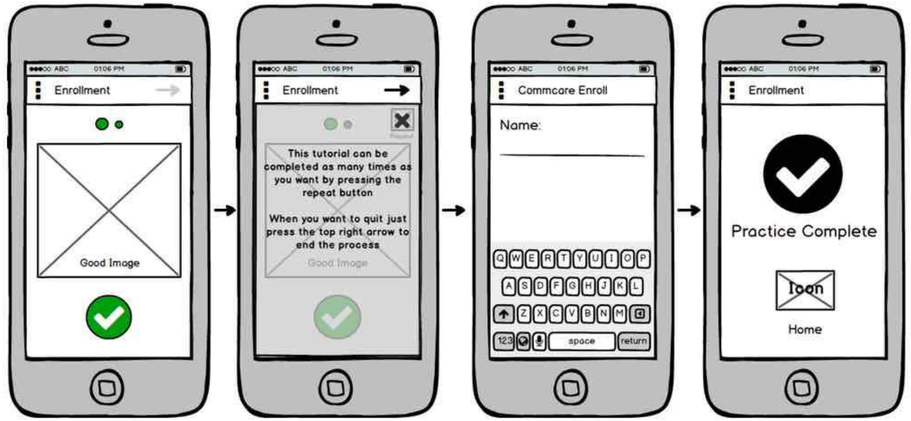
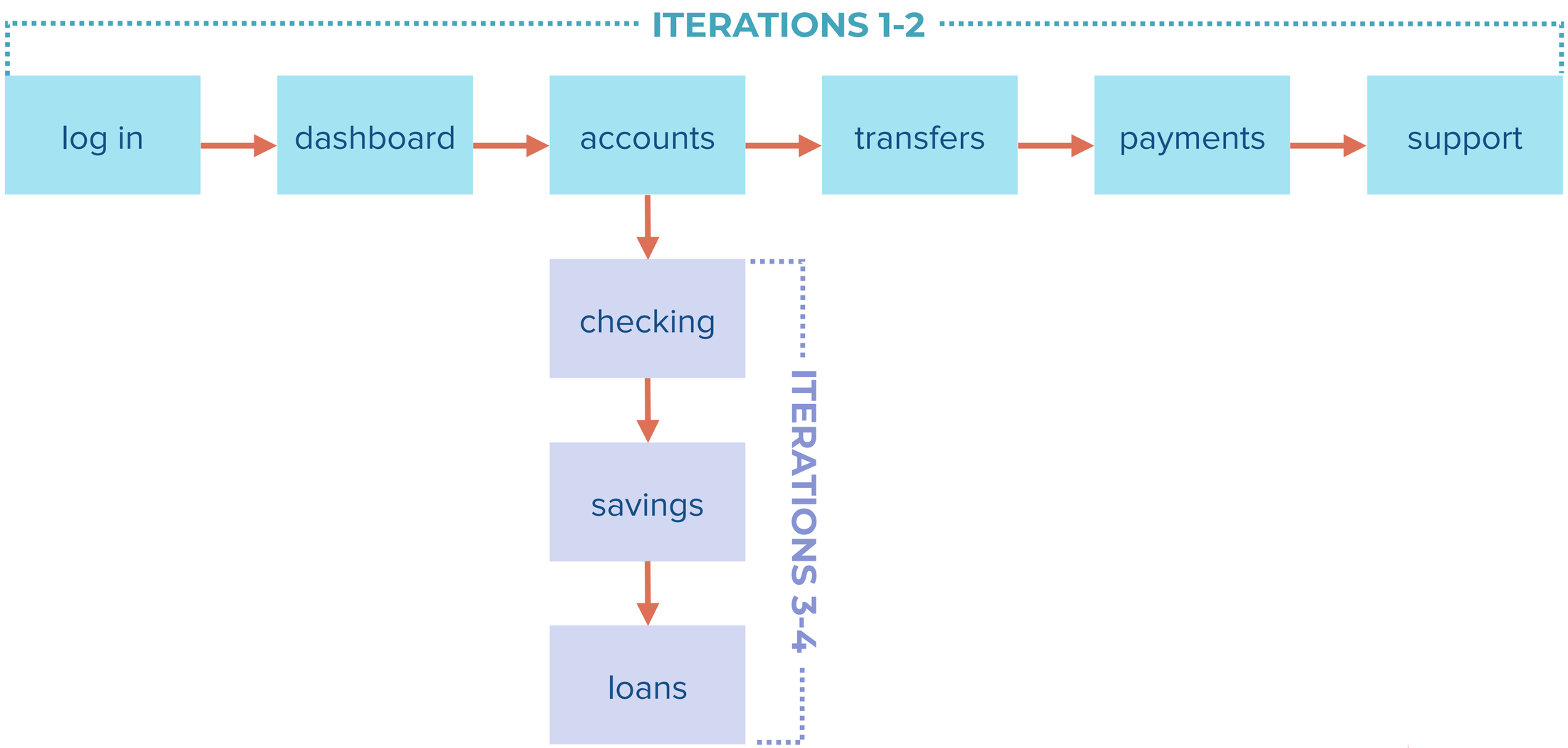


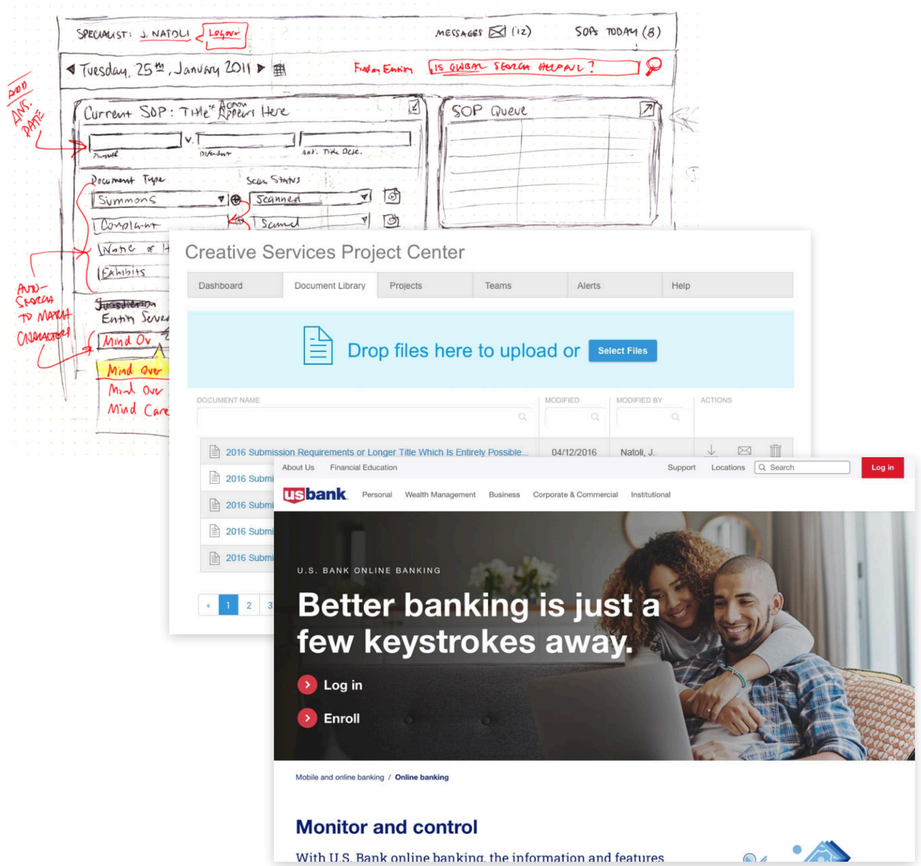
IMAGE SOURCE: Balsamiq; *balsamiq.com*

go wide first, *then* deep.



wireframes, prototypes + mockups (oh my!)

- unlike sketches, wireframes or mockups, a prototype is an **interactive simulation**.
- sketches, wireframes and mockups are essentially **static** components.
- if you can't tap, click or swipe through it and have it respond to those actions + gestures, **it's not a prototype**.



protoyping software

Adobe XD, UXPin, Figma, Sketch + others will export static screens to a **clickable, interactive prototype**.

you create low-fidelity wireframes + *export* them as a prototype.

there are many — so how do you choose the best tool to use?

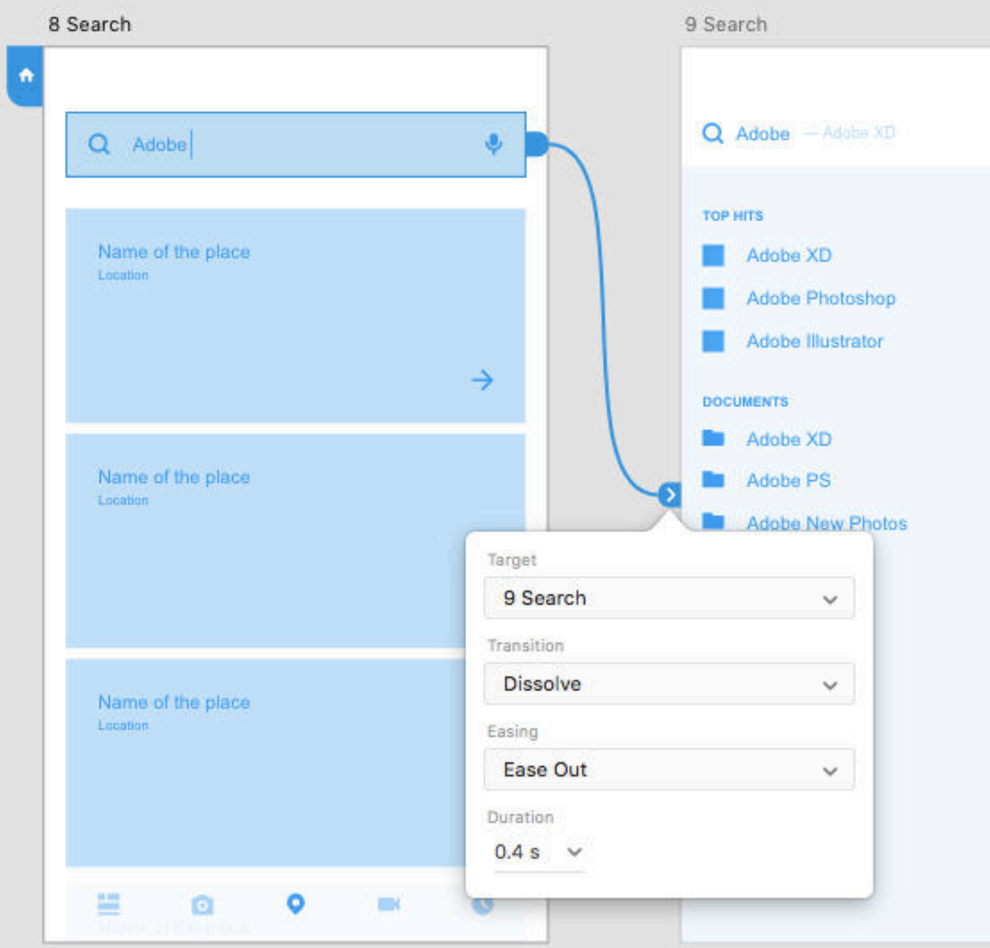


IMAGE SOURCE: Adobe XD; *adobe.com/products/xd.html*

choose a tool that...

is **easy** for you to learn and use.

is **flexible**, meaning it can be used to create prototypes for web, desktop + mobile apps.

has a **large library** of stencils, templates + widgets, either built in or available from third parties.

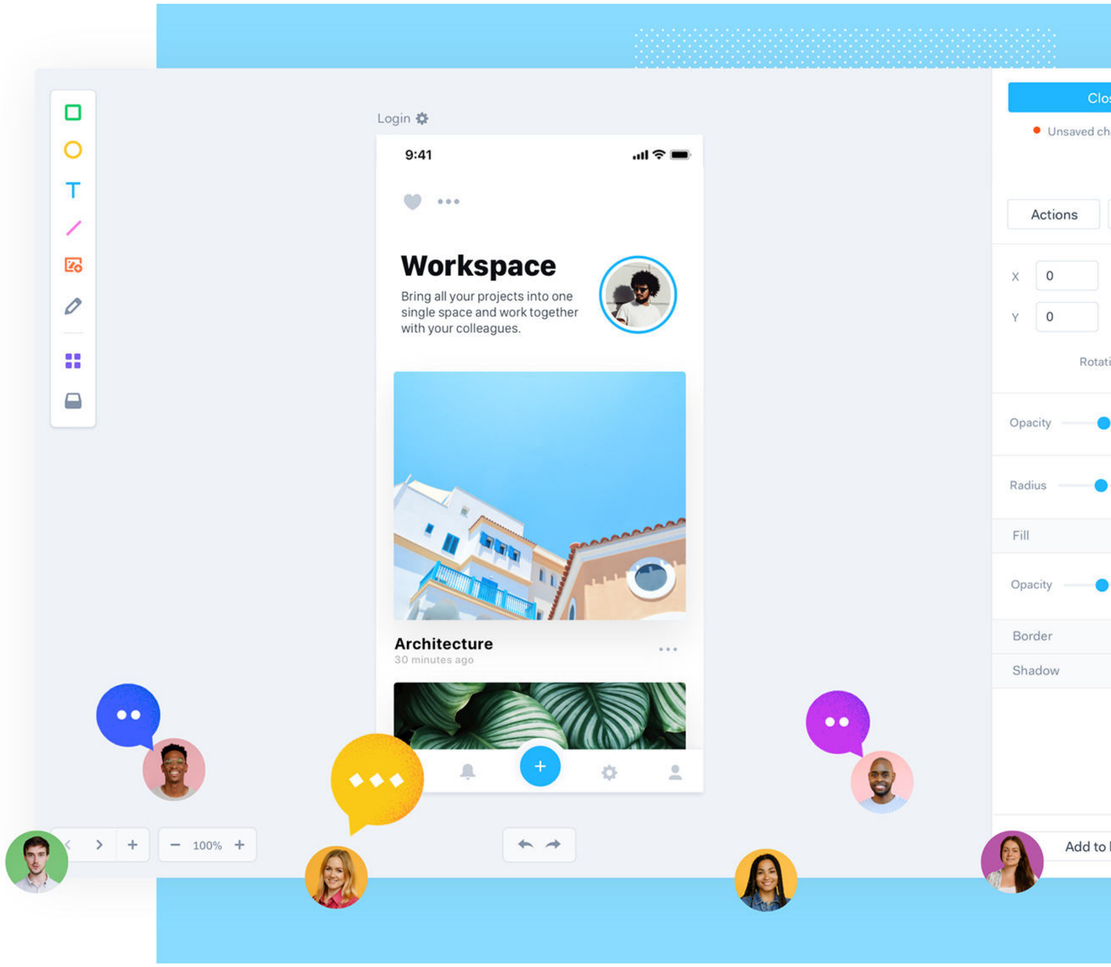


IMAGE: Marvel App; *marvelapp.com*

choose a tool that...

makes it easy to **share** the prototype with others for testing + review.

can **capture** feedback during use.

allows **changes on the fly**, while getting feedback in a session.

allows **collaboration**, e.g. multiple people working on the prototype at the same time.

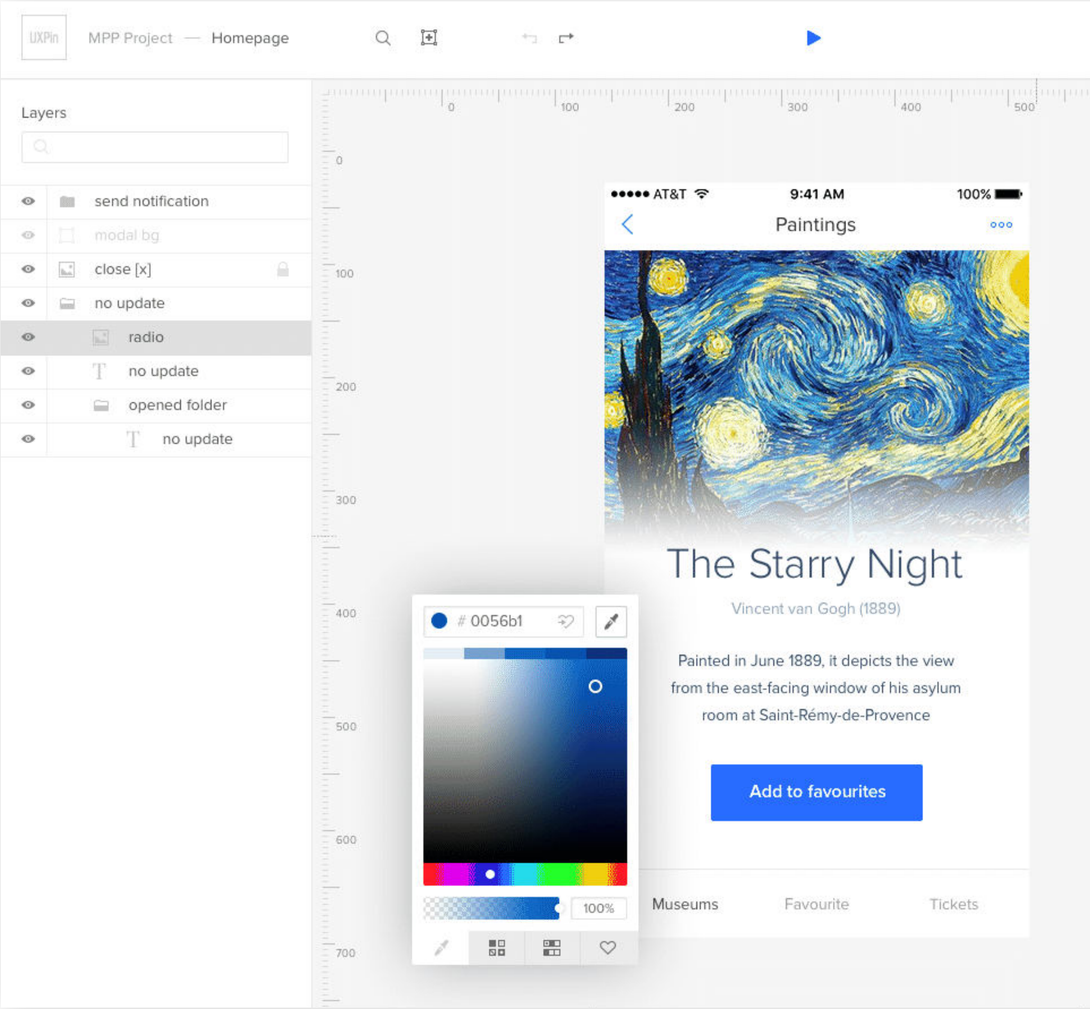


IMAGE: UXPin; *uxpin.com*

the **problems** prototypes solve

how's this going to **work**?

what's it going to **look** like?

there are two choices at the outset of a project.

1

start building with production-ready code, launch it and see how people react.

2

build + test low-fidelity prototypes early + often, to figure out what's actually *worth building*.

guess which approach is usually chosen?

prototyping makes a lot more sense + creates a lot less waste in terms of unnecessary work / rework.

however, far too many organizations take the first approach + **dive right into coding**.

the result?

IMAGE: Pexels, pexels.com

teams are grossly **overworked + stressed out**.

they're spending 50% or more of their time **reworking things** that either aren't functioning properly or aren't delivering the intended user or business results.

backlogs are growing exponentially + management isn't getting the **results** they hoped for (usually money made or money saved).

IMAGE: Pexels, pexels.com

new requirements + requests keep showing up even after code is committed.

emergency meetings are called; changes and revisions and rework are **endless**.

people are working themselves to death long past dark seven days a week, trying to reach a finish line that **keeps moving** every time they get close to it.

IMAGE: Pexels, pexels.com

and if all of that weren't enough, **users are frustrated** — and often angry.

they're taking their business elsewhere.

to a competitor who isn't using them as guinea pigs to test a bunch of new features they *didn't ask for*.

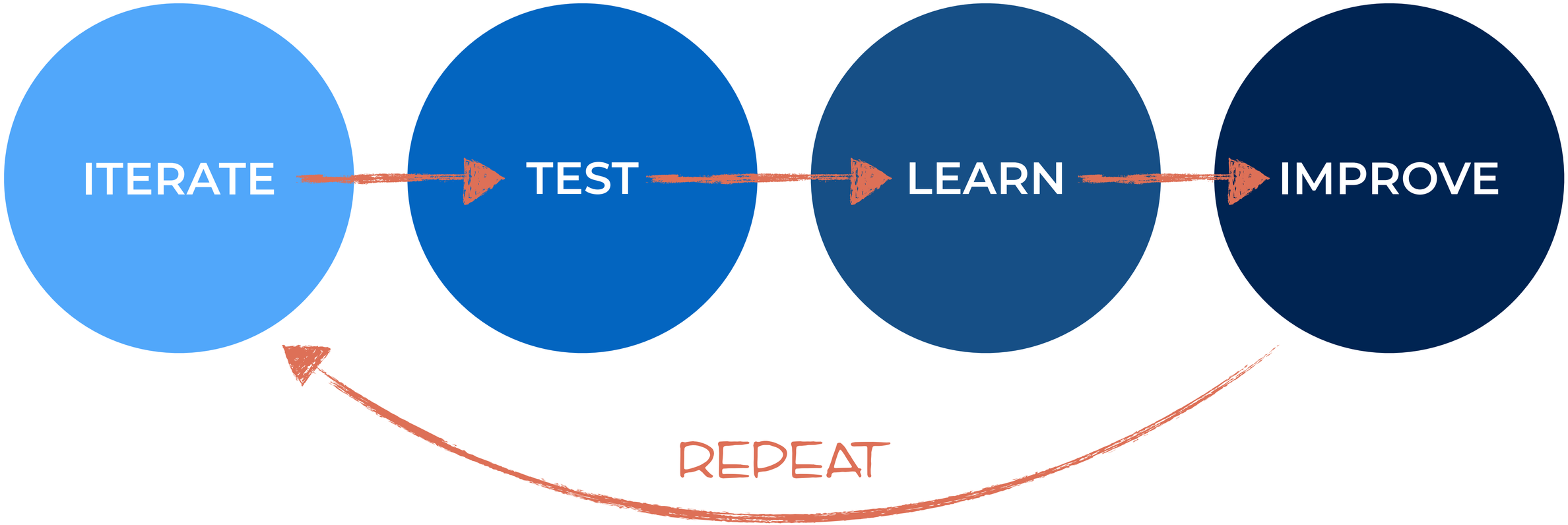
IMAGE: Pexels, pexels.com

I've seen this chaos + overload inside almost *every company* I've ever consulted with.

there's only one antidote.

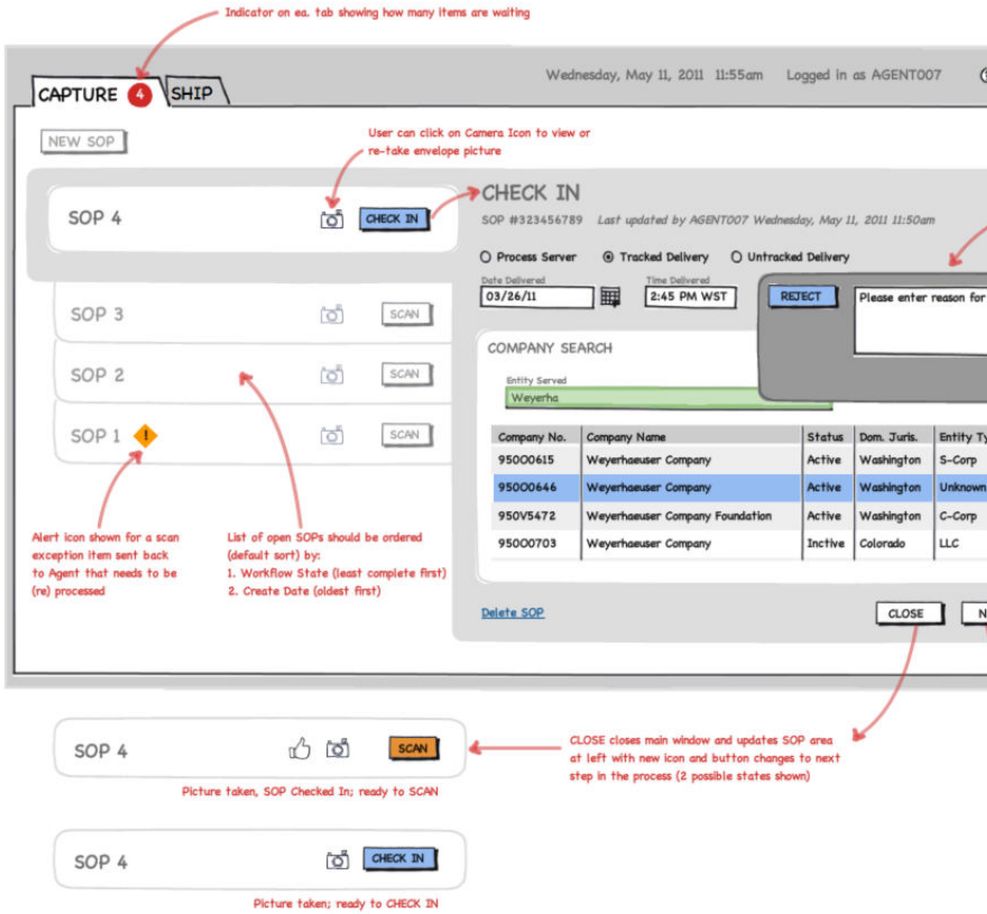
early, rapid, **continuous** prototyping.

early, rapid, **continuous** prototyping



starting the *minute* a project becomes real.

- allows everyone to quickly get a handle on what **features + functionality** could or *should* be.
- how it will actually **work** when someone uses it.
- prototypes communicate the *experience* in UX; allows us to test its **value** and **usefulness**.



1

enables better UX + design decisions.

critical interactive design choices can be finalized in prototyping.

hi-fi prototypes deliver instant feedback in testing, allowing informed, educated decisions.

instead of a *guess* based on “best practice” design / UX principles.

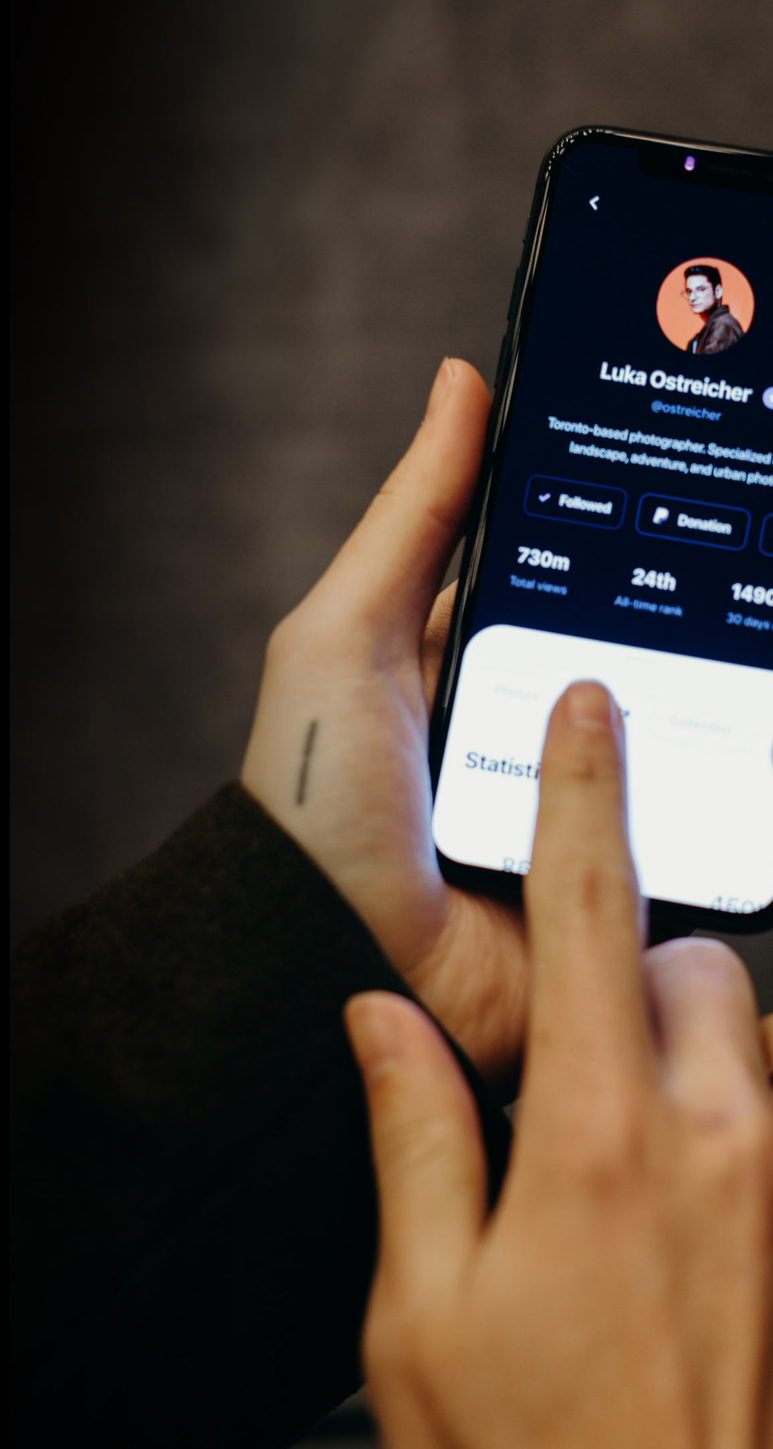


IMAGE: Pexels, pexels.com

2

keeps focus on UX + context of use.

you're using all your senses when working with a prototype.

so when something doesn't work as expected, it's *real* to you.

you experience the same visceral, emotional reaction a user does.

IMAGE: Pexels, pexels.com

2

keeps focus on UX + context of use.

users normally test prototypes in a “real life” environment + context.

external, contextual factors that affect ease of use, like distractions or noise or device limitations — are all *present*.

experiences are truer to life, so your test results are more useful.

IMAGE: Pexels, pexels.com

3

non-sequential + doesn't hold up any other work.

developers are still coding, database folks + software architects are still working out structure + architecture.

"we don't have time for prototyping, everyone else will be waiting for it to be done, or for test results."

if it slows you down, **you're doing it wrong**.

IMAGE: Pexels, pexels.com

why do we **need** prototyping?

because designing a relevant, personalized, optimized experience for **seven billion humans** is impossible.

IMAGE: Pexels, pexels.com

products often fail.

prototyping can both **prevent** and **predict** that failure.

without prototyping, everyone is *guessing* at what will be useful, usable or valuable to people.

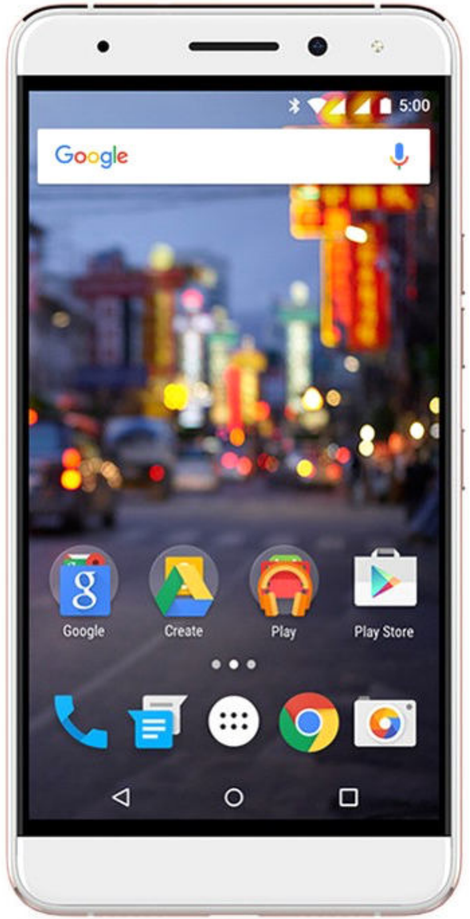
they're following a model of software design that focuses on **implementation** — instead of expectation and understanding.

this is a failure to design for user **mental models**.

we rely on mental models, abstract representations of how things work, to use just about everything.

we use them to try + understand complex behaviors and processes.

they're how we believe something **might** work or **should** work, based on our experiences with other things.



most product teams rush to start **coding**.

and here's what happens as a result.

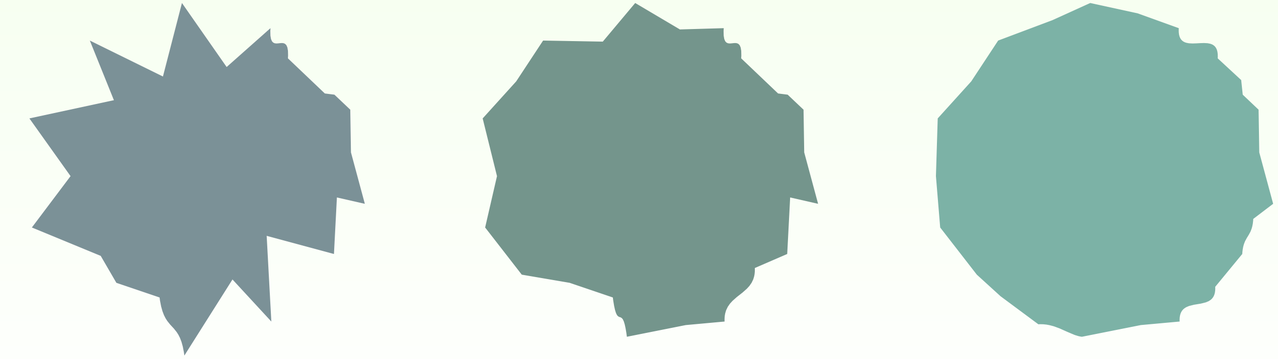
IMPLEMENTATION MODEL

reflects technology

REPRESENTED MODEL

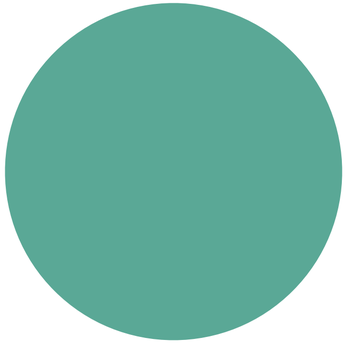
what's designed and built

BAD UX — GOOD UX



MENTAL MODEL

reflects user's understanding

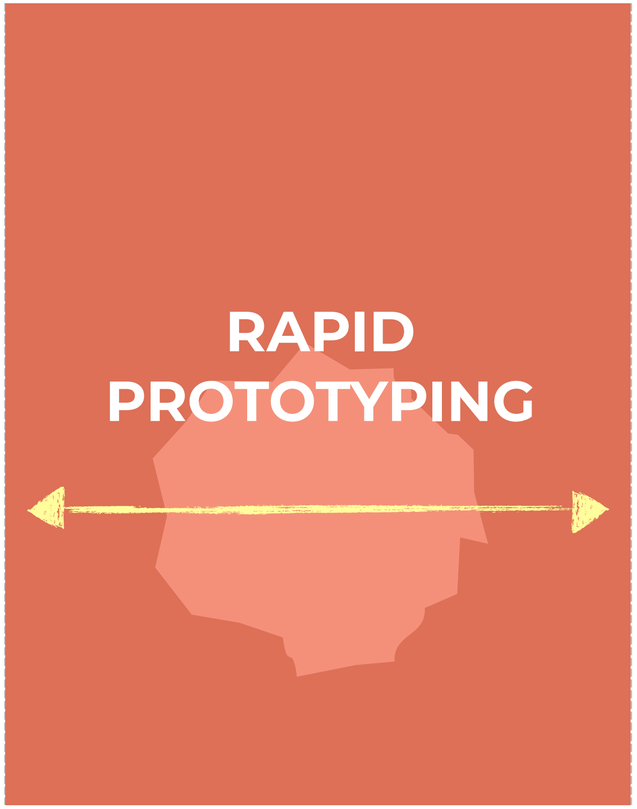


SOURCE: *About Face*, Alan Cooper, 2003

rapid prototyping narrows the gap.

IMPLEMENTATION MODEL

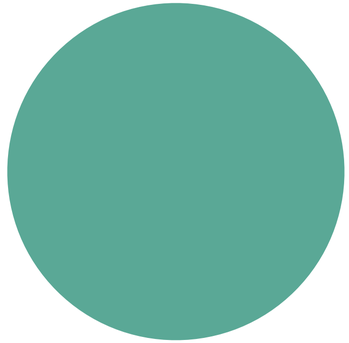
reflects technology



SOURCE: *About Face*, Alan Cooper, 2003

MENTAL MODEL

reflects user's understanding



rapid prototyping narrows the gap.

- we can quickly, cheaply + accurately learn what those models are.
- we understand how best to serve them via functionality, interaction + UI.
- how? by quickly iterating on design, testing, improving design, testing again.
- real-time learning, low design time/effort and minimal delay between cycles.

1

shared understanding across team, stakeholders + clients.

limitations + **possibilities** are evident in context of use; the evidence is in the experience itself.

people see for themselves how intuitive, easy or difficult it is to understand + use.

or if additional guidance or instruction or evidence is missing or necessary.

IMAGE: Pexels, pexels.com

1

shared understanding across team, stakeholders + clients.

conversation + documentation are wide open to misinterpretation + subjectivity.

shared experience is **shared understanding**.

prototypes keep everyone's focus where it belongs: on the **experience**.

(instead of on the visual elements they see.)

IMAGE: Pexels, pexels.com

1

shared understanding across team, stakeholders + clients.

prototypes help to get approval or buy-in from a client, stakeholder, product owner or fellow team member on an idea.

prototyping gets skeptics to **try something** *before* making up their minds it's the wrong approach or won't work.

IMAGE: Pexels, pexels.com

never go to a client meeting without a prototype.

its ability to communicate and persuade is often the difference between a YES or a NO.

DAVID + TOM KELLEY, IDEO

2

prototypes avoid critical mobile UX mistakes.

prototyping is **mandatory** for mobile.

mobile use spans a massive number of use cases, scenarios, technological variables, limitations + constraints.

mobile use environments change frequently and are almost *impossible* to predict.

IMAGE: Pexels, pexels.com

2

prototypes avoid critical mobile UX mistakes.

input methods are radically different from desktop, from limited keyboard functionality to complex gesturing.

screens are smaller + more restrictive, both in size and ergonomically.

people literally use their hands + fingers differently in different situations.

IMAGE: Pexels, pexels.com

2

prototypes avoid critical mobile UX mistakes.

mobile apps cost $15 - $500K to develop.

so mistakes + rework to correct those mistakes is really, *really* expensive.

catching + correcting these things in prototyping is a lot cheaper, easier + *less stressful* for everyone involved.

IMAGE: Pexels, pexels.com

3

the earlier you find + fix problems, the easier it is.

if we've made a wrong turn, I want to know *now*, at the beginning of the journey.

when it costs us nothing to turn the car around and try again.

if users will hate a feature, too expensive or time-consuming to build, I want to know *before* we've spent time + money.

IMAGE: Pexels, pexels.com

3

the earlier you find + fix problems, the easier it is.

if we learn this 6 weeks from now...

1. **we have to scrap what we have** and...
2. **re-work the entire thing**, which, in complex systems, usually also means...
3. **decoupling + reimagining** any number of third-party module or system dependencies.

IMAGE: Pexels, pexels.com

4

prototyping allows you to judge design feasibility.

requirements and specifications are words on paper; a prototype is a **dose of reality**.

teams can quickly experiment, explore + test to see if proposed interaction or data/content will give users what they need.

prototyping allows the team to get a handle on estimated effort and duration.

5

prototyping greatly reduces surprise requests.

managers + stakeholders can see for themselves exactly what is + **is not** going to be built.

so they're not left to wonder whether that list of things that exists *only in their heads* is going to make it into the build.

IMAGE: Pexels, pexels.com

6

closes the gap between design + development.

a Rosenfeld Media study found that when developers were given a prototype instead of flat designs and specifications to quote against and build to...

...development requests for clarification on design + UX **dropped by 80%**.

giving both the same goal to work toward.

IMAGE: Pexels, pexels.com

6

closes the gap between design + development.

designers and developers may speak multiple, different languages...

but a prototype speaks **all of them**.

quick iteration allows for collaboration; instead of “handing off,”devs are involved in the iteration, giving feedback on proposed features, interaction + functionality.

IMAGE: Pexels, pexels.com

7

answers the 3 critical questions of product design.

one: is this worth doing?

every minute spent researching, designing + developing a feature is an investment.

are we certain that every requirement on our list delivers a return on that investment?

7

answers the 3 critical questions of product design.

two: does everyone agree on + understand what we're creating?

does every member of the team share the *same understanding* of what's being built + why it matters?

7

answers the 3 critical questions of product design.

three: what value does it deliver?

does this improvement, feature, or function facilitate a desired outcome for both customers and the organization?

do we *know* it will deliver that value?

or are we *guessing?*

7

answers the 3 critical questions of product design.

tangible, evidence-based answers to these questions from prototyping + testing ensure we design + build features that matter.

standing them up onscreen in a way that users expect, can understand + are able to easily use.

7

answers the 3 critical questions of product design.

we know what we're doing.

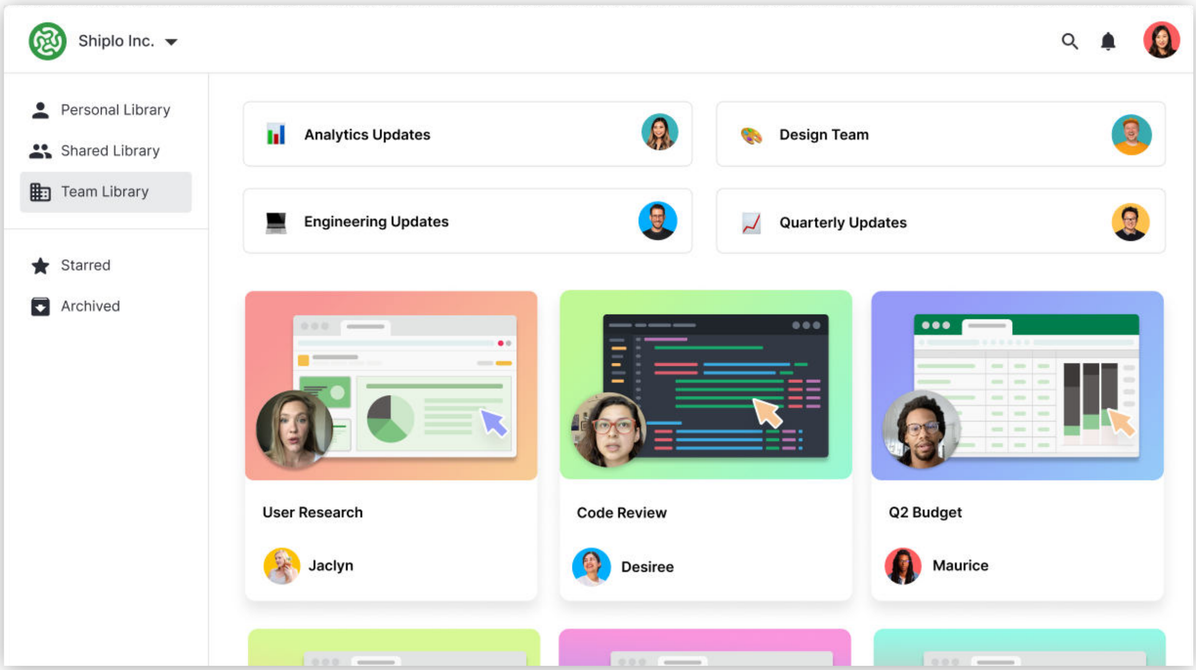
we know what we're *not* doing.

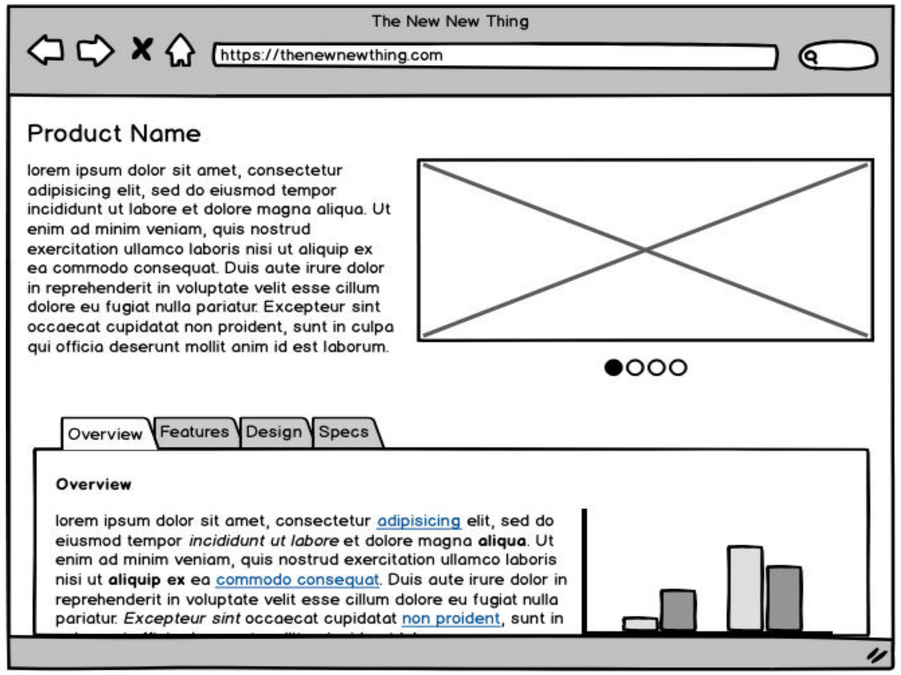
and in both cases...

we know WHY those things are or are not important.

how close to **“real”** should a prototype be?

what level of **fidelity** is appropriate?





should it look just like a **real-world product**?

should it be a **readable sketch** where everything is usable, but not styled?

it depends.

we use different levels of fidelity depending on what we need to **know + test + learn**.

differences in fidelity are most obvious in the areas of:

- **visual UI design**
- **content**
- **interactivity**

IMAGE: Pexels, pexels.com

it depends.

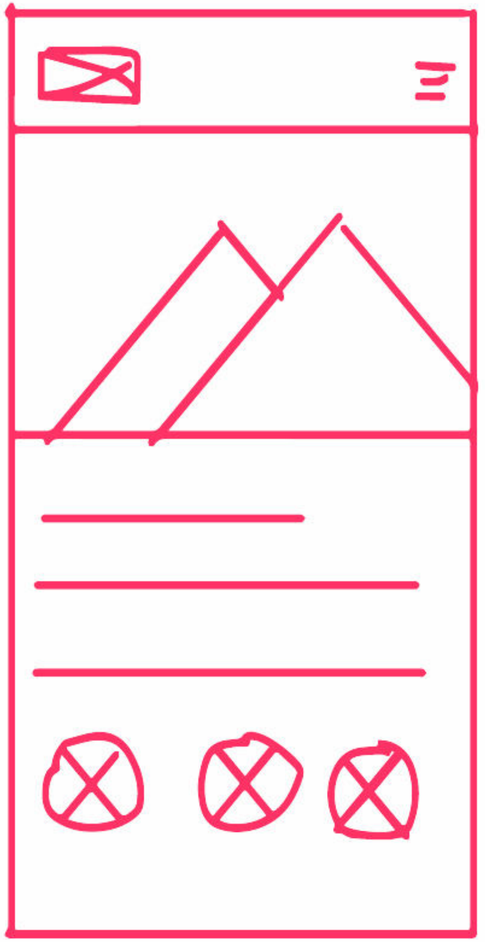
product teams choose the right fidelity based on three things:

- **goals** of prototyping
- **completeness** of design
- available **resources**

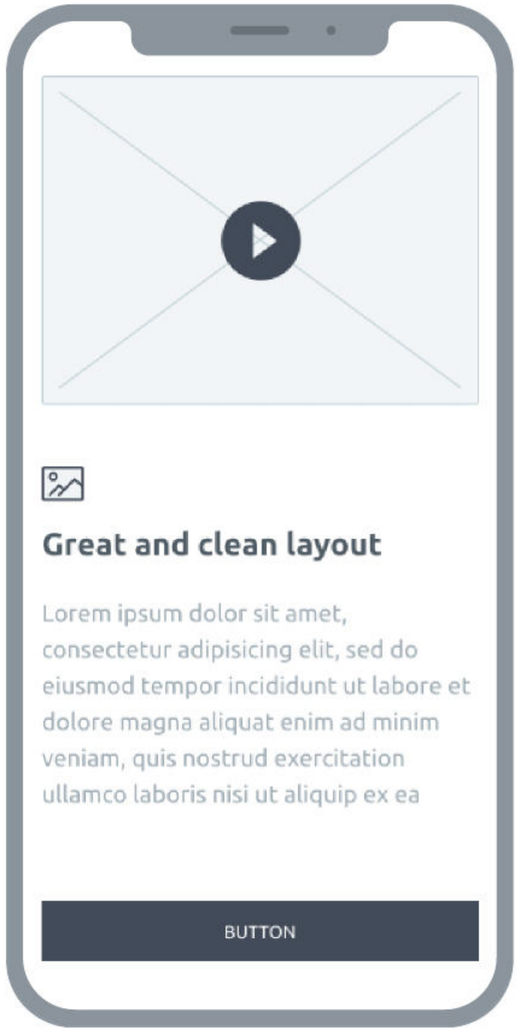
IMAGE: Pexels, pexels.com

there are **three levels** of fidelity to consider.

low-fidelity



mid-fidelity



hi-fidelity

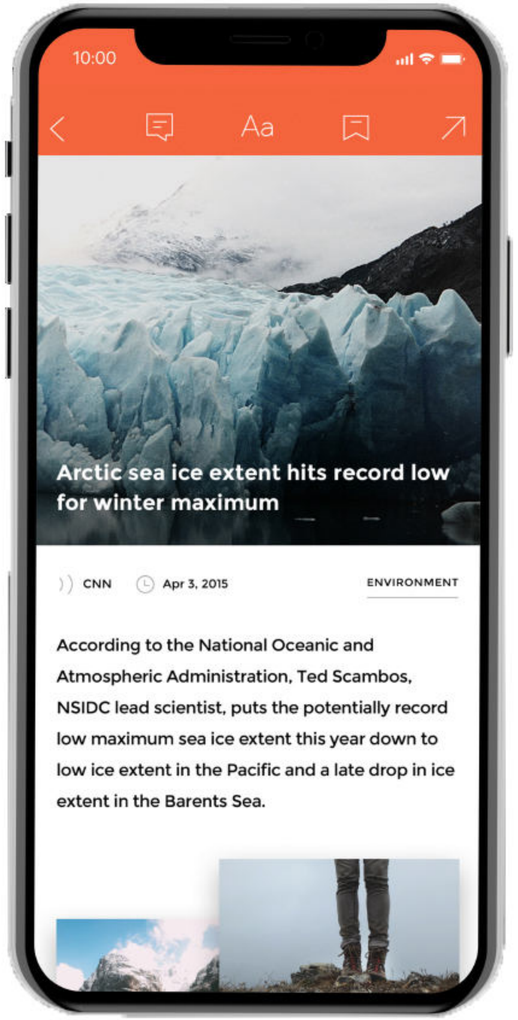
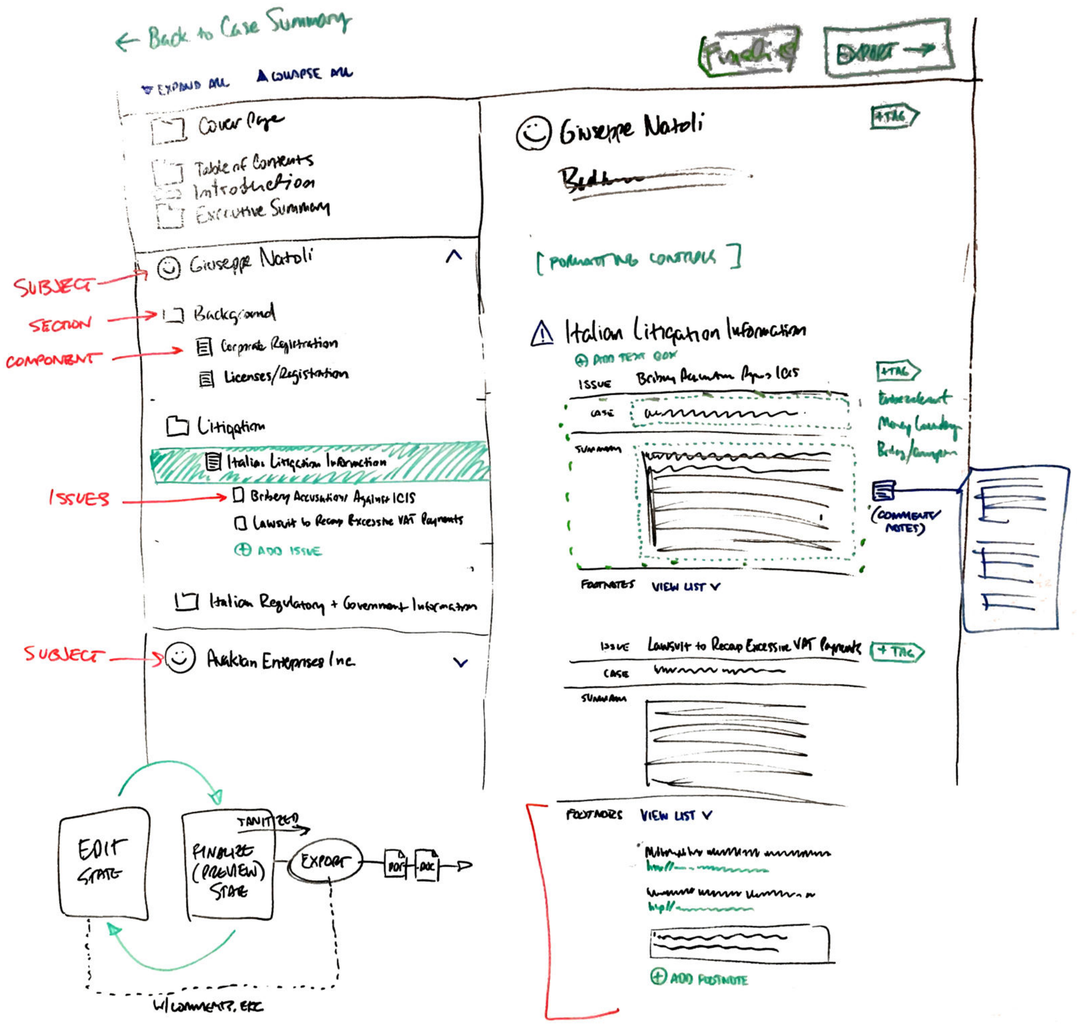


IMAGE: InVision, *Protoype Defined, https://www.invisionapp.com/design-defined/prototype/*

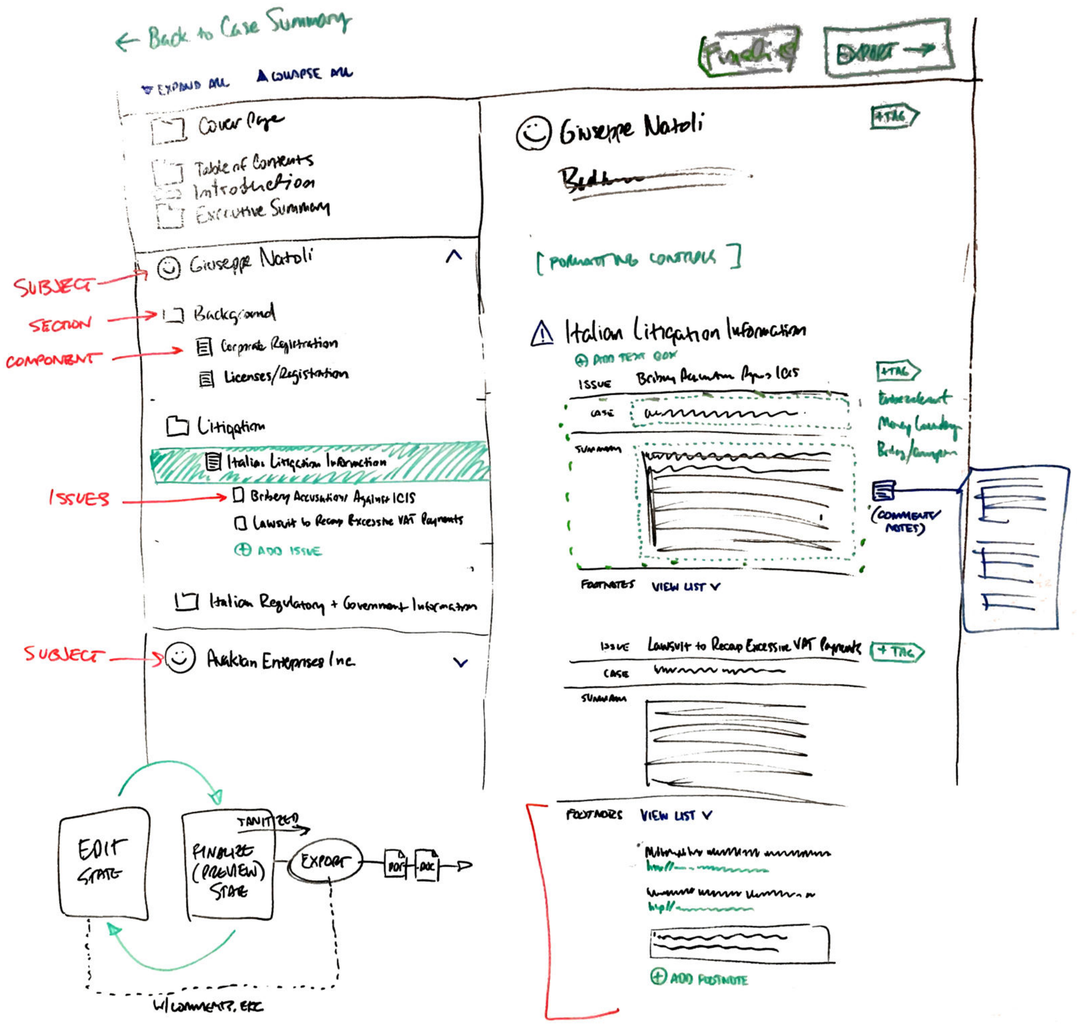
low-fi prototypes

- **hand sketched** (or whiteboard), paper-or computer-based.
- **limited user interaction** (or none if hand-drawn)
- **quickest to create**; no special tools or experience required; can be done alone with a sketchbook or in a group with a whiteboard.



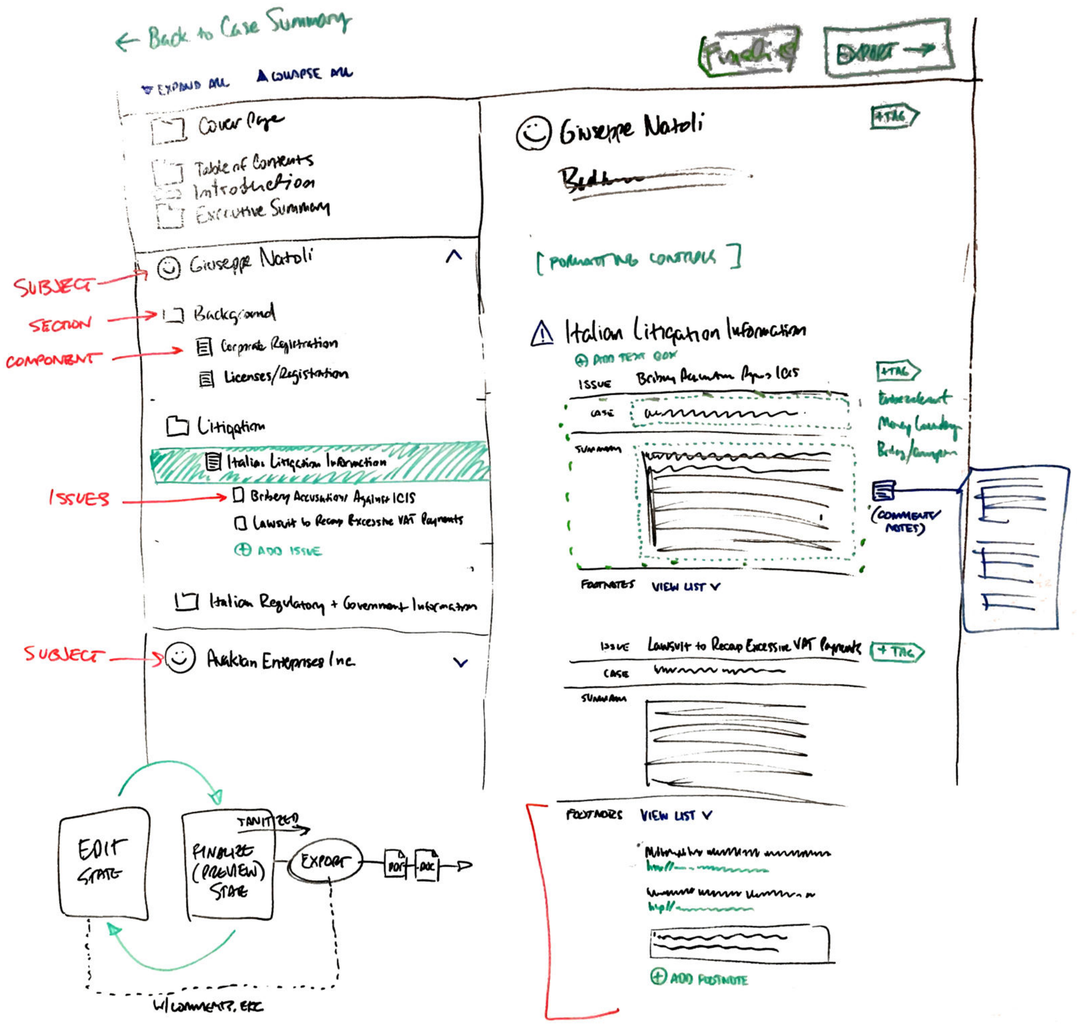
low-fi prototypes

- promotes innovation + improvement through **rapid iterations**.
- ideal for **multiple, iterative** design explorations.
- **quick way to get feedback from users** via rough approaches + concepts.



low-fi prototypes

- gives **comfort + psychological safety** to suggest changes.
- **invites collaboration, participation + feedback** because nothing feels “decided.”
- **ideal during brainstorming + conceptualization** work with stakeholders, developers, etc. to generate ideas.



mid-fi prototypes

- **happy medium** between high + low fidelity.
- **more content + UI detail is progressively added** based on learnings from lo-fi iteration + testing
- **minimal or “faked" interactivity** e.g. wireframes exported to clickable HTML format.

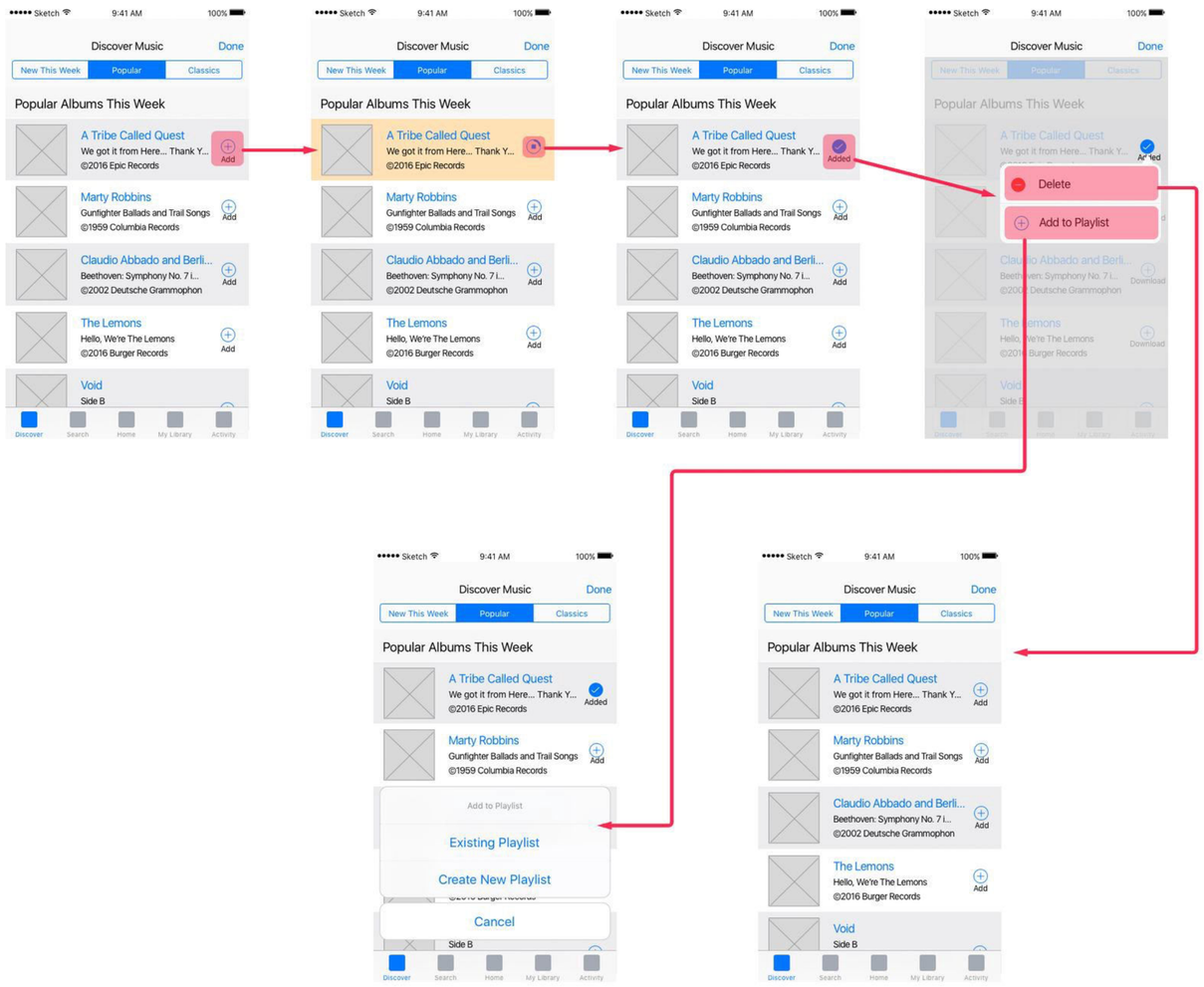


IMAGE: UX Collective, *Form and Function — a guide to the top wireframe tools*

mid-fi prototypes

- **true functional fidelity is minimal**; interactivity can be simulated by linking pages/ screens.

- **focus is on demonstrating the behavior** of the app, site or system, determining whether user needs are met + if the UX is appropriate + positive.

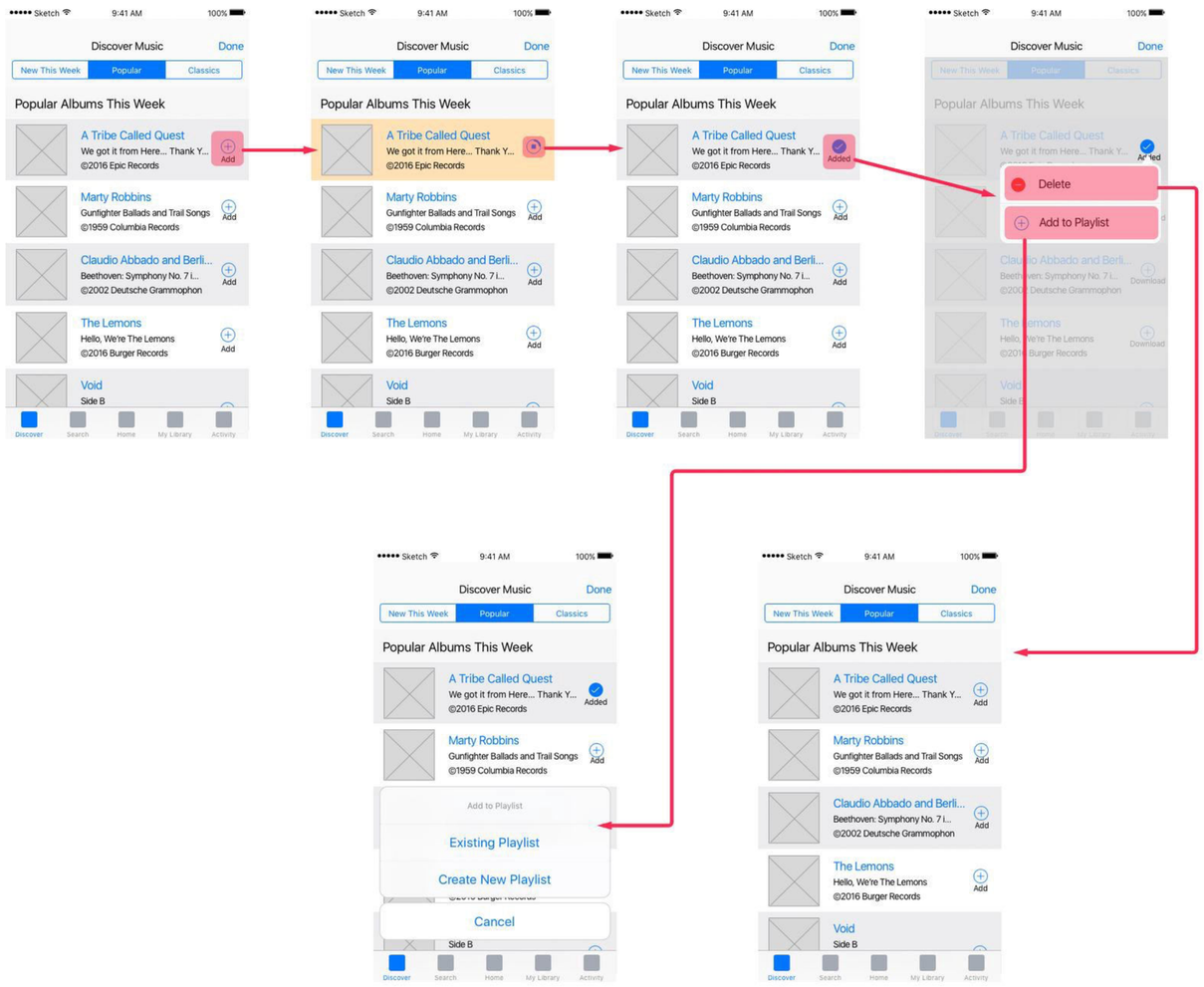


IMAGE: UX Collective, *Form and Function — a guide to the top wireframe tools*

hi-fi prototypes

- **realistic (click, tap, swipe, etc.) user interactions** that work on multiple devices.
- **exact representation of the UI + its interactivity**; true simulation of form + function; like using the real thing.
- **ideal for collecting measurable data** on user behavior.

low-fidelity prototyping

goal: check + test usability + functionality

we're not concerned with how the UI should *look*; our focus is on:

- how things **work**.
- how **well** they work.
- whether or not those interactions will be **understood + can be used** by people.

IMAGE: Pexels, pexels.com

working in low-fidelity **early on**
in a project is always wise.

why?

IMAGE: Pexels, pexels.com

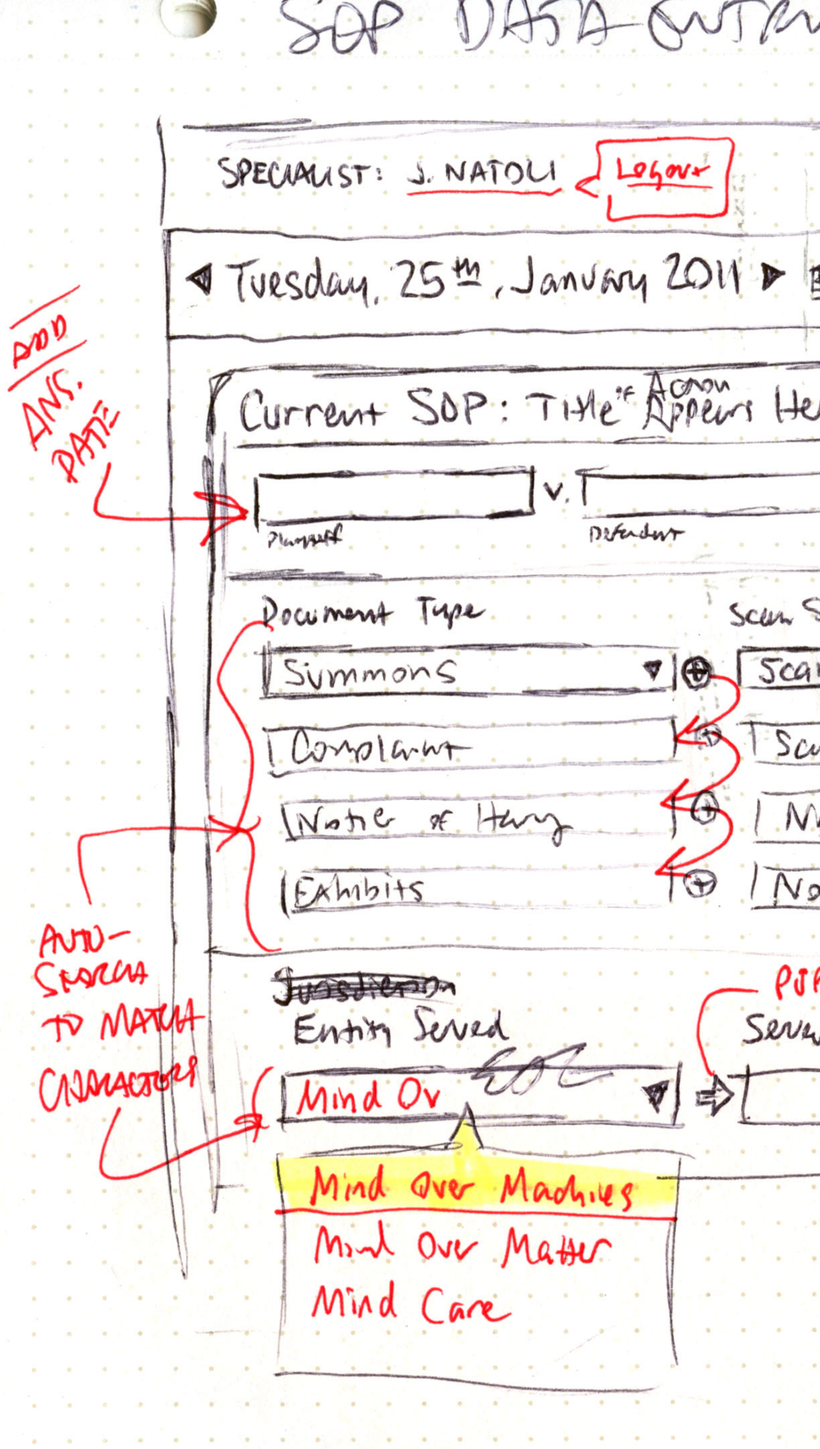
it’s not their fault.

it’s **ours**.

IMAGE: Pexels, pexels.com

we're failing to give stakeholders what they need to evaluate requirements.

so they can see the **gaps** between what people need or expect + what we're proposing to build.



the longer stakeholders have to wait
to see something visual...

...the bigger the gap will be between
what they expect + **what you build**.

in the absence of evidence,
personal preference wins.

IMAGE: McDonalds, *mcdonalds.com*

what typically happens

product requirements are all too often *“gathered.”*

they come from meetings where stakeholders **talk about what they want** — the design/dev team *captures* those items.

these aren’t really requirements.

they’re **assumptions + guesses**.

IMAGE: Pexels, pexels.com

what typically happens

instead of drawing from actual user research, those mandates are a combination of:

- **personal opinion**
- **political pressure** (fear)
- **knee-jerk reaction** to competitor features

IMAGE: Pexels, pexels.com

there's only one way to prevent this.

build a simple prototype. ***immediately***.

then iterate *with* stakeholders + developers to *generate* requirements.

if it slows things down, **you're doing it wrong**.

software makes it ridiculously easy for anyone to build a **solid low-fidelity prototype** in an hour for feedback.

I've seen a **two-hour review session** with stakeholders replace one or more two-week sprints.

BUT: iteration and review/discussion must happen *simultaneously*.

IMAGE: Pexels, pexels.com

the key phrase here: **low-fidelity**.

you *must* stick to the following constraints, otherwise you'll spend more time than necessary.

and you'll be inviting everyone who sees it to fixate on **what it looks like** instead of **what it *does***.

lo-fi prototyping has to be three things...

FAST

+

INEXPENSIVE

+

ACCURATE

rules for low-fidelity prototyping

1. **limited colors (blue for buttons + hyperlinks)**; everything else is filled with shades of grey to indicate visual hierarchy.

2. **no images or graphical data displays**; introduce either and reviewers will fixate on them instead of casting a critical eye to layout, content structure, controls, interactivity + workflow.

rules for low-fidelity prototyping

3 **no fonts other than Arial**; as with colors and images, fonts only invite analysis and speculation.

4 **real labels on all interactive components**; give real examples in order to get good feedback, even if nothing's approved: navigation menu items, data tables, buttons, form fields, etc.

rules for low-fidelity prototyping

5

real text content, to whatever degree possible; content provides *context* to design around; a rough draft or guess at your content is much better than “lorem ipsum.”

6

keep interactions simple; anything requiring significant, real coding for demonstration purposes is something you need to drop, re-think, or table for a hi-fi prototype.

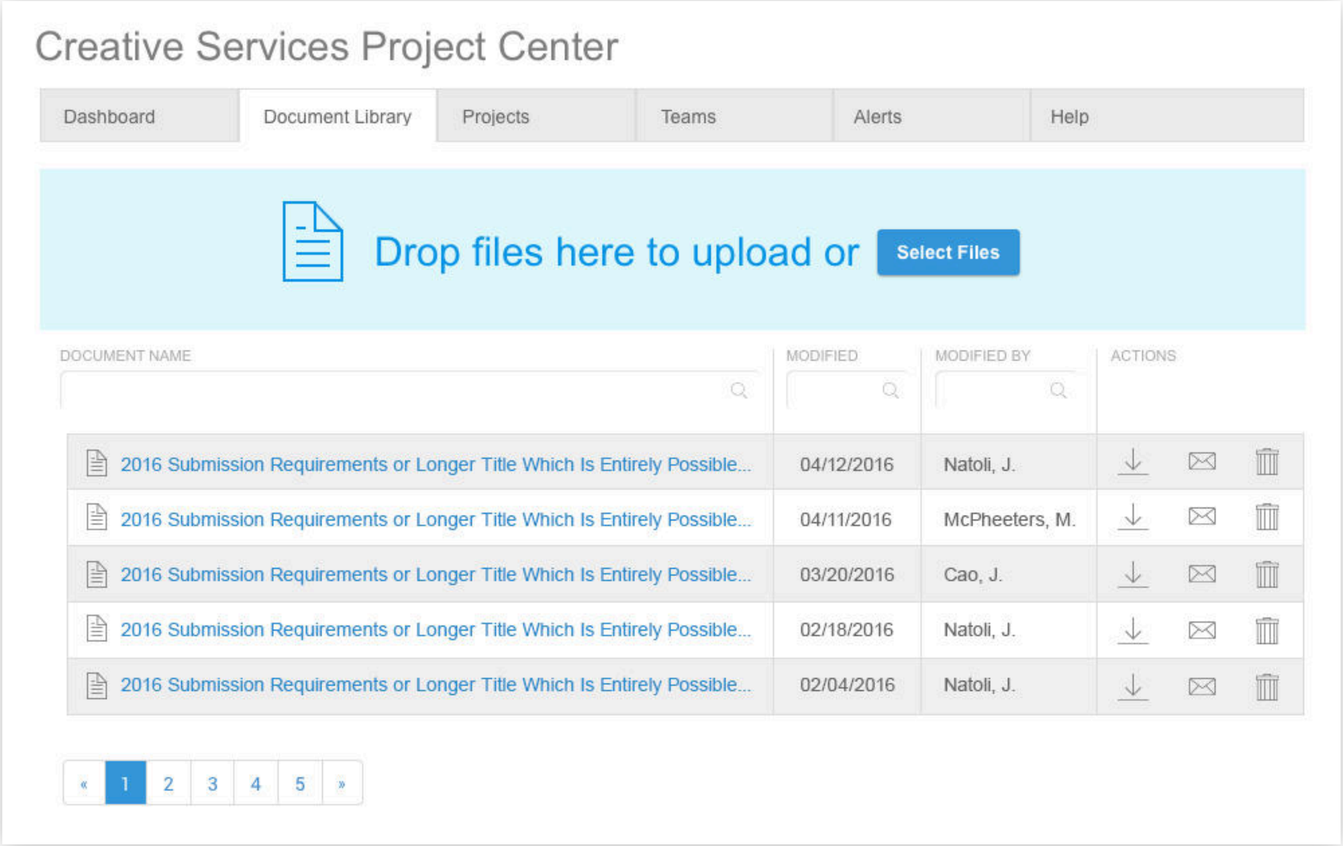
rules for low-fidelity prototyping

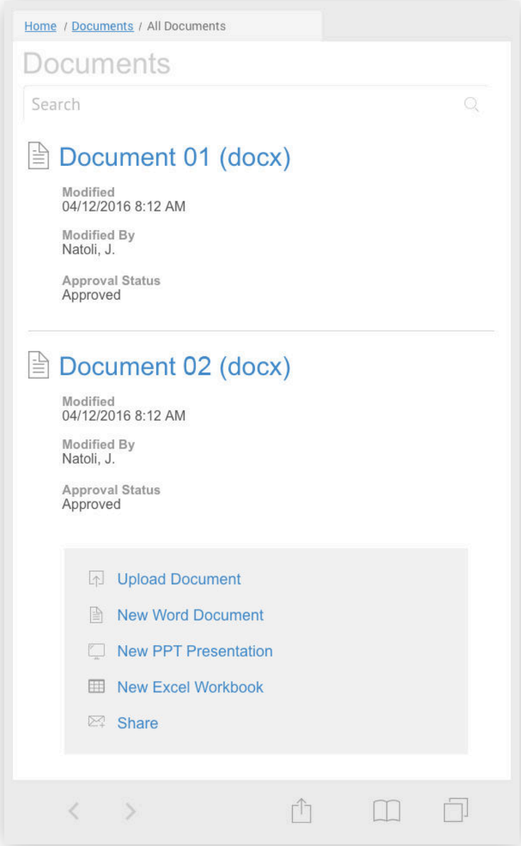
7

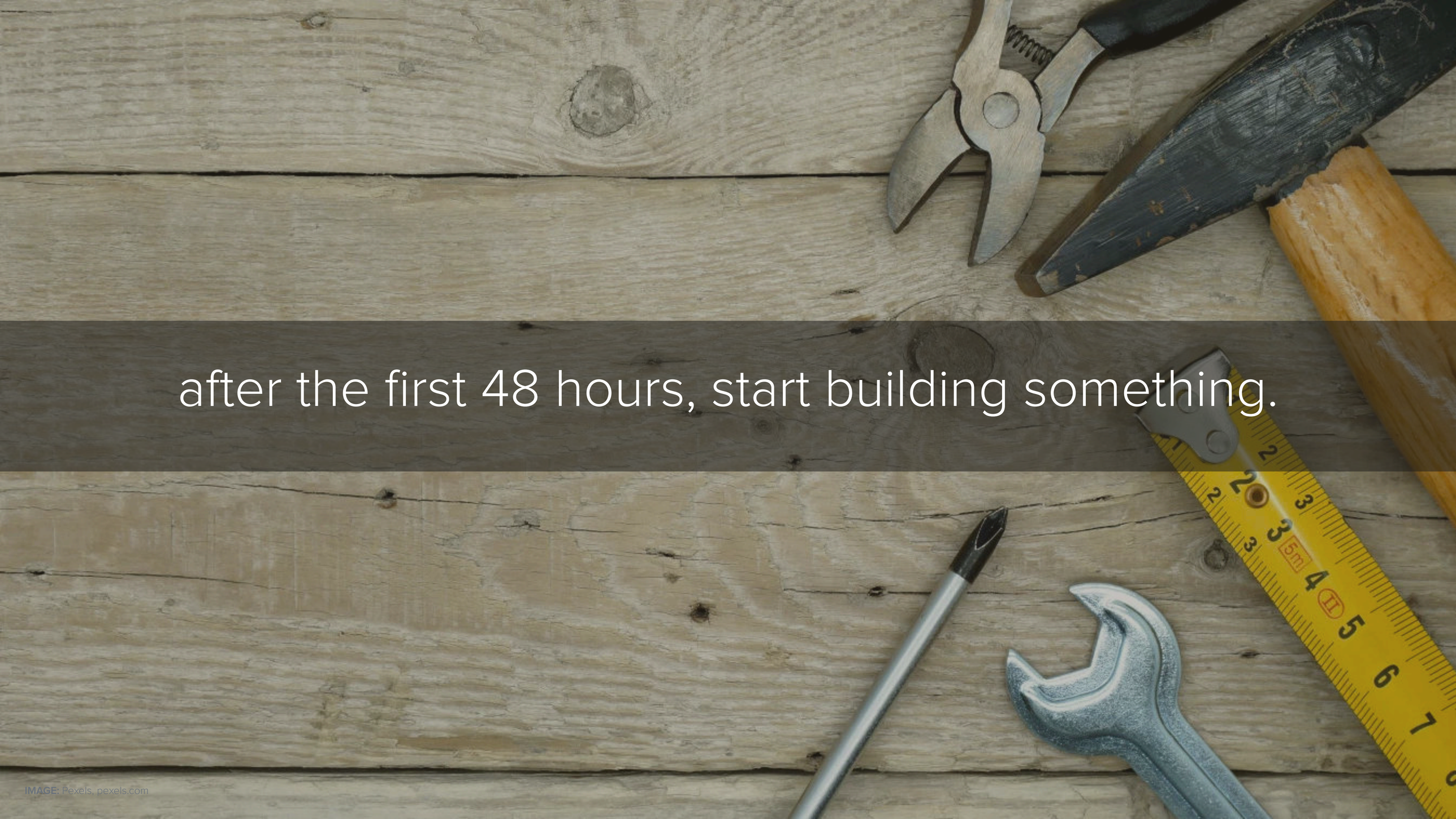
annotate to begin suggesting and documenting functionality and requirements; it's your responsibility to communicate the rationale of all interaction decisions.

annotated prototypes reduce misinterpretation — you create *contextual* rather than "paper trail" documentation.

sample low-fidelity prototypes







after the first 48 hours, start building something.

IMAGE: Pexels, pexels.com

there are always more questions than answers.

at the outset of a project, that's the normal order of things.

some people insist to try to **answer them all** before doing anything.

some people think they **already know** all the answers.

guess how *that* works out?

IMAGE: Pexels, pexels.com

a parable: the mystery messaging project

"we need messaging – for evaluators to talk to approvers.

but other roles might be involved.

they should only be able to talk to each other, but there might be times when they need to talk to other departments.

and clients.

but only about certain issues.

it should work like Outlook.

except no mailboxes or sending anything.

a parable: the mystery messaging project

it should be like a comment thread.

but not everyone can reply.

there shouldn't be a central messaging screen, because it may need to appear on other screens.

what screens the messaging feature appears on depends on what's being asked.

and we need to notify everyone when a question is asked."

this was *literally* all the information the team got.

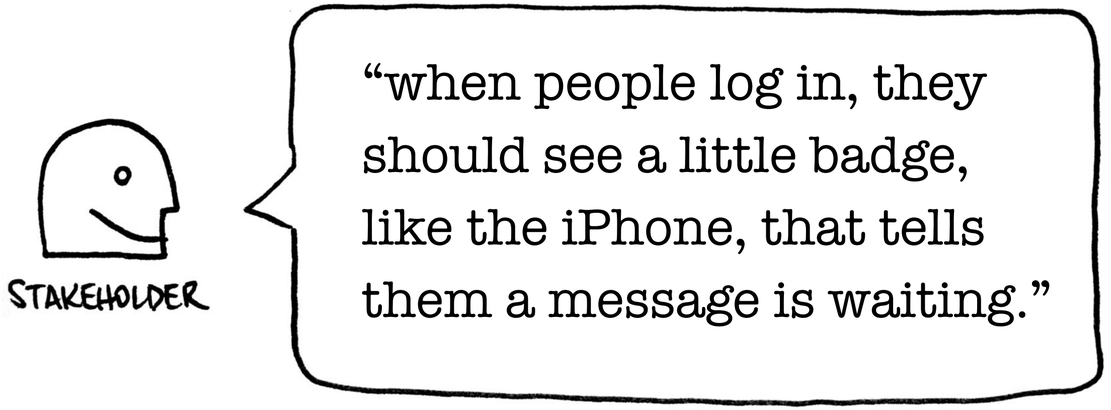
I'm sure you're seeing the red flag lurking here.

anything this wide-open and undefined **has no business being a requirement**.

we didn't plan on including comment threads on every screen + we weren't convinced that doing so even made sense.

but we could see the need for communication in context, so we agreed to **prototype it**.

sprint one prototype review



a badge *where?*

remember, there's no central "messages" screen; messages were simply **comments** related to a specific summary.

our scope has expanded exponentially: we are now building some kind of messaging app, complete with alerts + notifications.

which no one is really sure is *necessary*, let alone how it might work.

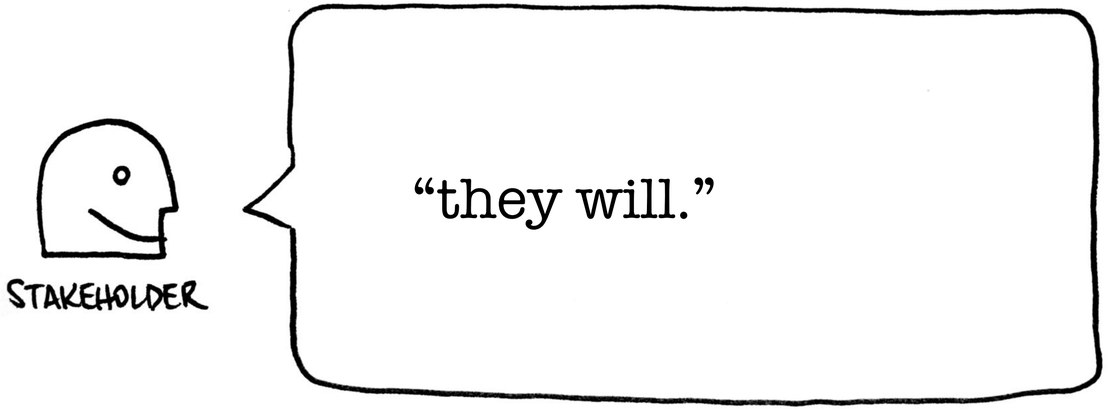
sprint one prototype review

so we ask:

“shouldn’t there be some kind of alert or prompt that a customer sees when checking their status that this is requested/needed?

how do we know they’ll bother to check messaging?”

the answer:



almost $70K is being spent on two words: *they. will.*

we voice these concerns as thoroughly + diplomatically as possible.

we suggest this messaging component is big enough to warrant a **separate system and/or project** in + of itself.

it'll be **time + cost intensive**.

but you know the end result:

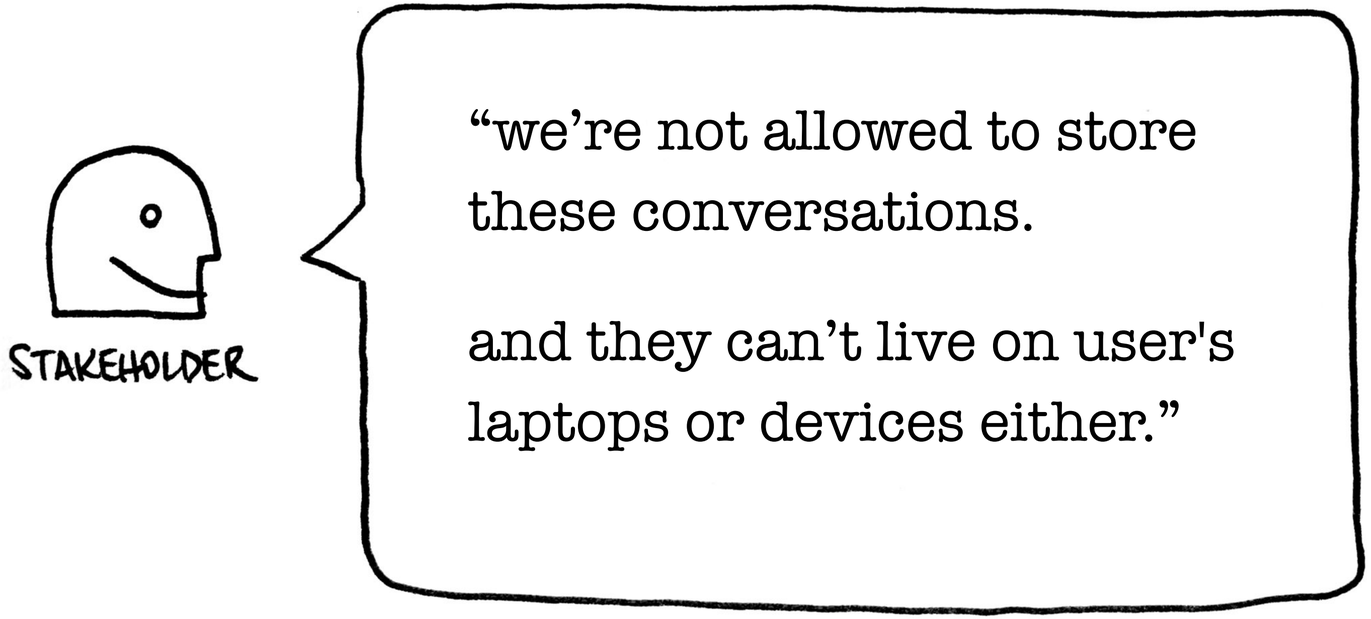
we run the next sprint.

sprint two review

we now have a **messaging dashboard** in place similar to Outlook's approach to email.

threads, filters, address book, the whole nine yards.

in the middle of the review session, **this gem** comes out:



this is a black hole of *epic* proportions.

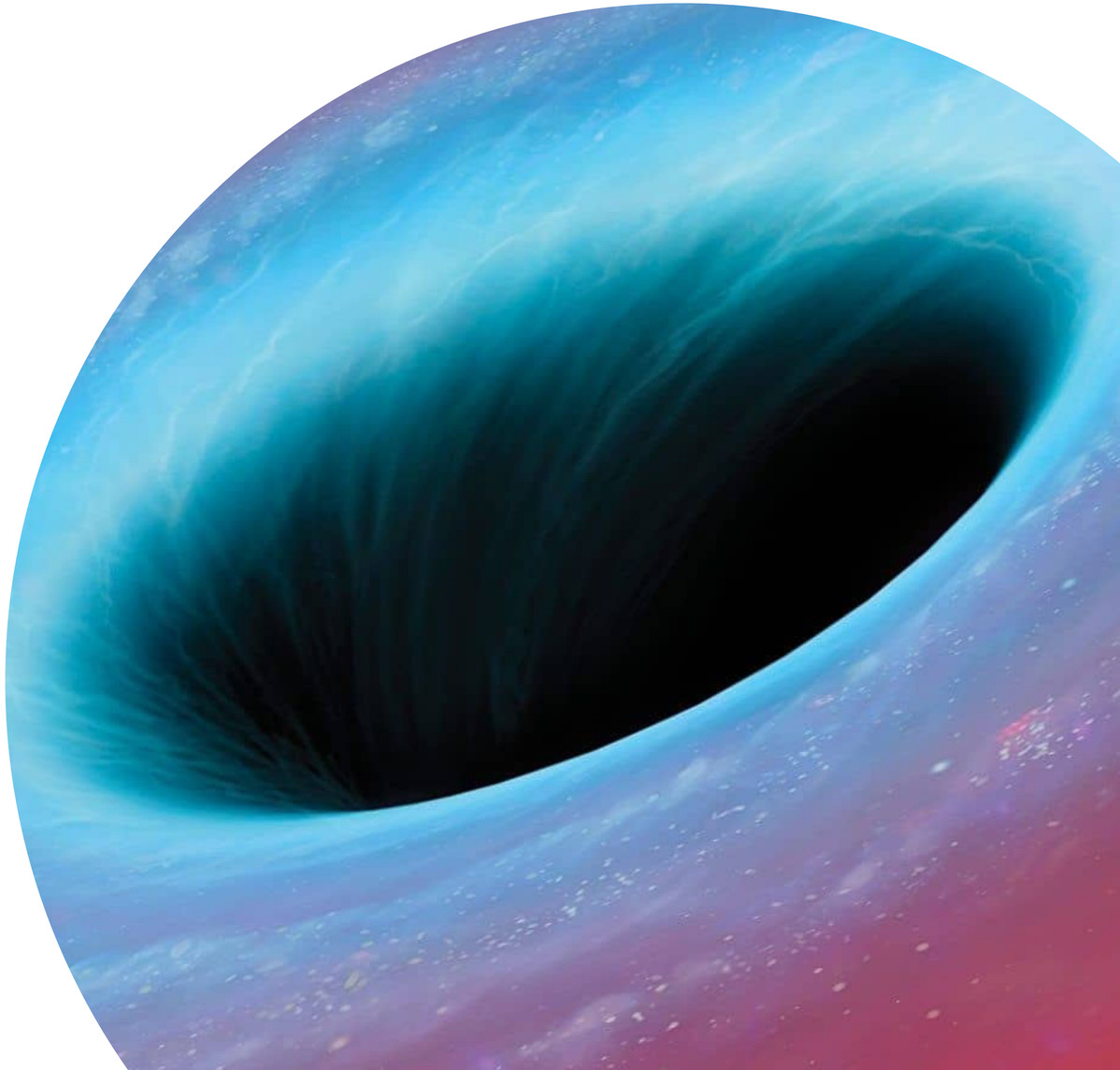
at the very least, it suggests that a canned process for alerting + collecting missing data needs to be considered in detail.

in a way that doesn't expose the customer or the organization to *any* degree of legal liability.

no small thing, to say the very least.

we collectively agree to table it until after launch of version 1.0.

which never happens.



the only way to get clarity...

the only way to know for sure what's needed...

or *possible*...

is to start prototyping.

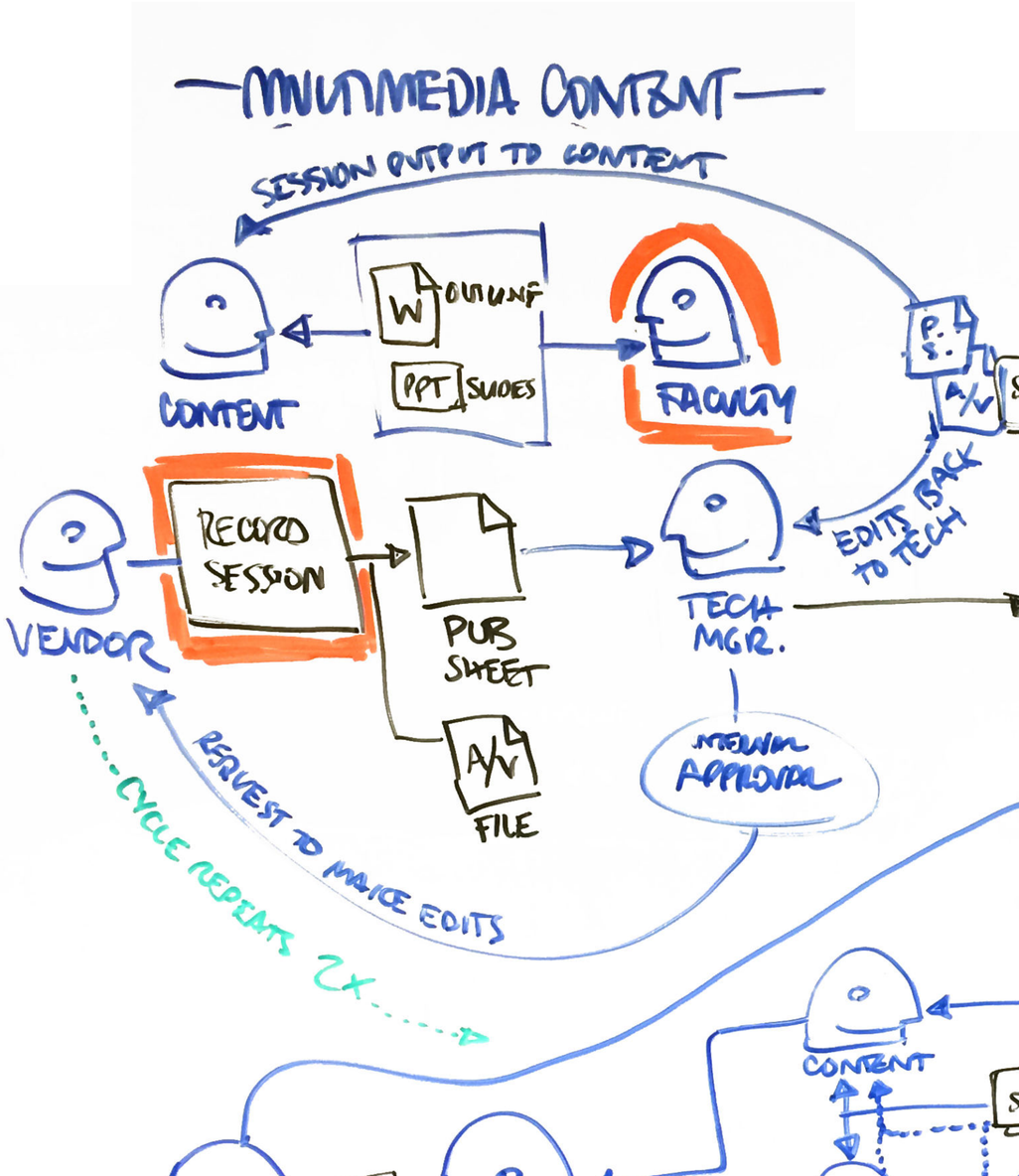
my 48-hour rule plays out like this.

days 1 + 2: contextual use scenarios

first: get the lay of the internal land, **politically** as well as tactically.

what does each stakeholder expect to happen? where do their goals **conflict** with other people or departments?

these collaborative sessions focus on a **specific user group**, or a specific **department** in the organization.

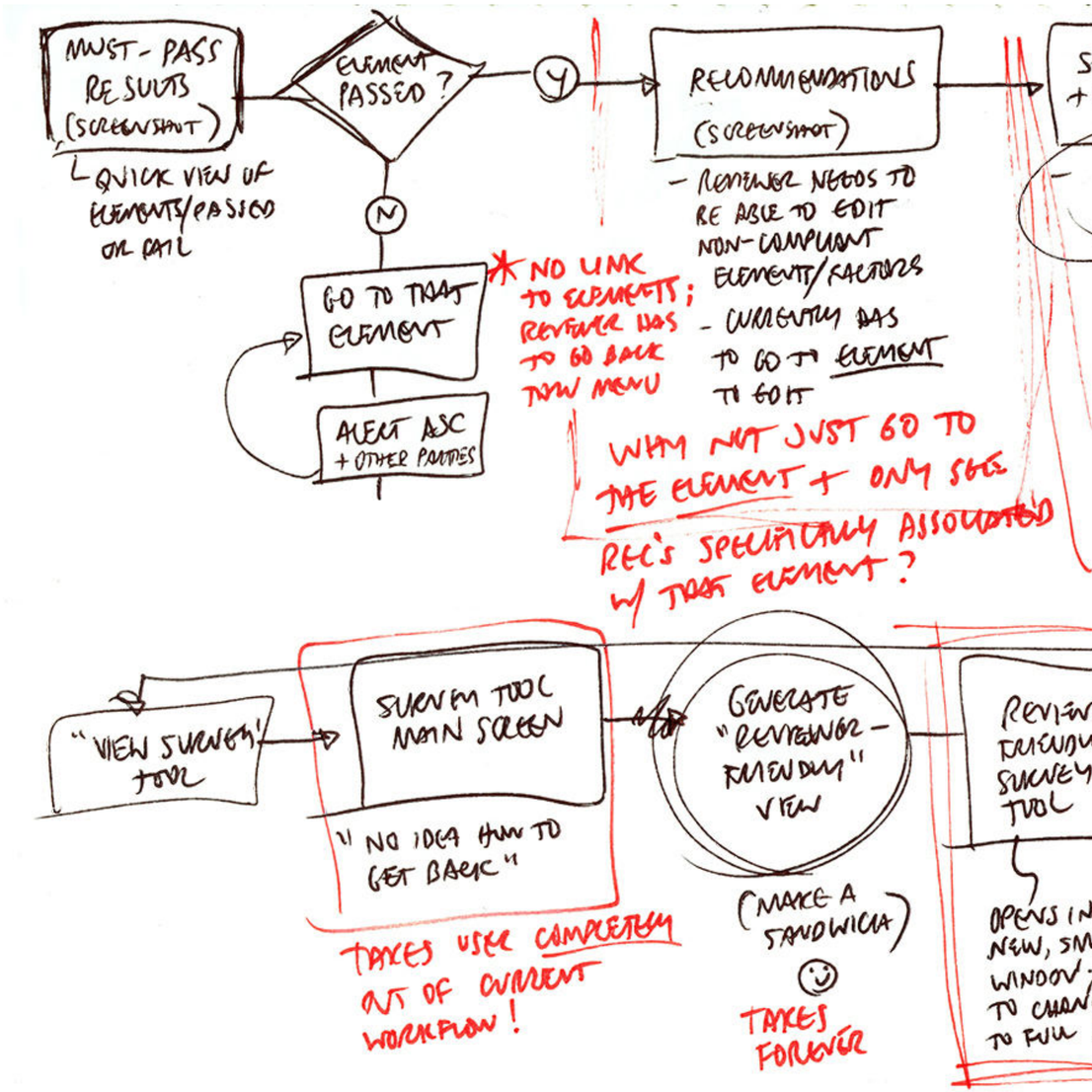


days 1 + 2: contextual use scenarios

the day is spent walking through:

- **how people use the system now**
- **how we *think* they'll use the new system**
- **diagramming these processes**

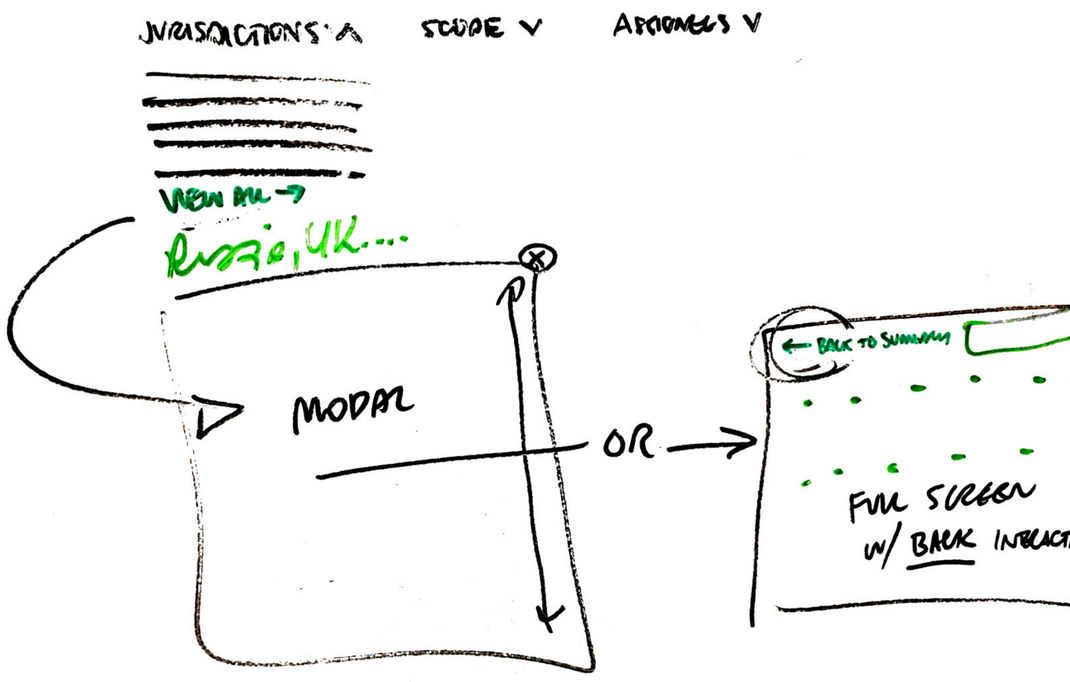
lo-fi whiteboarding is what makes this *collaboration* instead of *combat*.



day 3: prototyping information + interaction

you keep two things distinct:

- **things the user needs to *know*.** accessible, but out of the way, used only when needed, doesn't obscure or compete with the user's action.
- **things the user needs to be able to *do*.** this stuff is all up-front and center, mission-critical things the user needs right *now*.



← Back to Case Summary

Finalize | Export →

EXPAND ALL | COLLAPSE ALL

- Cover Page
- Table of Contents
- Introduction
- Executive Summary

SUBJECT → Giuseppe Natoli

SECTION → Background

COMPONENT → Corporate Registration

Licenses/Registration

Litigation

Italian Litigation Information

ISSUES → Bribery Accusations Against ICIS

Lawsuit to Recap Excessive VAT Payments

⊕ ADD ISSUE

Italian Regulatory + Government Information

SUBJECT → Avakian Enterprises Inc.

Giuseppe Natoli

+TAG

~~Background~~

[FORMATTING CONTROLS]

⚠ Italian Litigation Information

⊕ ADD TEXT BOX

ISSUE Bribery Accusations Against ICIS

CASE

SUMMARY

FOOTNOTES VIEW LIST ∨

+TAG

Enforcement

Money Laundering

Bribery/Corruption

(COMMENTS/ NOTES)

ISSUE Lawsuit to Recap Excessive VAT Payments

+ TAG

CASE

day 3: prototyping answers questions from prior work

- **how much content** and/or data do we need to expose?
- how many **different types/formats** do we have (e.g. data, text, audio, video, etc.)?
- how much of the content is **static** vs. **dynamic** or **interactive**?
- **where is it coming from** – how many sources or systems?
- how will people **get to all of it**?
- how should it be **organized**, **categorized**, **prioritized**, **labeled**?
- how will we **stand it up** in the UI?
- what **interaction patterns** match what the user needs or expects?
- is each pattern **appropriate** for the *volume* and *type* of content they're manipulating or viewing?

day 3: try, evaluate, revise

to answer those questions quickly, we're quickly **iterating + exploring** data types, navigation, labeling, interaction patterns.

we're **socializing** each iteration with users, product owners, stakeholders.

we keep what works + ruthlessly *throw out* what doesn't.

IMAGE: Pexels, pexels.com

JOE'S TIP

if it's undefined after two sprints...

table it or **drop it**.

if you can’t figure it out after two passes...

1. it’s a complex issue **deeply entrenched in associated processes or systems**...

or

2. you’re dealing with **too many unknowns**.

in either case, *it waits*.

characteristics of **low-fidelity** prototypes

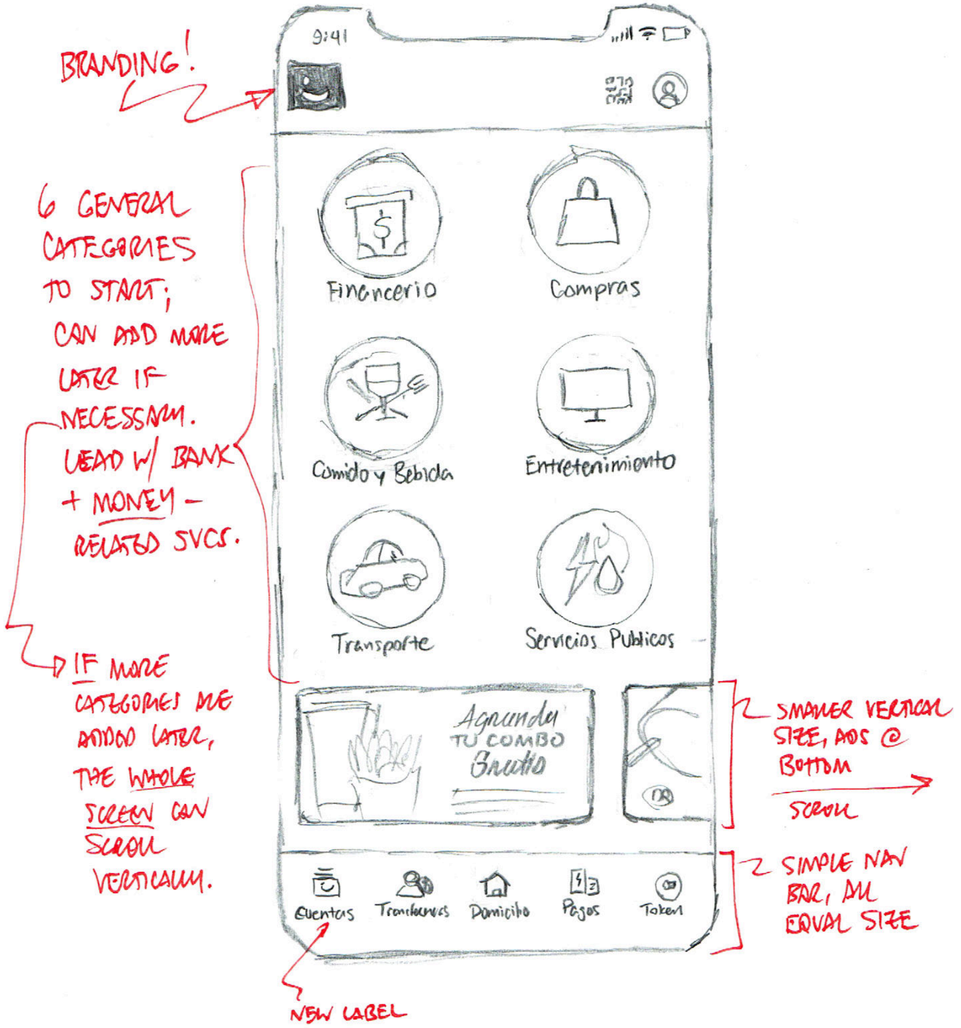
1

bare-minimum visual design

visual attributes of the app, site or system are represented, but **not styled** in any way.

a lo-fi prototype lacks font style, color + graphics; it can be a sketch.

the main focus is on **functionality**, **interaction** + **volume, organization + priority of content**.



1

bare-minimum visual design

it's a *skeleton* for the digital product you're designing.

it's a rough sketch; sometimes one you can click or tap through.

the purpose is to allow you, your team + your stakeholders to focus on **how people will browse + interact** with the site, app or system.

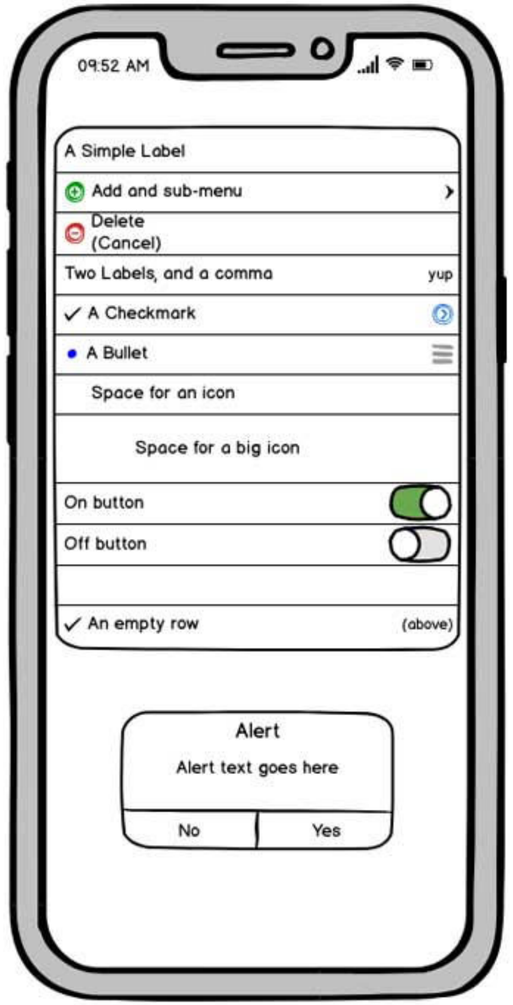


IMAGE: Balsamiq, *Balsamiq.com*

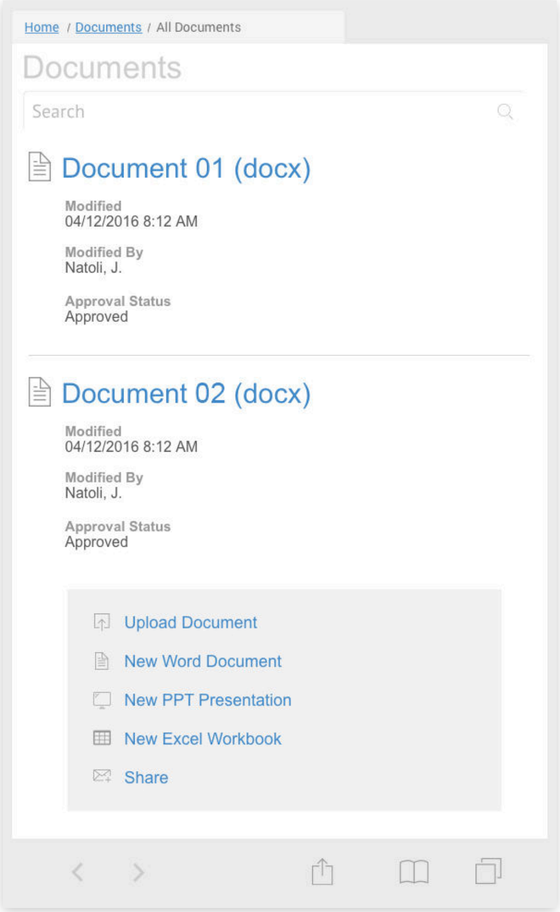
1

bare-minimum visual design

we only include **basic elements + shapes**; everything is black or shades of grey.

the sole exception: **blue for hyperlinks**, so interactive elements stand out for your test users.

the only design decisions you're making are **spatial**; how much will fit on the screen? *where?*



Creative Services Project Center

Dashboard | Document Library | Projects | Teams | Alerts | Help

Drop files here to upload or **Select Files**

DOCUMENT NAME	MODIFIED	MODIFIED BY	ACTIONS
2016 Submission Requirements or Longer Title Which Is Entirely Possible...	04/12/2016	Natoli, J.	
2016 Submission Requirements or Longer Title Which Is Entirely Possible...	04/11/2016	McPheeters, M.	
2016 Submission Requirements or Longer Title Which Is Entirely Possible...	03/20/2016	Cao, J.	
2016 Submission Requirements or Longer Title Which Is Entirely Possible...	02/18/2016	Natoli, J.	
2016 Submission Requirements or Longer Title Which Is Entirely Possible...	02/04/2016	Natoli, J.	

« 1 2 3 4 5 »

.com

1

bare-minimum visual design

if your goal is to get feedback on functionality and interaction, or to make sure every screen and sequence and element is present and accounted for...

your prototype *cannot* look like this.

you'll never get the feedback you really *need*.

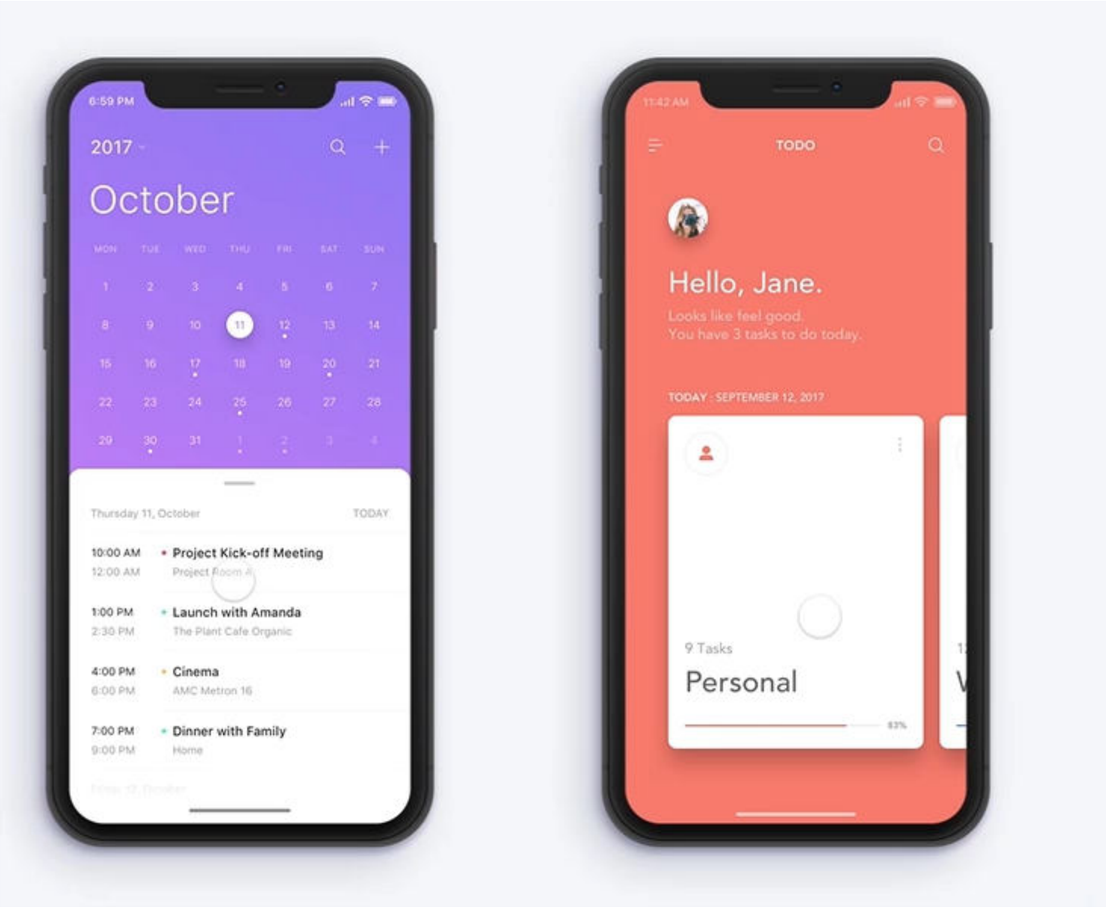


IMAGE: Marvel, *marvelapp.com*

1

bare-minimum visual design

you'll spend all your time **redirecting conversation** to get stakeholders or users to respond to appropriateness of navigation, interaction cues or the structure + organization of content.

you'll be stuck hearing some variation of *"I really don't like that color scheme"* on endless repeat.

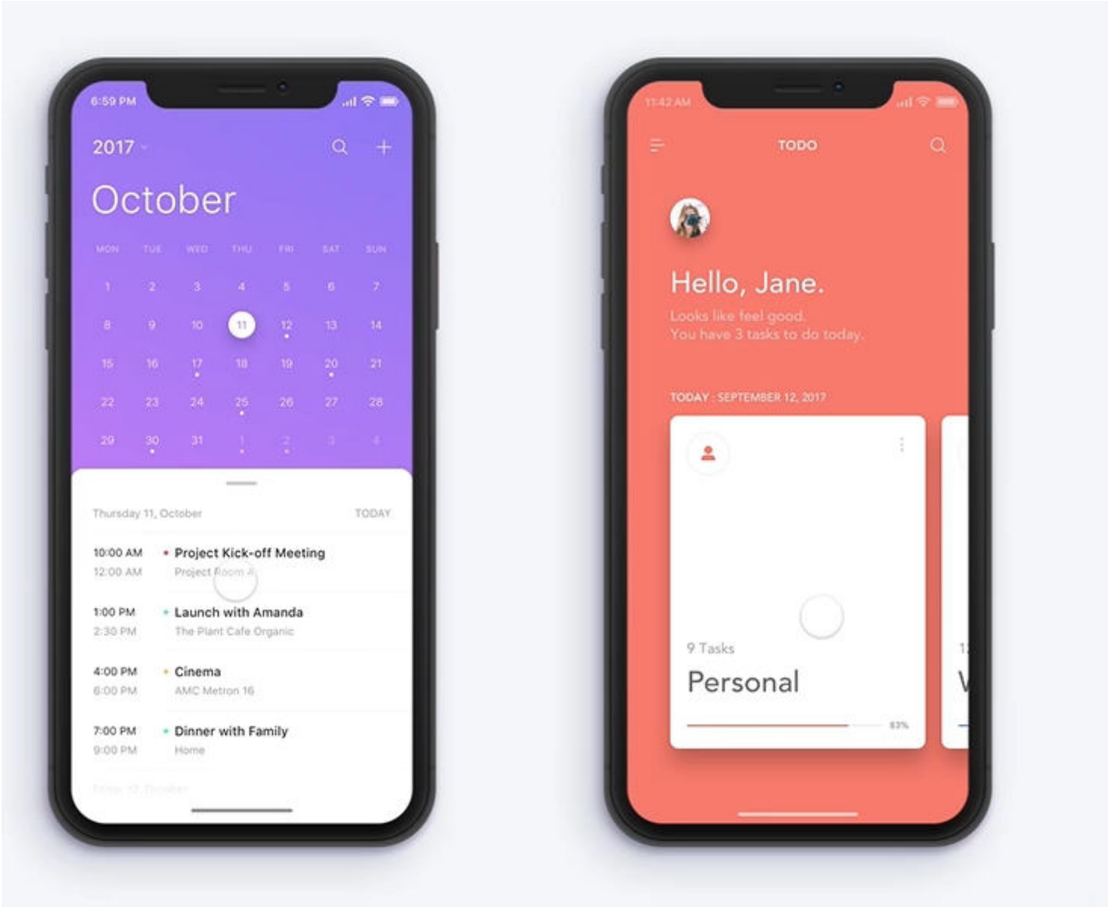


IMAGE: Marvel, *marvelapp.com*

2 only include key content elements.

the goal is to indicate **how much content** there might be.

along with how it'll be **organized**, **prioritized** + **labeled** on any given screen + across *all screens*.

you're trying to figure out **chunking + length**; you cannot use *lorem ipsum* for this!

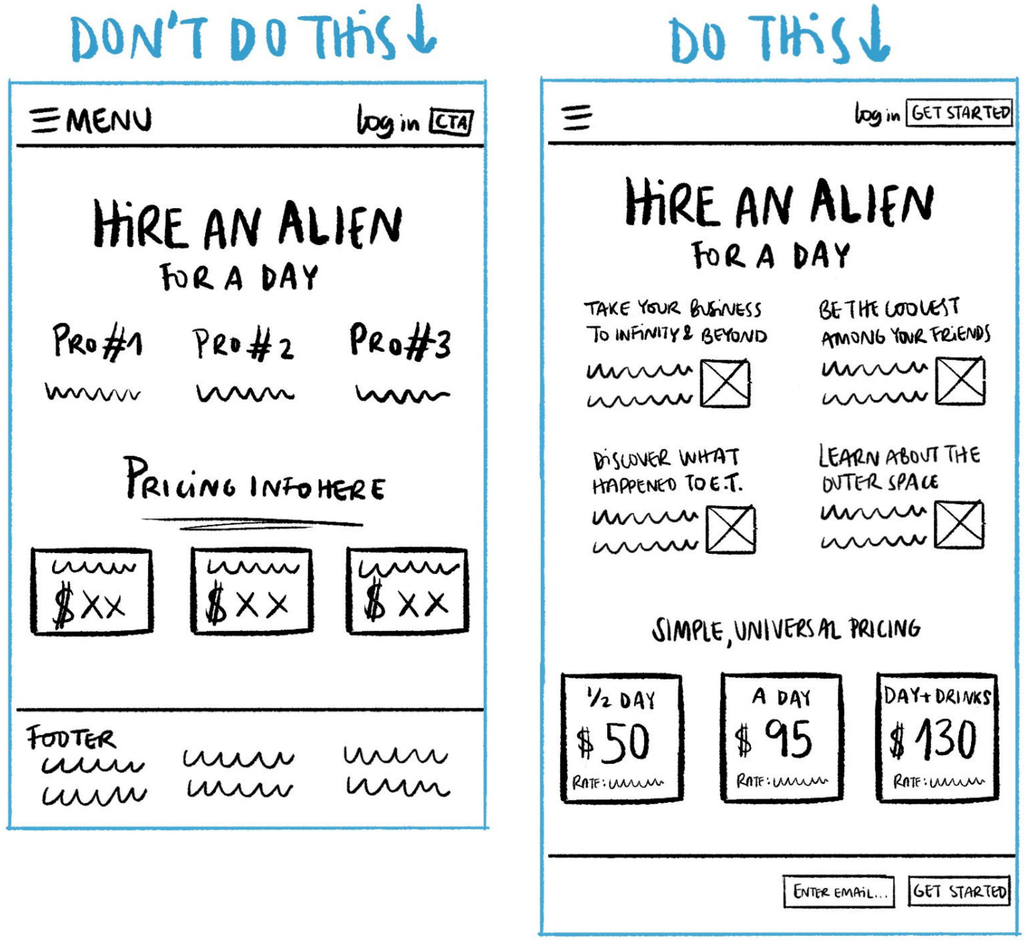


IMAGE: Biz Sanford, Words and the design process, *https://ux.shopify.com/words-and-the-design-process-f41472a249fb*

which leads us to an age-old debate:

which should come first: content or design?

all too often, UI design happens before content.

that's like building the house before **we understand how many rooms it should have or how big it needs to be**.

we have thousands of fully-designed
themes, **UI kits** and **website builders**.

they make UI design seem *instantaneous*.

so the temptation to **design in a vacuum**
becomes *very* strong.

Bootflat .sketch v0.1.0

Color Swatches

BLUE JEANS #5D9CEC, #4A89DC	AQUA #4FC1E9, #3BAFDA	MINT #48CFAD, #37BC9B	GRASS #A0D468, #8CC152	SUNFLOWER #FFCE54, #F6BB42	BITTERSWEET #FC6E51, #E9573F
GRAPE FRUIT #ED5565, #DA4453	LAVENDER #AC92EC, #967ADC	PINK ROSE #EC87C0, #D770AD	LIGHT GRAY #F5F7FA, #E6E9ED	MEDIUM GRAY #CCD1D9, #AAB2BD	DARK GRAY #656D78, #434A54

Buttons

NORMAL HOVER PRESSED NORMAL HOVER PRESSED

NORMAL HOVER PRESSED NORMAL HOVER PRESSED

NORMAL HOVER PRESSED NORMAL HOVER PRESSED

NORMAL HOVER PRESSED

NORMAL DEFAULT PRIMARY SUCCESS INFO WARNING DANGER

NORMAL DEFAULT PRIMARY SUCCESS INFO WARNING DANGER

Labels

NORMAL DEFAULT PRIMARY SUCCESS INFO WARNING DANGER

Badges

NORMAL DEFAULT PRIMARY SUCCESS INFO WARNING DANGER

Tooltips

Tooltip on top Tooltip on right Tooltip on left Tooltip on bottom

Dropdown

Settings
Action
Another action
Something else here
Separated link

Action1
Action2
Another action
Something else
Separated link

Popover

Popover top
Sed posuere consectetur est at lob ortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover right
Sed posuere consectetur est at lob ortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover left
Sed posuere consectetur est at lob ortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover bottom
Sed posuere consectetur est at lob ortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Forms

Text input
Text input
$ .00
Action Action
Search

Input with success

Input with warning

Input with error

Stepper

Selecter

One
GROUP ONE
One
Two
Three
GROUP TWO
Four
Five

Google Search

Checkbox and Radios

default disabled checked checked & disabled

default disabled checked checked & disabled

Toogles

Alerts

Warning! Best check yo self, you're not looking too good.

Heads up! This alert needs your attention, but it's not super

Well done! You successfully read this important alert

Oh snap! Change few things up and try submitting again.

Warning!
Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.
Take this action Or do this

Head up!
Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.
Take this action Or do this

Well done!
Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.
Take this action Or do this

Oh snap!
Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.
Take this action Or do this

Tabs

Home Profile Dropdown

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica.

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica.

Home Profile Dropdown

Home
Profile
Dropdown

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica.

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica.

Home
Profile
Dropdown

Navbars

Bootflat Link Dropdown

Bootflat Sign in as Mark Otto

Bootflat Sign in as Mark Otto

Bootflat Sign in

Bootflat Link Dropdown

Bootflat Sign in as Mark Otto

Bootflat Sign in as Mark Otto

Bootflat Sign in

Pills

Home 42 Home 42 Dropdown

Home 42

Profile

Message 20

Breadcrumbs

Home / Library / Date

Home Library Date

Home Library Date

Progresses

60%

60%

Paginations

PREV 1 2 3 4 5 NEXT

Pagers

PREVIOUS NEXT

Accordions

Collapsible Group Item #1

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #1

Collapsible Group Item #1

Collapsible Group Item #1

Collapsible Group Item #2

Collapsible Group Item #3

Lists

- Cras justo odio 14
- Dapibus ac facilisis in 91
- Morbi leo risus 38
- Porta ac consectetur ac 56
- Vestibulum at eros 20
- Dapibus ac facilisis in 99+

- Cras justo odio
- Dapibus ac facilisis in
- Morbi leo risus
- Porta ac consectetur ac
- Vestibulum at eros
- Dapibus ac facilisis in 38

List group item heading
Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.

List group item heading
Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.

List group item heading
Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.

Jumbotron

Implementing the HTML and CSS into your user interface project

HTML Structure

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

HTML Structure

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

Implementing the HTML and CSS into your user interface project

HTML Structure

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

HTML Structure

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

Thumbnails

Thumbnail label
Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id ...
Button Button

Thumbnail label
Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id ...
Button Button

Contact
Feel free to contact us for any issues you might have with our products.

Name Email
Name Name

Message
Cras mattis consectetur purus sit amet fermentum. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Morbi leo risus, porta ac consectetur ac

Close Send

Wells

Look, I'm in a well!

Panels

Panel title
Panel content

Panel title
Panel content

Panel title
Panel content

Panel title
Panel content

Panel title
Panel content

Panel title
Panel content

Panel heading

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

Panel heading

Some default panel content here. Nulla vitae elit libero, a pharetra augue. Aenean lacinia bibendum nulla sed consectetur. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Nullam id dolor id nibh ultricies vehicula ut id elit.

Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

Media heading

Media heading
12 Apr, 2013 at 12:00
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Nested media heading
12 Apr, 2013 at 12:00
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Nested media heading
12 Apr, 2013 at 12:00
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Nested media heading
12 Apr, 2013 at 12:00
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Media heading
12 Apr, 2013 at 12:00
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Jeffrey Zeldman
@zeldman

Follow

Content precedes design. Design in the absence of content is not design, it's decoration.

RETWEETS **1,893** LIKES **1,720**

1:48 PM - 5 May 2008

everything about the design has to support, communicate and reinforce the **meaning** of the content.

not the format, delivery method, length or volume.

the meaning.

IMAGE: Markus Spiske, Pexels, pexels.com

2 only include key content elements.

if you haven't seen or read the content, you don't fully understand what needs to be **communicated** via the prototype.

you cannot test or evaluate what you don't *understand*.

that understanding has to drive *every aspect* of design, from content to navigation to interaction.

IMAGE: UX Planet, *uxplanet.com*

2 only include key content elements.

the number one job of every single element on the screen is to:

1. allow **access** to content.
2. expose + explain its **relationship** to other content.
3. **present**, **communicate**, **support** + **reinforce** the meaning of that content so it's clearly understood.

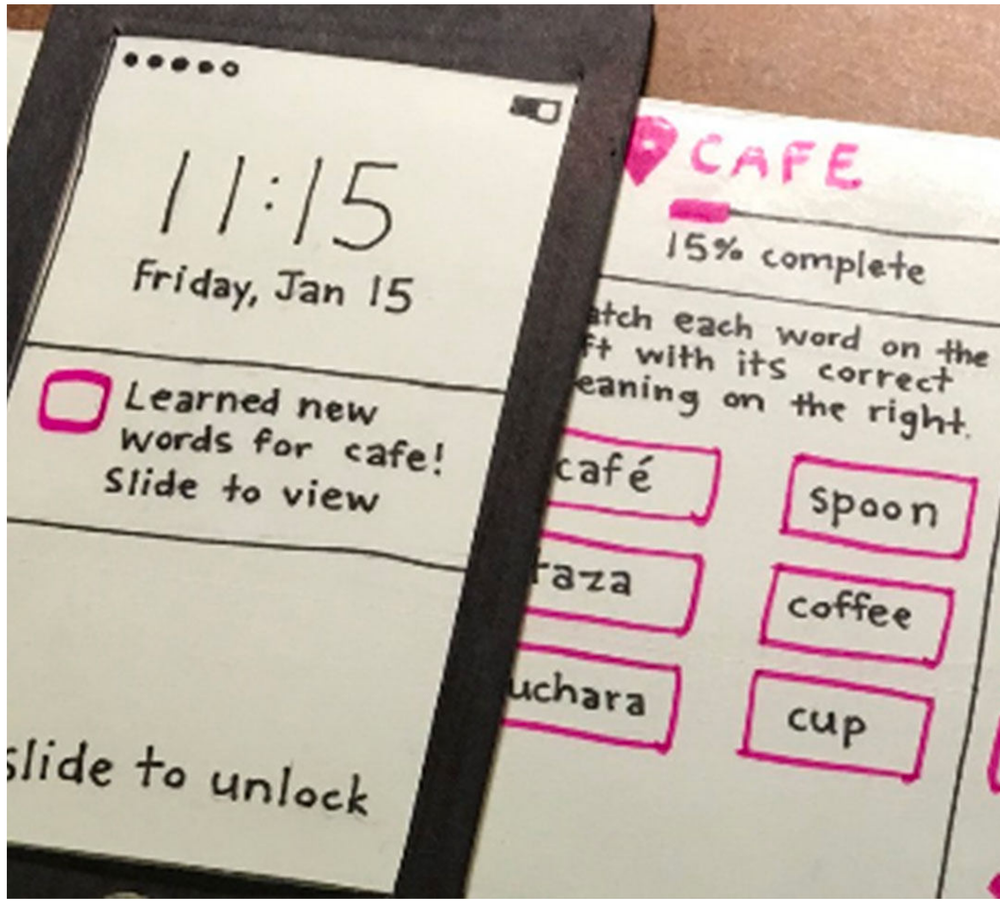


IMAGE: Ergomania, *ergomania.eu*

2 only include key content elements.

designing without that understanding isn't designing; it's *decorating*.

that doesn't help a user **parse**, **navigate**, **understand**, **anticipate** + **predict** the content she'll find.

so you want as much *real content* — labels, language, data, blocked out video placeholders etc. — in place as possible.

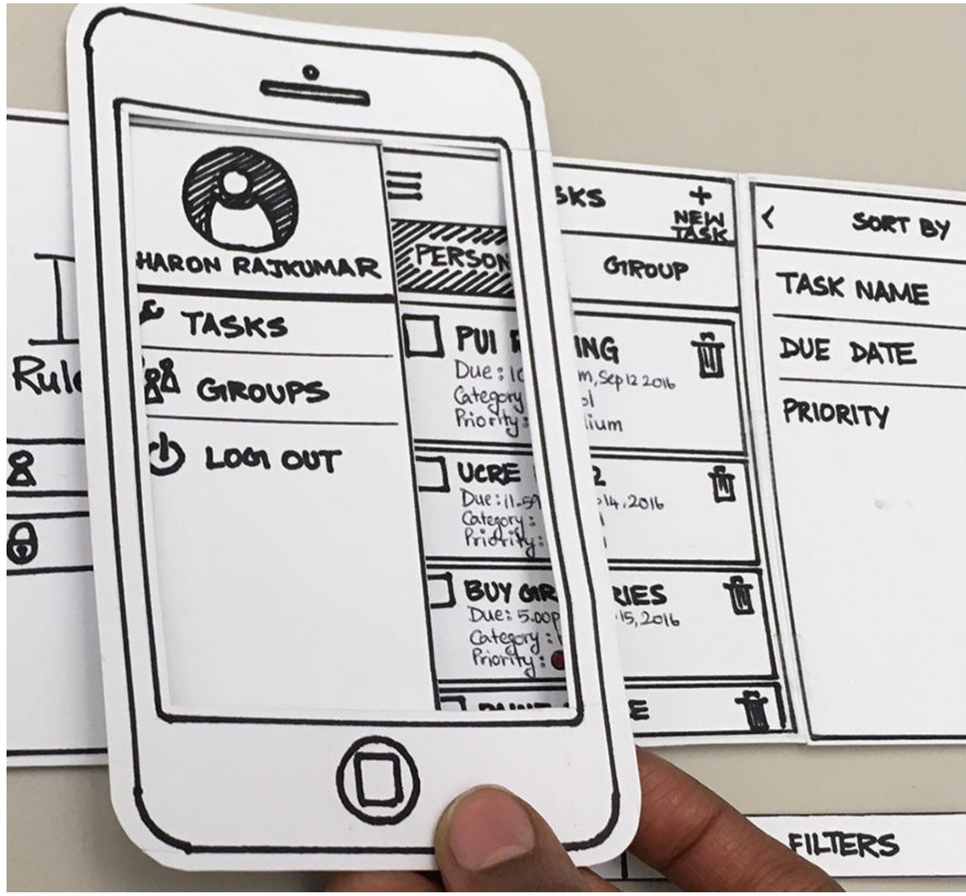


IMAGE: Sharon Rajkumar, Behance, *behance.net/gallery/43064215/Power-Paper-Prototyping*

3

interactivity is *simulated*.

users who test the low-fi prototype may be able to click, tap or swipe, but there's **no real production code** underneath those interactions.

with software tools, you *fake* this by creating a screen for **each state of interaction** and linking them together.

this interactivity can also be simulated manually with paper prototypes.

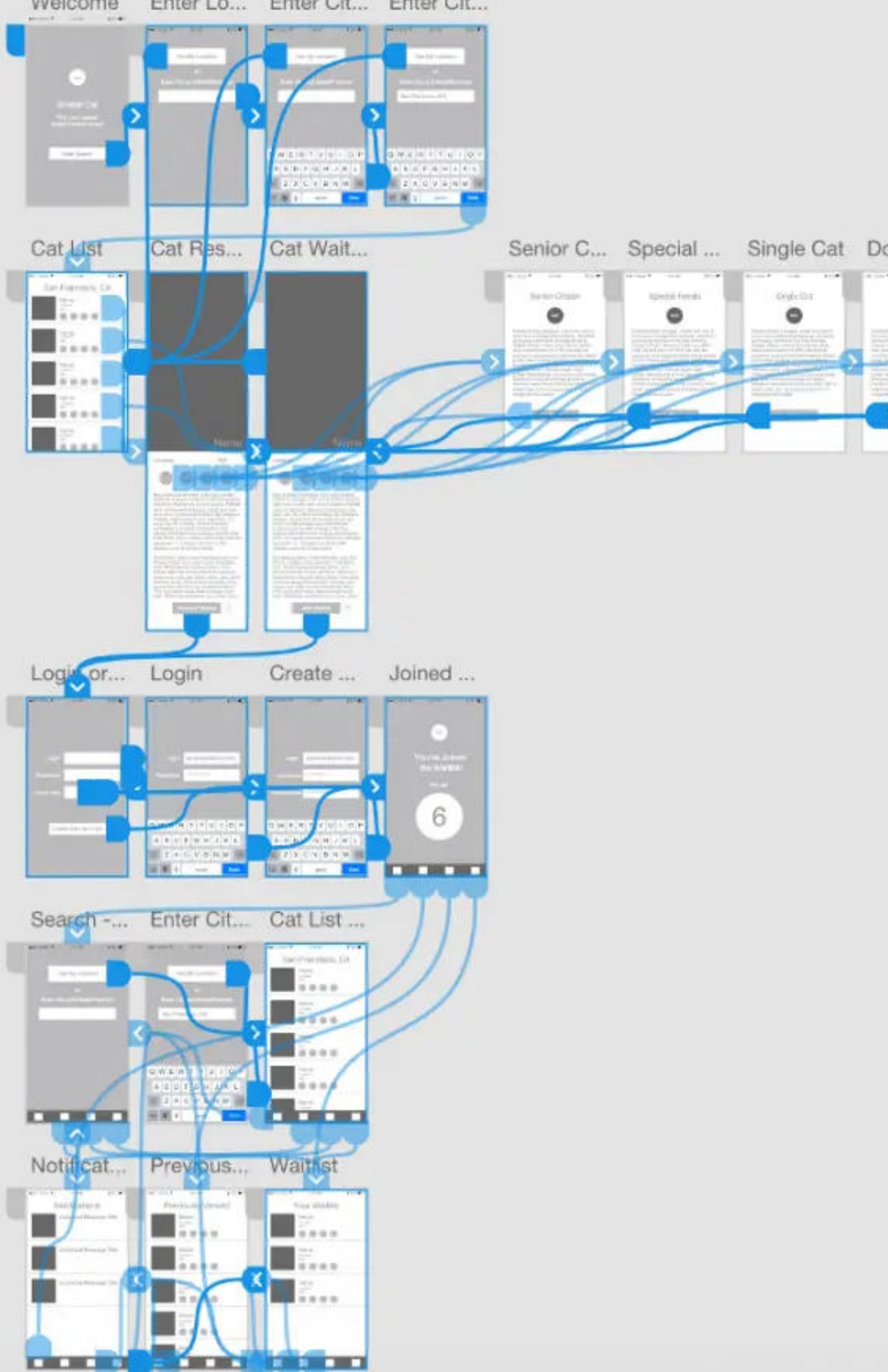


IMAGE: Adobe XD, *adobexd.com*

3

interactivity is *simulated*.

what's most important is that this lo-fi prototype allows us to ask + get answers:

- what **navigation categories** + **interaction mechanisms** are most appropriate?
- are we **sequencing information** properly so users can easily find what they need?
- are the **interactive cues** + **labels** used ones that users will readily *understand?*

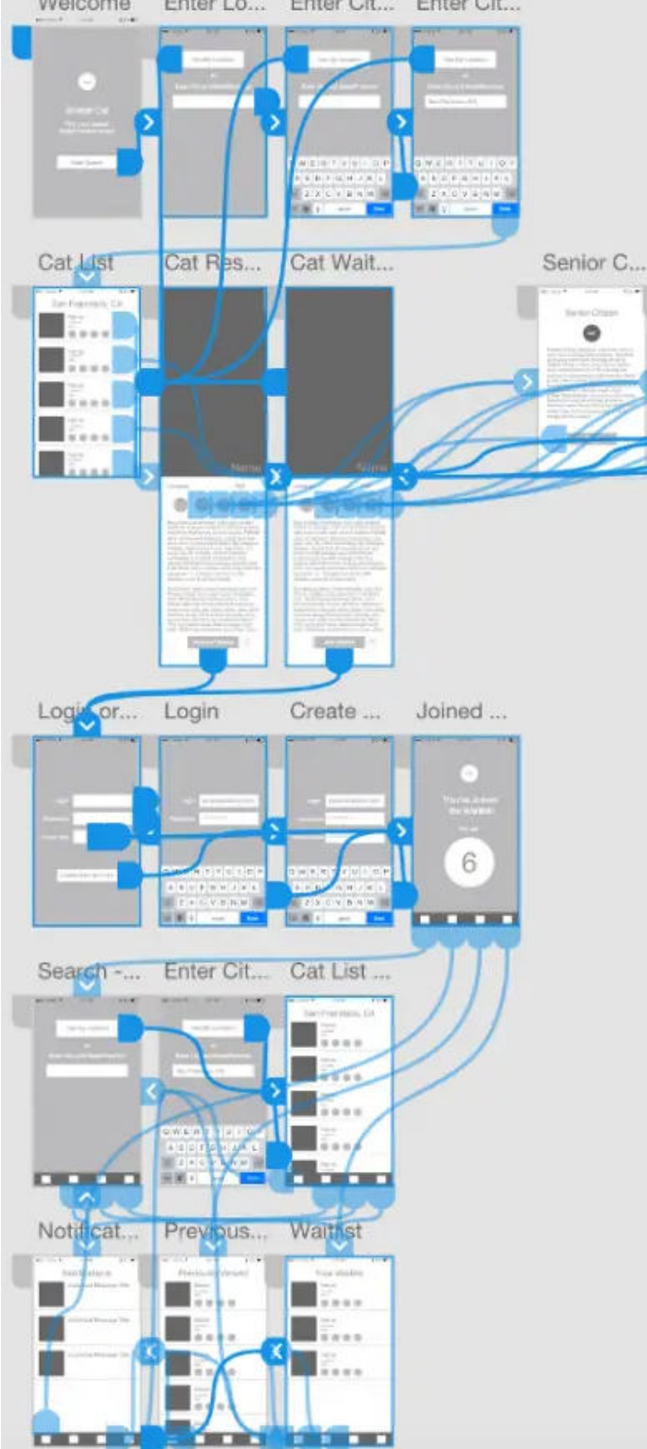


IMAGE: Adobe XD, *adobexd.com*

3

interactivity is *simulated*.

- does interactive behavior allow users to easily + accurately **predict the outcomes of their actions**?
- what about associated **labels + visual cues**, like buttons or menu or form elements?
- is the proposed interactive behavior + related functionality **appropriate to the user's mental model**?

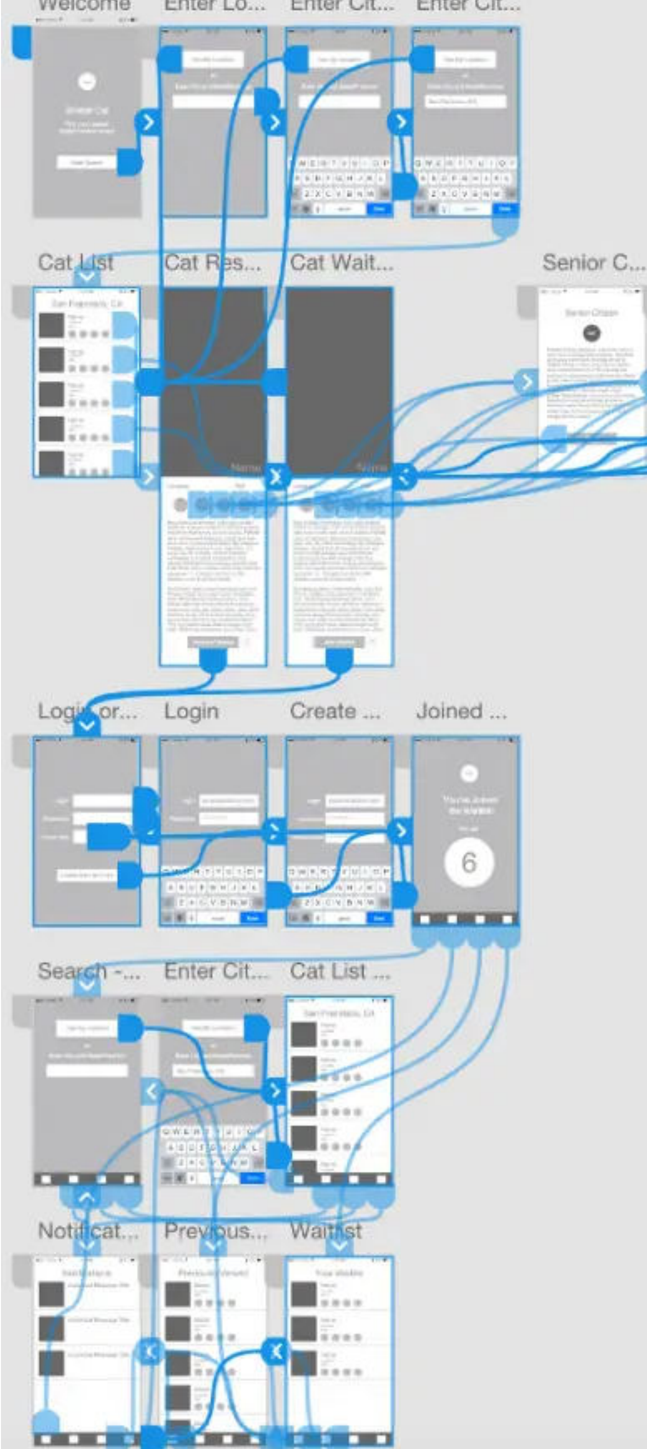


IMAGE: Adobe XD, *adobexd.com*

3

interactivity is *simulated*.

- **how is data surfaced + what can users do with it** once they have it?
- how can they **manipulate it** in terms of entry, retrieval, modification, addition, deletion, sorting, filtering, etc?
- are these kinds of interactions + data manipulation **feasible to build** with current constraints (time, budget, personnel)?

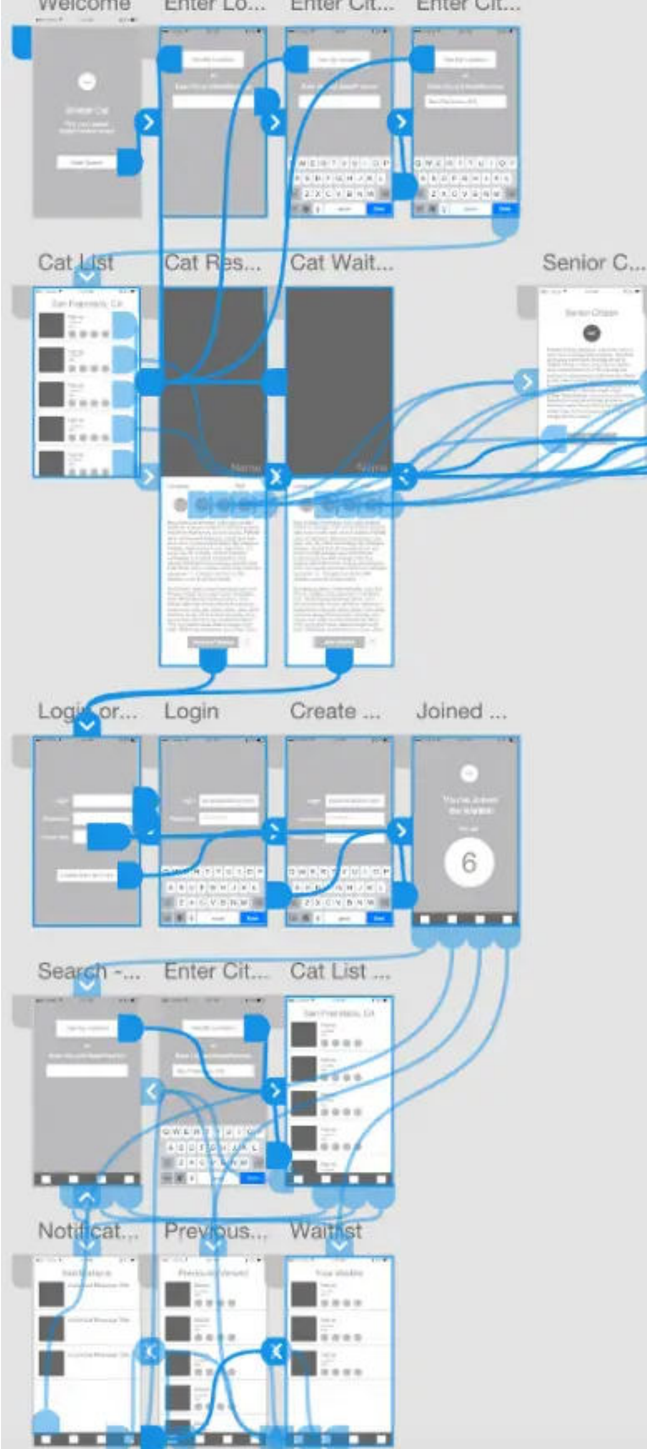


IMAGE: Adobe XD, *adobexd.com*

again, work to separate *information* from *action*.

you're building a model for that separation via design + interaction patterns; work quickly, implement one way of doing it, test it with users to find out:

- are these patterns what the user **needs** or **expects**?
- is each pattern **appropriate** for the **volume + type of content** they're manipulating or viewing?

IMAGE: ResearchGate, *researchgate.net*

actions always take priority in lo-fi.

the struggles that will make or break the product will be **struggles with interaction**, figuring out how to use something.

so data inputs and controls should be iterated + tested *before* their content-based counterparts.

the most important task: figure out what's needed, **when + how** the user needs it.

IMAGE: UX Planet, *uxplanet.com*

pros of **low-fidelity** prototyping

PRO

it’s inexpensive.

the deeper into a project you are, the more code that’s been created + committed, the more expensive it will be to fix mistakes.

once the team is working on a finished build, those changes become **extremely expensive** to implement.

IMAGE: Skitterphoto, Pexels

PRO

it’s inexpensive.

that expense is the team’s **time + effort** to first create and then rework and re-create what’s been done.

if we’re near the finish line, it also means **uncoupling + reworking or modifying** any number of third-party components or interdependent applications or systems or databases.

work multiples *exponentially*.

IMAGE: Skitterphoto, Pexels

PRO

it’s inexpensive.

from a business standpoint, the more time + effort spent on a project, the *less profitable* that project is.

so even if the goal is to improve an internal system to save the company money, those savings won’t happen if the project is **10X over budget**.

IMAGE: Skitterphoto, Pexels

PRO

it’s fast.

cost is also minimized because we’re not spending any time on **finished design**; just boxes, buttons, menus, text, hyperlinks.

all grey or black, all unstyled.

so changes to a prototype take a day at most — in some cases, only *minutes*.

IMAGE: Scott Webb, Pexels

PRO

it’s fast.

compare that to a week, month or several months’ worth of work, where the company’s paying 12 people to **change everything they’ve already built**.

the wasted effort, lost productivity + project cost are non-factors when we’re working in low fidelity.

IMAGE: Scott Webb, Pexels

PRO

it’s fast.

quick changes means getting the prototype **in front of users** quickly.

which means we **learn** more quickly.

which means we can change things quickly to **fix the places where they’re struggling** more quickly.

which means...you get the idea!

IMAGE: Scott Webb, Pexels

PRO

it’s a collaborative effort.

on any projects of any size, there are any number of different **stakeholders**: execs, subject-matter experts, product owners + managers who have influence + authority over what gets built and what doesn’t.

who are often *excluded* from design.

which makes them nervous, anxious + *combative*.

IMAGE: Blue Bird, Pexels

PRO

it’s a collaborative effort.

turn that **combat** into **collaboration**.

engaging them + involving them in the design process early on prevents all kinds of **surprise requests** later on down the road.

collaborating with them, getting input on the journey from idea to finished product, ensures you don’t miss anything mission-critical to them.

IMAGE: Blue Bird, Pexels

PRO

it’s a collaborative effort.

otherwise you’ll hear ***“I thought we were going to include A, B and C? we have to do that, we promised the customer,”*** at the point where everything is almost *done*.

having to now rework + include those things means a massive amount of rework, time, expense and *stress*.

who wants *that?*

IMAGE: Blue Bird, Pexels

PRO

it’s a collaborative effort.

low-fi prototypes make it much easier to include designers, developers, project managers, business analysts + users in the design, iteration + feedback loops of prototyping and testing.

they surface important conversations about **expectations + intentions**, keeping those conversations going throughout the life of the project.

IMAGE: Blue Bird, Pexels

PRO

it’s a collaborative effort.

designers don’t disappear for weeks only to come back with complex UI solutions that are unnecessarily difficult to code or *don’t really apply* to the problem at hand.

developers don’t go away + build a bunch of stuff that looks *nothing* like what design specified.

IMAGE: Blue Bird, Pexels

PRO

it’s a collaborative effort.

stakeholders have a concrete model that allows them to see exactly which features are included, which aren’t + how they’ll work.

they have ample opportunity to provide concrete, usable feedback based in reality, instead of how they imagine the thing looks or will work in their heads.

IMAGE: Blue Bird, Pexels

PRO

it’s a collaborative effort.

it also means that **stakeholders** won’t be frustrated + surprised after waiting 4 weeks into the project to finally see something that’s *not what they assumed it would be*.

everybody wins here.

BUT...

IMAGE: Blue Bird, Pexels

PRO

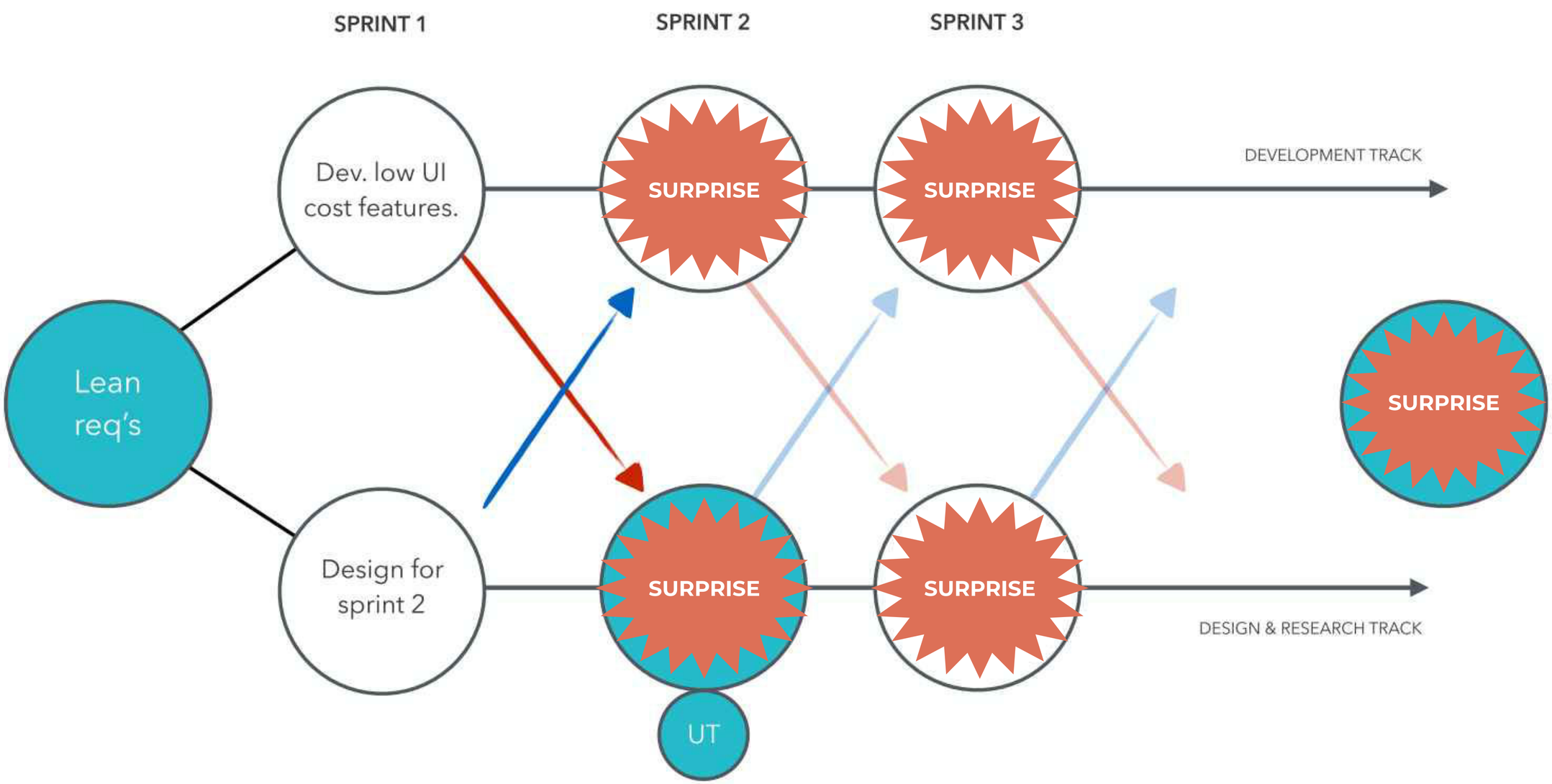
it’s a collaborative effort.

those wins only happen if there is true collaboration between UXers/designers + developers.

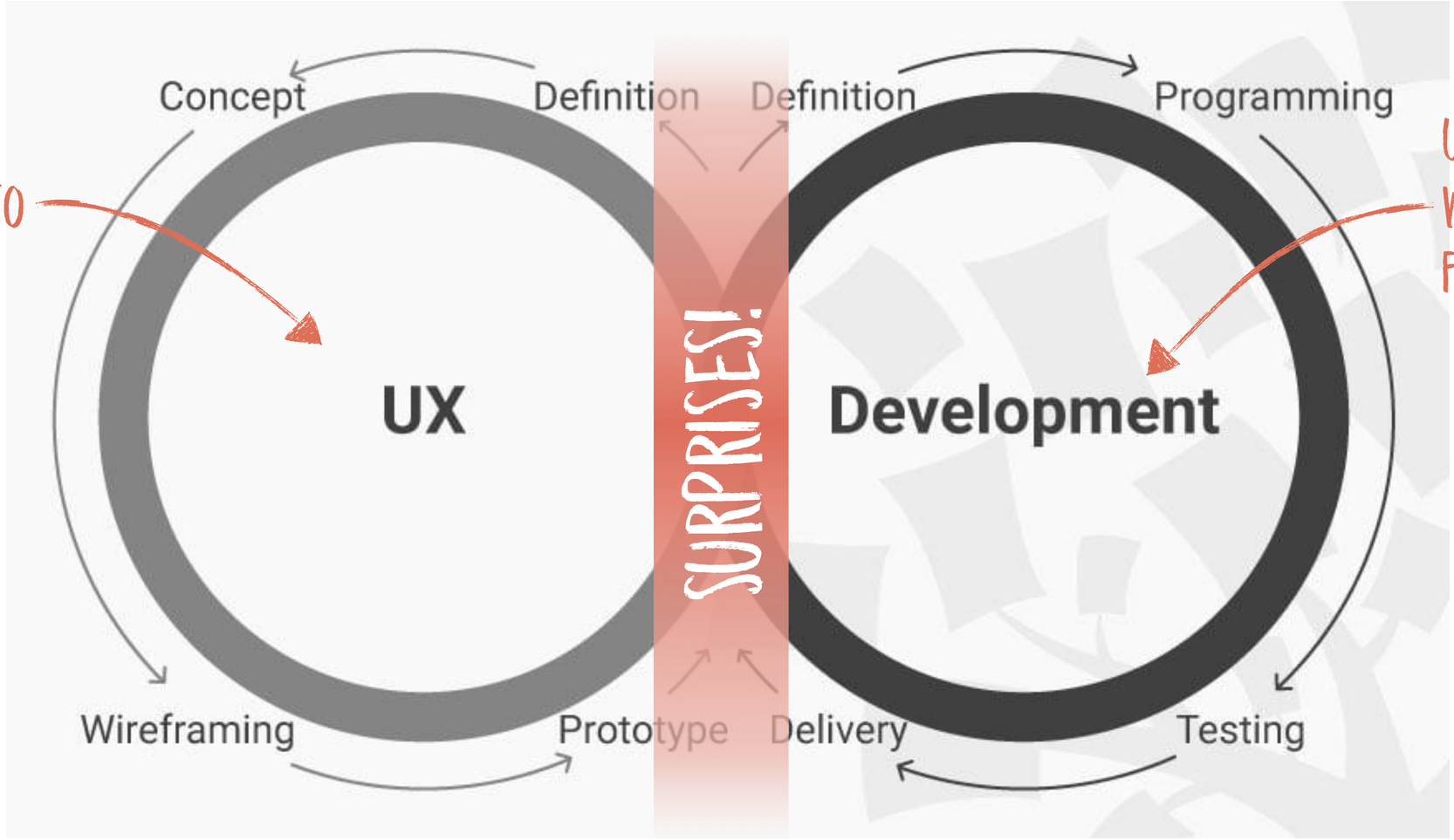
no handoffs.

which is the *opposite* of how most companies implement agile or lean product development.

IMAGE: Blue Bird, Pexels


SPRINT 1
SPRINT 2
SPRINT 3
Lean req's
Dev. low UI cost features.
Design for sprint 2
SURPRISE
SURPRISE
SURPRISE
SURPRISE
SURPRISE
SURPRISE
UT
DEVELOPMENT TRACK
DESIGN & RESEARCH TRACK

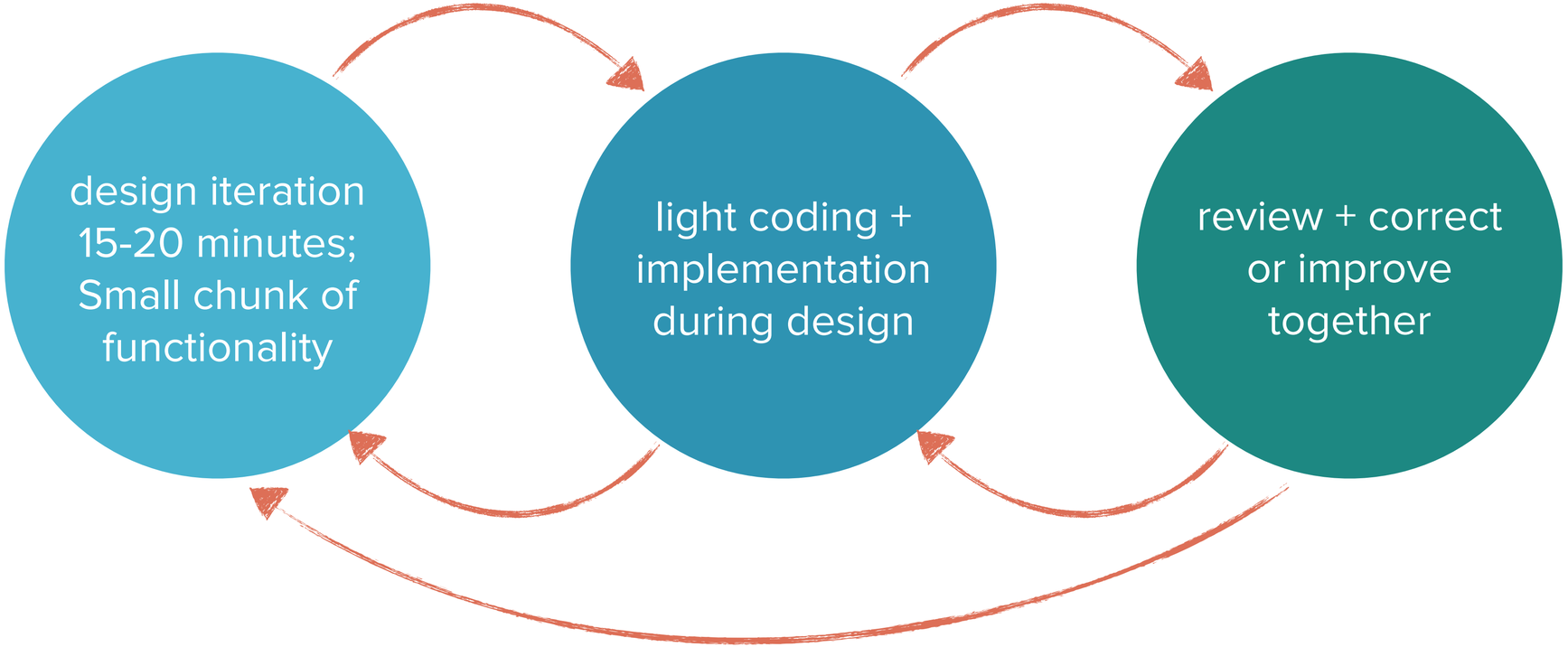
DEVS ARE WAITING FOR UX/DESIGNERS TO FINISH THEIR WORK



UX/DESIGNERS ARE WAITING FOR DEVS TO FINISH THEIR WORK

DESIGNER
OR UXER

DEVELOPER

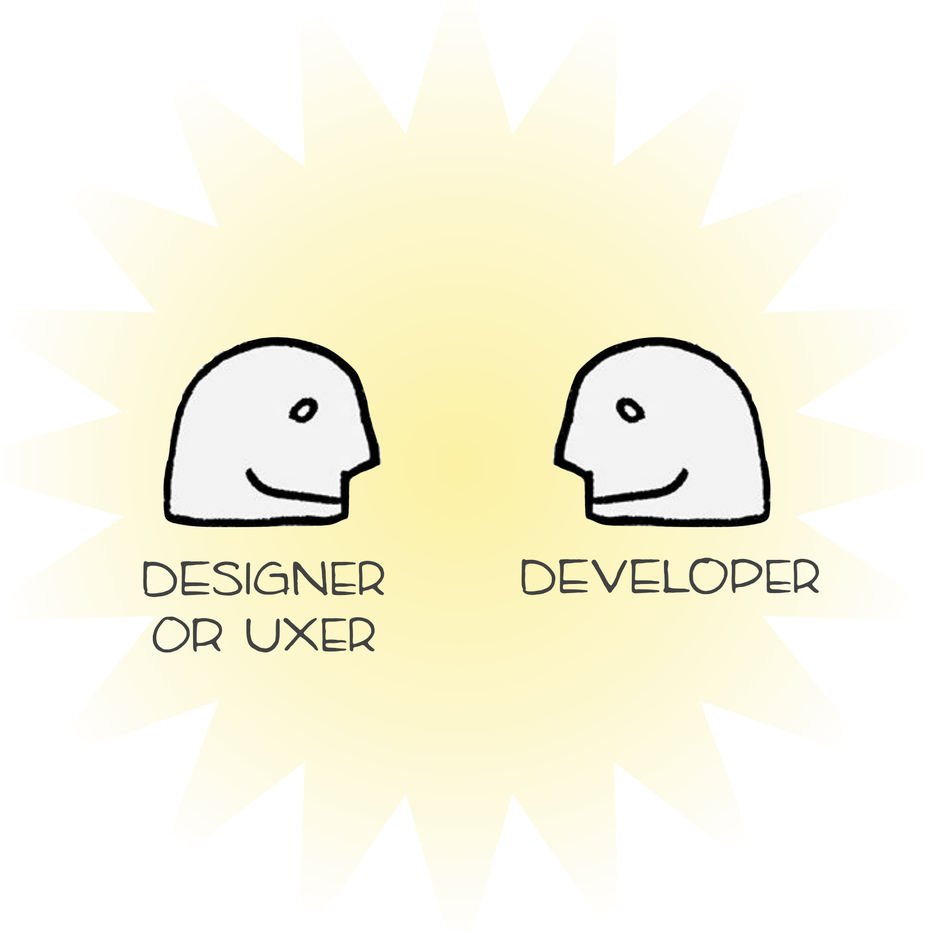


PRO

it’s a collaborative effort.

I’ve seen ONE ten-minute conversation replace **two weeks’ worth** of artifacts/ deliverables countless times.

two hours together in a room hacking through what’s possible + what isn’t can save you MONTHS in design + development work (and re-work).



PRO

it clarifies expectations.

both product team members + stakeholders will have a much clearer expectation about an upcoming project.

for all the reasons mentioned previously, because they can **see it** + **use it** + **experience it** for *themselves*.

IMAGE: fauxels, Pexels

cons of **low-fidelity** prototyping

CON

user confusion skews results.

users can get confused about what's supposed to work + what isn't.

so they may not engage with specific interactive components or interactive sequences.

which only indicates the clarity of the UI, instead of the **value** or **usefulness** of those skipped interactions.

CON

misplaced user focus.

test users will focus on the unfinished design instead of **what they're supposed to be doing**.

"why is everything grey? is it really going to look like this?"

this can make them slow in navigating the UI + trying things.

which gives us **false clues** about how *usable* the product actually is.

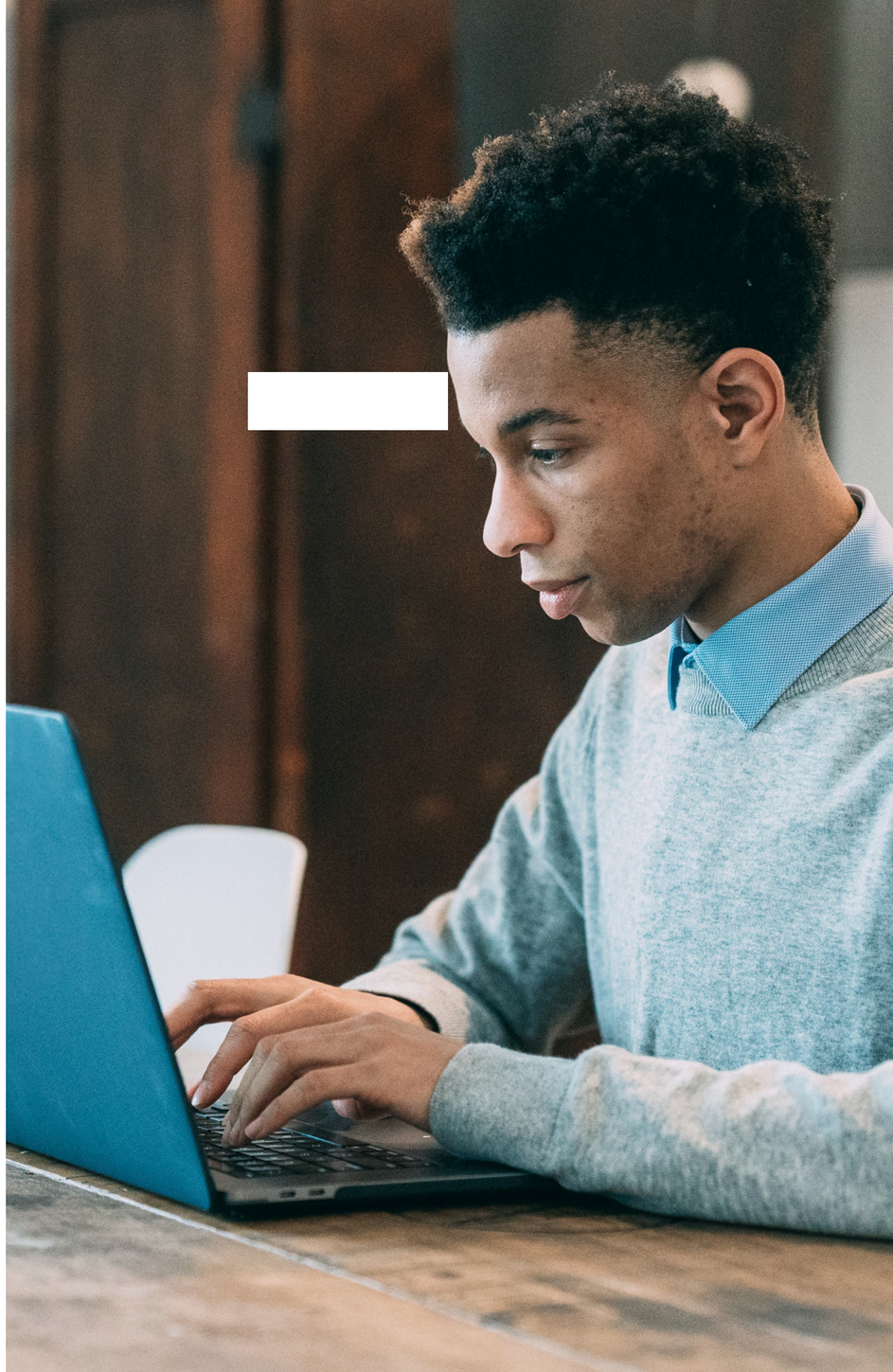


IMAGE: Greta Hoffman, Pexels

CON

limited interactivity.

the more complex an interaction sequence is, the harder it is to give a user **clear, obvious ways to manipulate that data** in low-fidelity.

so for complex systems + applications, it's very difficult to simulate *realistic* user scenarios.

which also means that there's a definite **limit to what can be tested**.

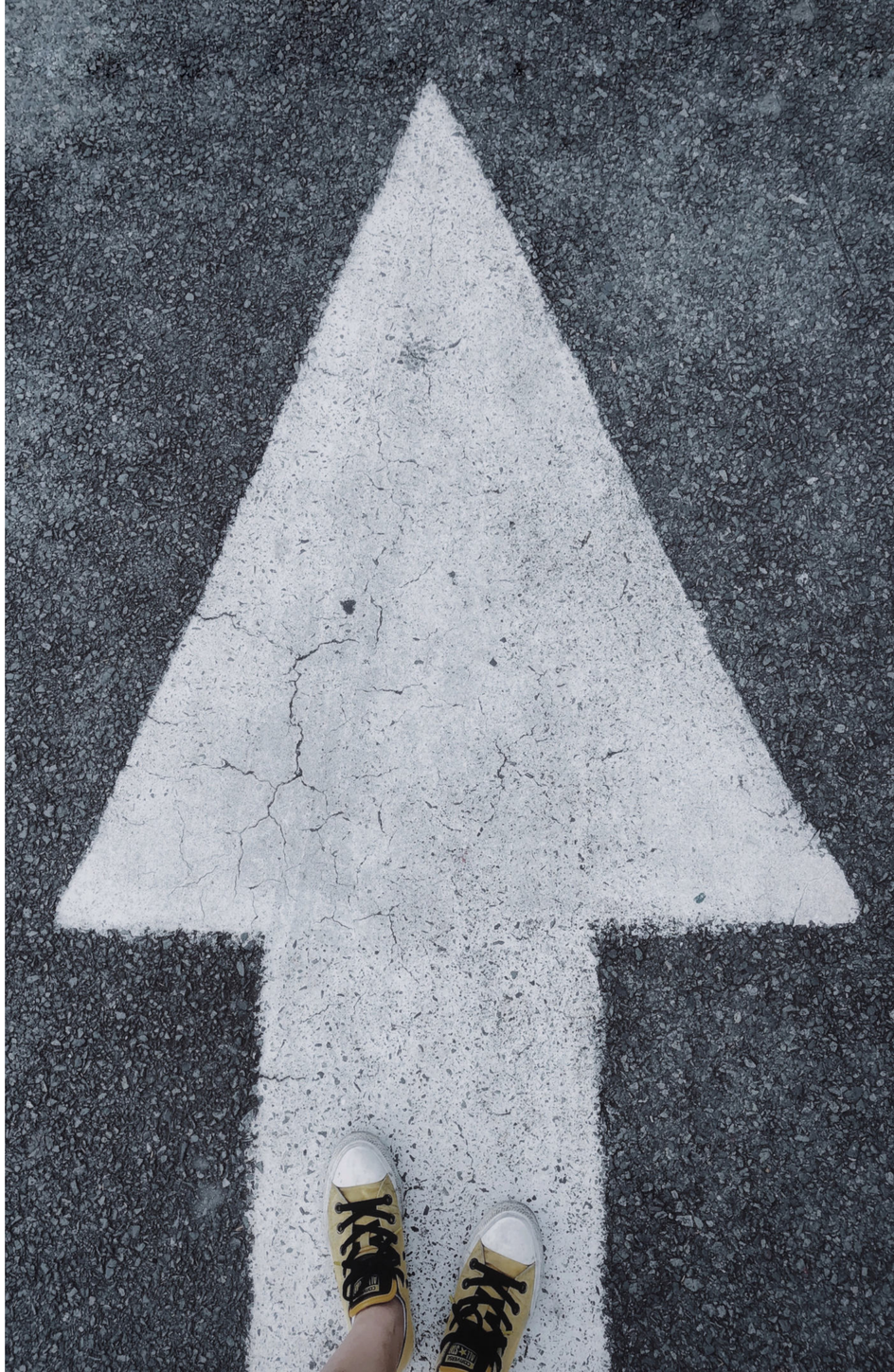


IMAGE: Vicky Tran, Pexels

CON

limits developer contribution.

a low-fi prototype is appropriate for testing what users expect or need...

but it doesn't convey much yet that's useful to your **development team**.

that lack of detail makes it hard for them to make **accurate judgments** about tech constraints or solutions.

so collaboration is *limited*.

IMAGE: Tom Leishman, Pexels

CON

low visual impact.

If you're **pitching an idea** to investors or to a new client you're trying to win, low-fi isn't the right choice.

the unpolished, unfinished nature of lo-fi prototypes makes them **less impressive** in these situations.

especially when you have prospects or stakeholders who are completely clueless about the design process.

IMAGE: Karina Zhukovskaya, Pexels

lo-fi is *still* the right place to start.

it's still much **easier + faster** to evolve a lo-fi prototype to the level of detail that solves all these issues...

than it is to slog through **complex technical requirements + religious debates** about what should or shouldn't be included, how it should be designed, etc.

high-fidelity prototyping

goal: simulate + test the real thing.

hi-fi prototypes are the closest you can get to a live, finished product.

we're still working with the same information architecture, content, screens + interaction sequences...

...but in much more complete level of interaction, UI detail + accuracy.

IMAGE: ThisIsEngineering, *Pexels; pexels.com*

goal: simulate + test the real thing.

we're trying to deliver a crystal clear picture of what the final product is intended to be.

how it's intended to work + how it should respond to user (inter)actions.

interactivity is all faked or simulated, but it's as close to the real thing as humanly possible.

IMAGE: ThisIsEngineering, *Pexels; pexels.com*

final quality inspection.

this is the point where any major issues have to be caught + addressed, so they don’t wind up in the final build.

otherwise, it’ll be a lot harder + a lot more expensive to correct anything we miss later on.

IMAGE: Erik Mclean, *Pexels, pexels.com*

1

comprehensive, detailed UI design.

this is a **pixel-perfect imitation** of the final product, often without real code.

that means interactive elements, icons, colors, gradients, shadows, real images, graphics + typography.

if it'll be in the finished app, site or system, it's included + simulated here.

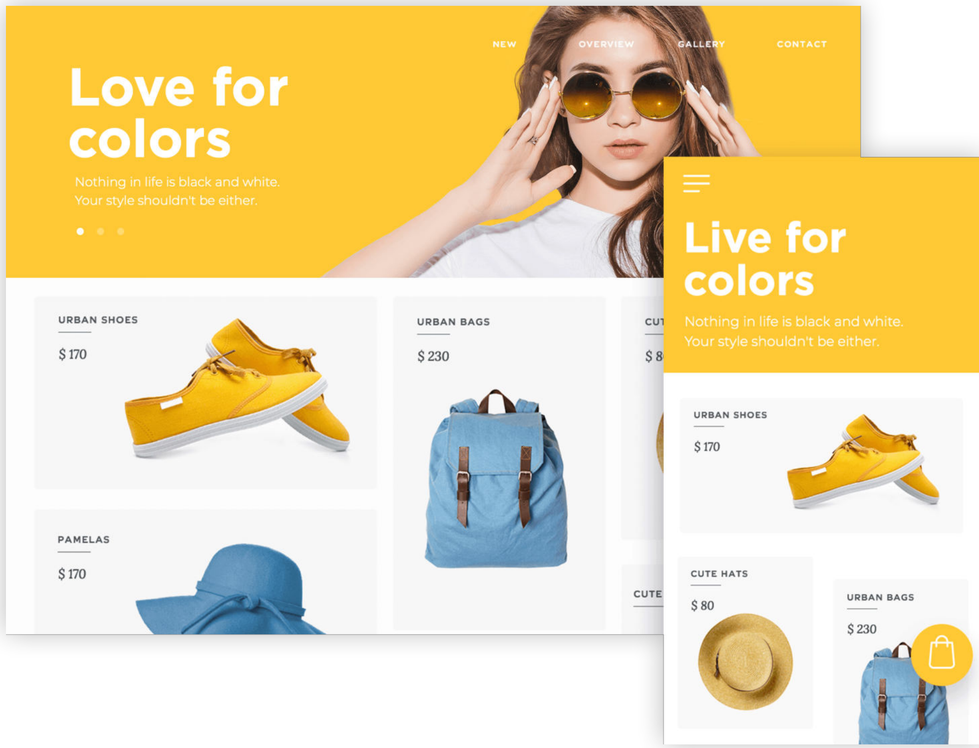


IMAGE: Justinmind, *justinmind.com*

2

advanced interactions + functionality.

interactive behaviors like scrolling, parallax scrolling, accordion menus, dropdown lists, form submission, drag + drop, etc. are all represented here.

a hi-fi prototype often uses **moving graphics + page/element animations**.

onscreen elements, from menus to buttons to forms, can be **manipulated** by the user.

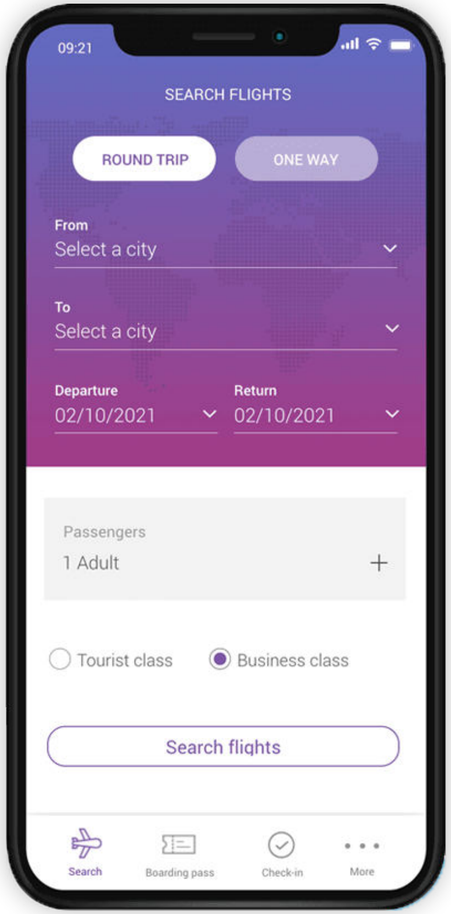


IMAGE: Justinmind, *justinmind.com*

2

advanced interactions + functionality.

we simulate error messages, inline form validation, as well as any kind of table, grid or list of items or search results that can be filtered + sorted.

in the case of **coded prototypes**, the ability to actually **enter, store + reproduce data** across various other screens is included.

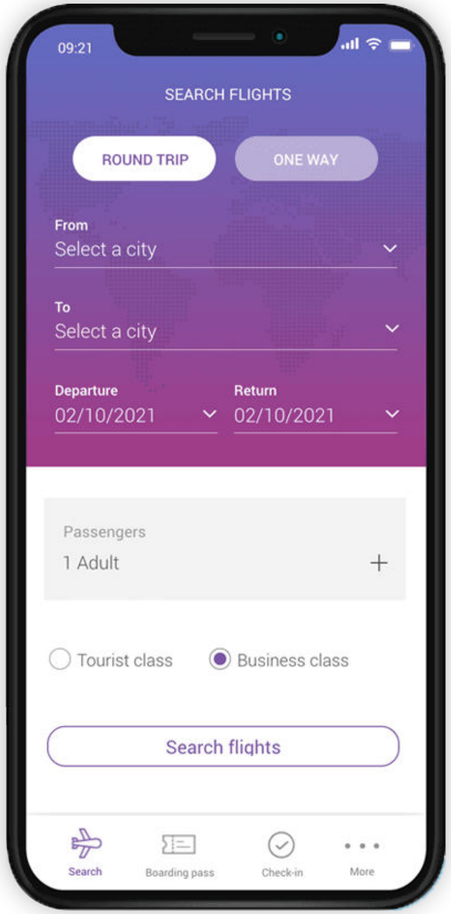


IMAGE: Justinmind, *justinmind.com*

QUESTION

why bother with a high-fidelity prototype?

why not just test and validate the low-fidelity version and *start coding?*

because the **bar**, the standard of *expected quality*, for any kind of digital product is *extraordinarily* high.

sales, adoption + use depend on great UX.

if you want people to try, buy or even use that product, it has to deliver what users need + expect.

immediately. during *first use.*

launch is the *worst time* to learn you've built something that people don't want, don't need or can't use.

IMAGE: Cottonbro, *Pexels, pexels.com*

In-house app UX is just as critical.

even in situations of compelled use, like in-house systems for employees, solid UX upon launch is critical.

because if not, people *don't use* what's in front of them.

instead, they find ways to **work around** the parts of the system they find difficult, confusing or frustrating.

IMAGE: Michael Burrows, *Pexels, pexels.com*

there’s a cost to this.

people + the organization both lose:

- efficiency + productivity
- cost savings or profit
- lost time dealing with duplicated, incorrect or contradictory work
- work that can’t be tracked because it’s happening outside the system

IMAGE: Karolina Grabowska, *Pexels, pexels.com*

everyone knows what's up.

a hi-fi prototype makes sure everyone has a view into what people **can + are willing to use**.

into the **outcomes** of that use + how it impacts users' or customers' ability to succeed, to get what they *want*.

to do what the **organization needs them to be able to do**.

IMAGE: RODNAE Productions, *Pexels, pexels.com*

there are no

SHORTCUTS.

if you want to avoid the pain + consequences of bad UX in any kind of finished product...

hi-fi prototyping is a *requirement*.

when do we move to hi-fidelity?

1

when basic functionality is established, tested + *done*.

it's time to move on to more detail:

- once you've got the **basic interactive elements, screens + interaction sequences** needed to satisfy core user goals.
- once you've **tested, improved + validated** that those elements are the *right* choices.

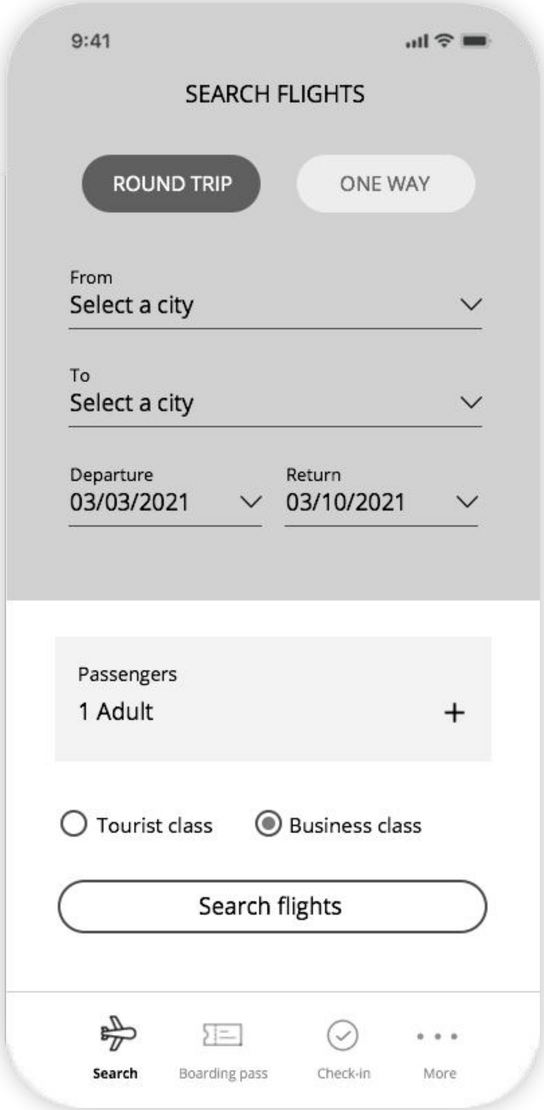


IMAGE: Justinmind, *justinmind.com*

1

when basic functionality is established, tested + *done*.

- you know which screens from your lo-fi prototype are **mission-critical** to user + product success.
- you have a **rough visual map + structure of the UI** of each screen.
- you understand **how people move from one screen to another**, or one interaction state to another.

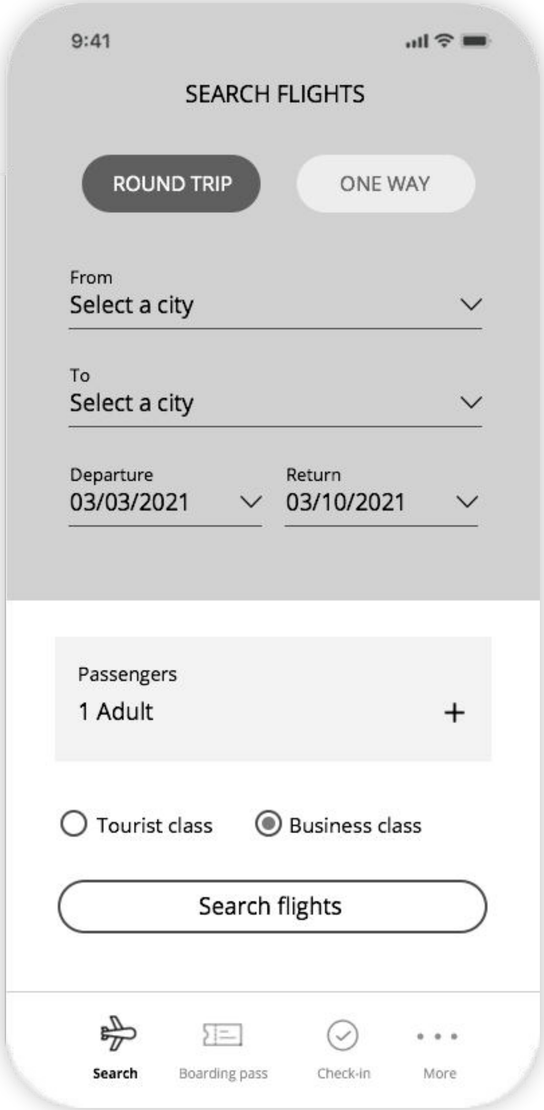


IMAGE: Justinmind, *justinmind.com*

1

when basic functionality is established, tested + *done*.

now it’s time to start working out the details of how those elements **look, behave + function**.

in collaboration with developers.

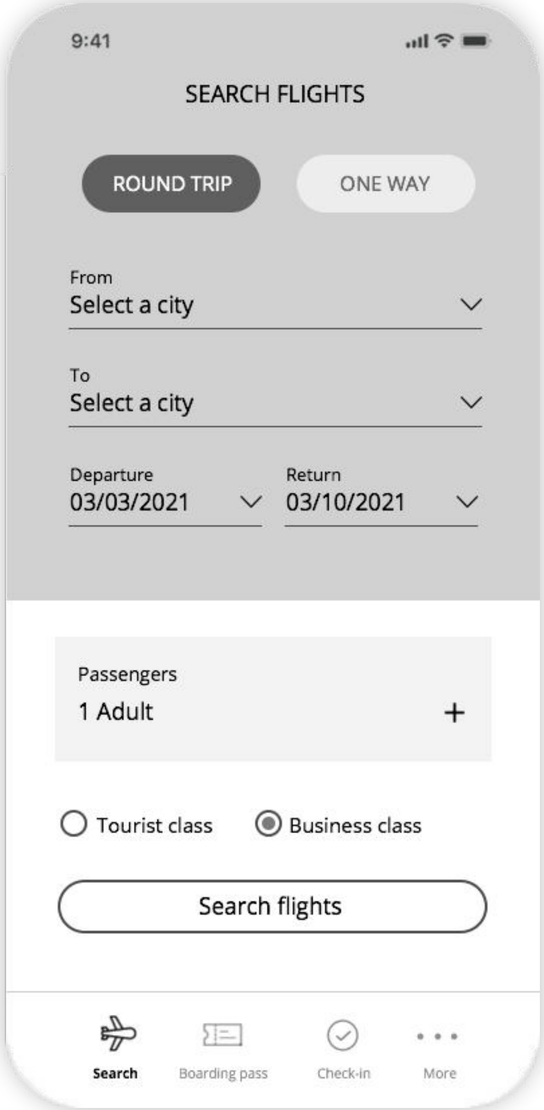


IMAGE: Justinmind, *justinmind.com*

2

when you need to fully design + test advanced interactions.

when it's time to determine whether complex interactions + functionality like data visualization, filtering + sorting results or views, etc. are **appropriate, understandable + usable**.

in order to **test** them accurately, you need to get as close as possible to how they'll actually look + work.

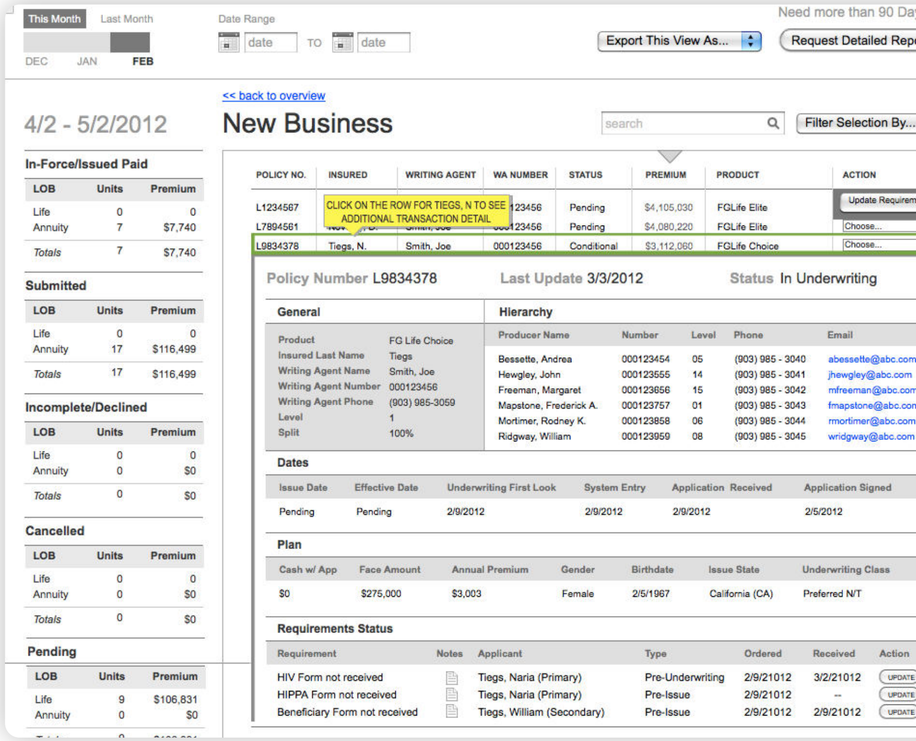


IMAGE: Marvel, *marvelapp.com*

2

when you need to fully design + test advanced interactions.

otherwise you'll get **false clues** that will lead you, your team members, stakeholders or bosses to believe everything will work as intended.

users *will* get what they need, customers *will* line up to buy.

without high-fidelity prototyping, you'll be wrong about *all of that*.

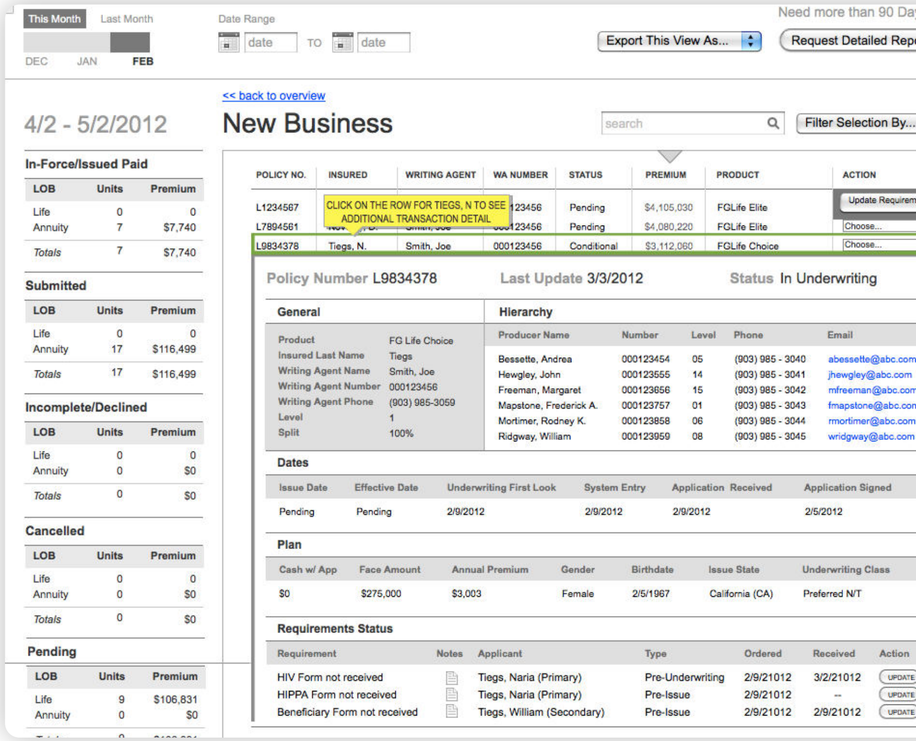


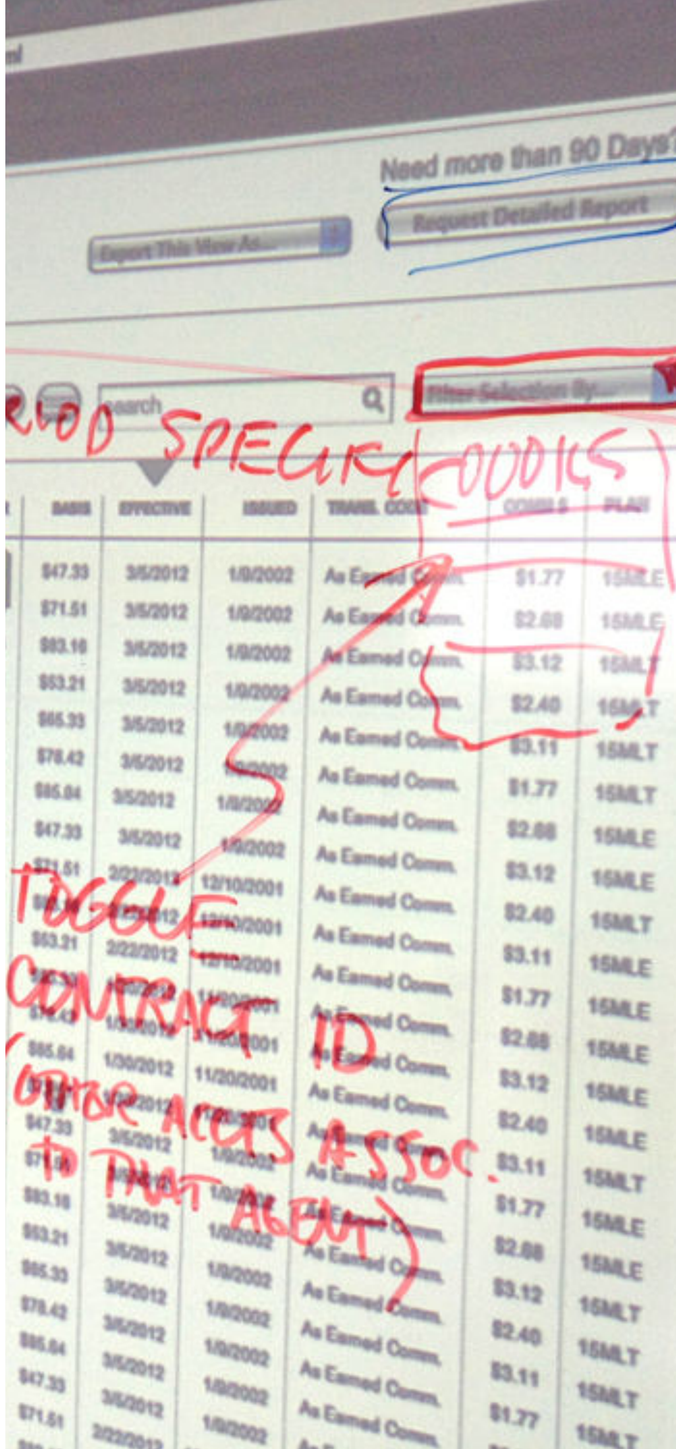
IMAGE: Marvel, *marvelapp.com*

3

when collaborating on complex functionality with developers.

when it looks and acts like the real thing, developers can quickly identify **what's possible**, along with the **level of effort** required to pull it off.

which means you both have **answers** for project managers hounding you about how long this will take!

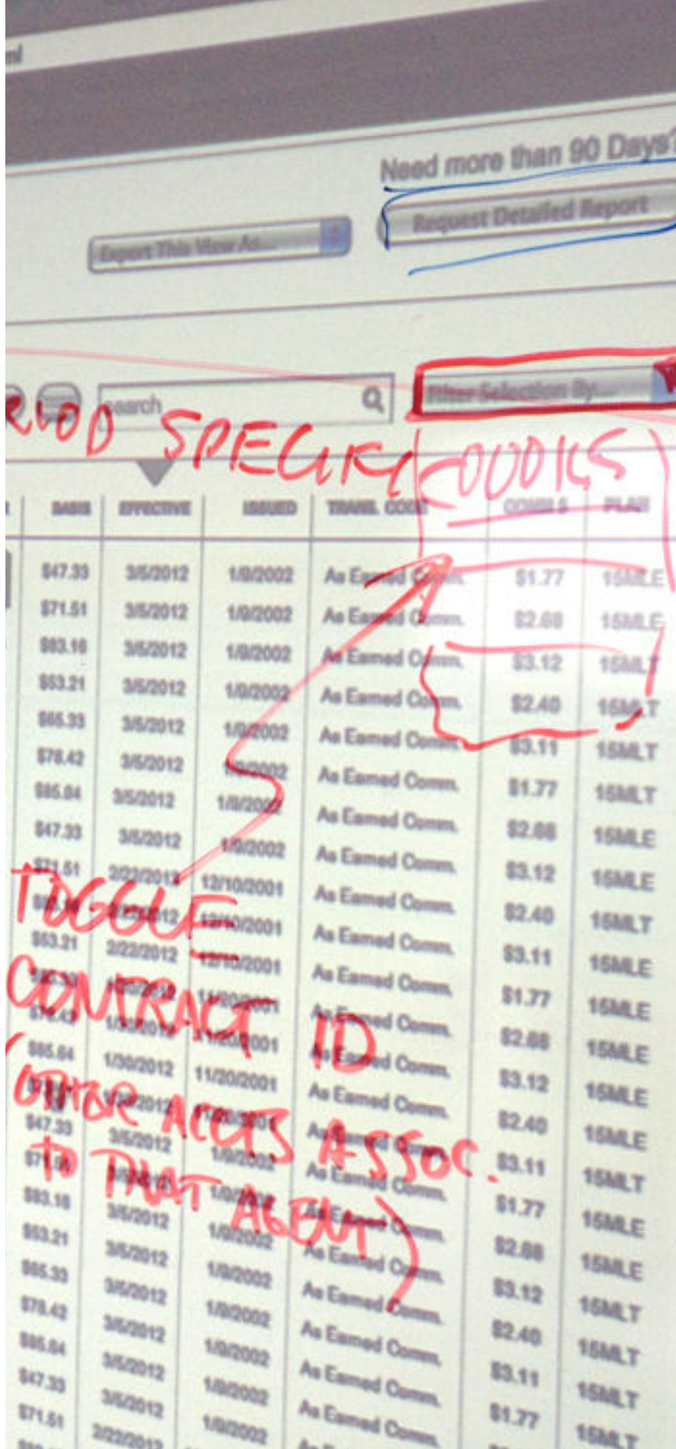


3

when collaborating on complex functionality with developers.

co-designing keeps both of you from decisions that make the other's work more difficult than it needs to be.

you'll both understand what the **functional requirements** are in terms of front-end interaction, required logic, database and backend processes.

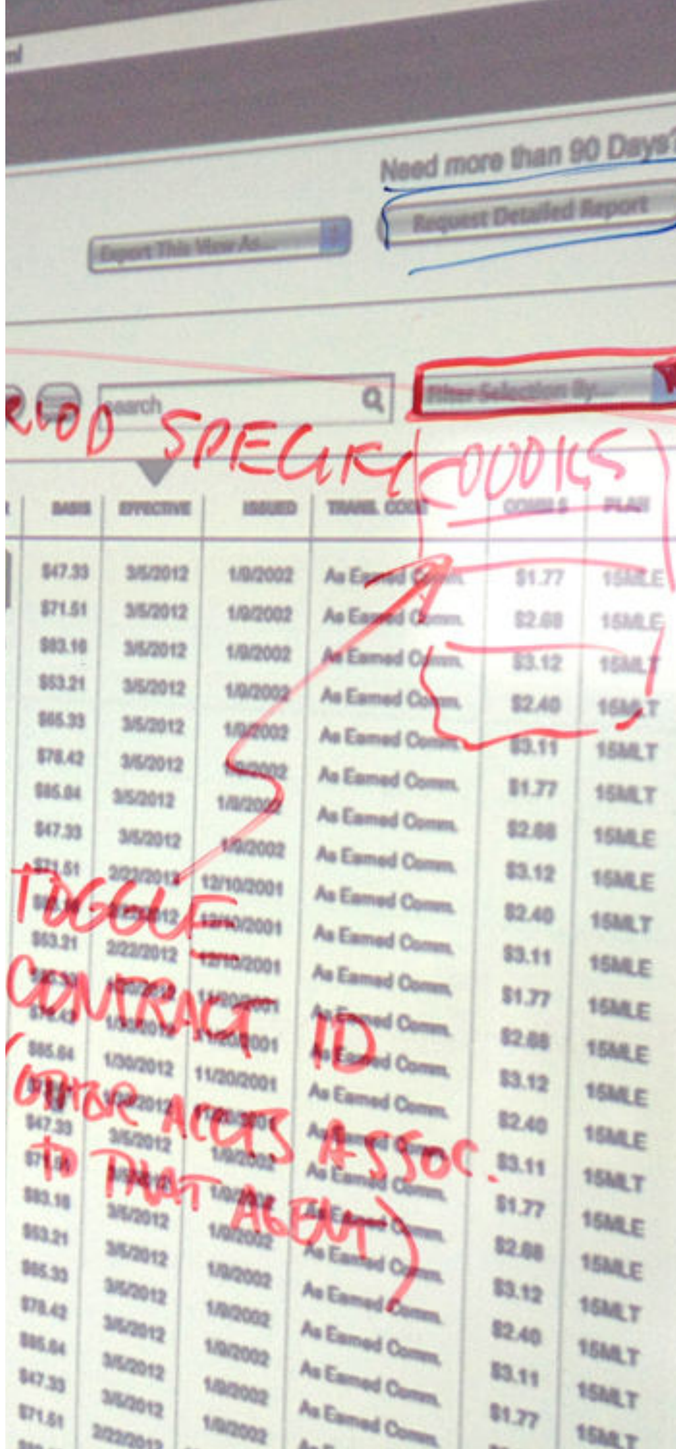


3

when collaborating on complex functionality with developers.

you'll be working out crucial UI design assets together, so:

- you're not designing things they can't build + mimic.
- they're not building things that don't deliver the intuitiveness or UX you know is necessary.



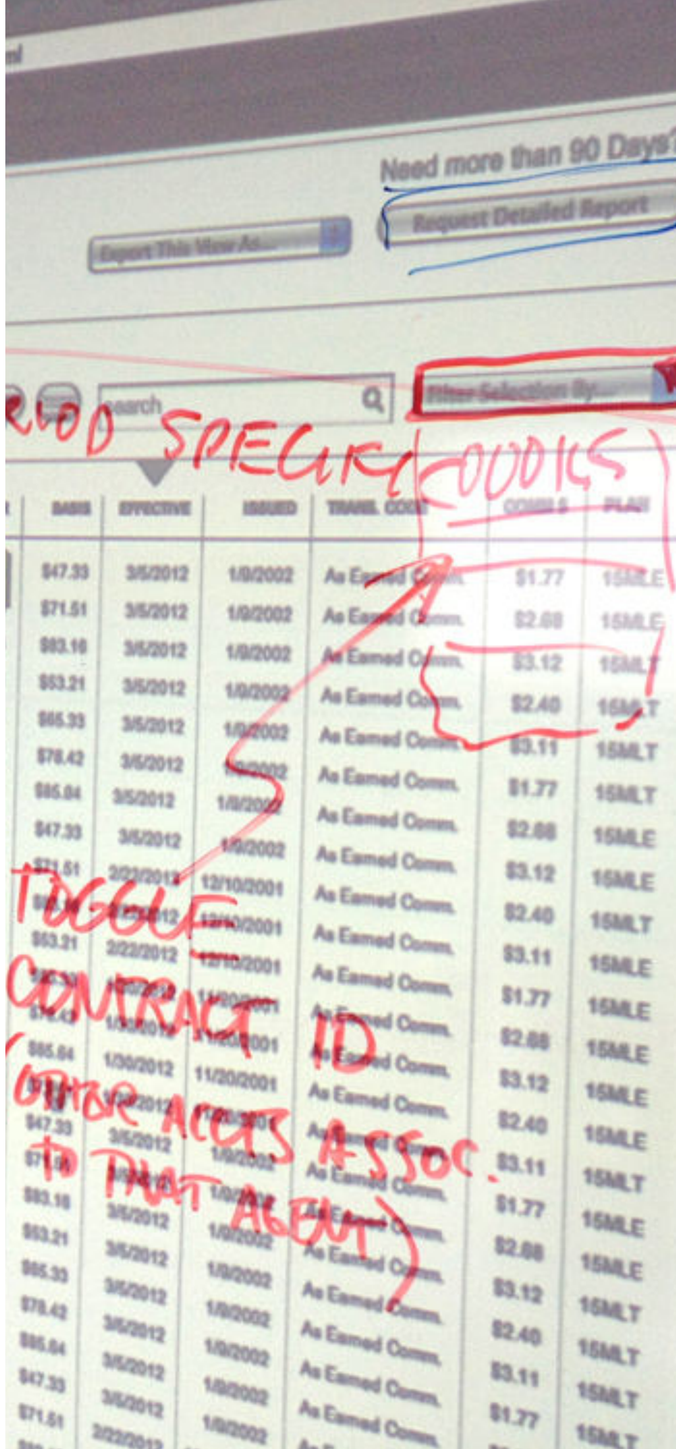
3

when collaborating on complex functionality with developers.

this **reduces time spent designing + coding**; launch happens sooner.

it **reduces error** + drastically cuts down + can eliminate **rework**.

which makes your managers + executives *very* happy.



PUCRS online uol edtech