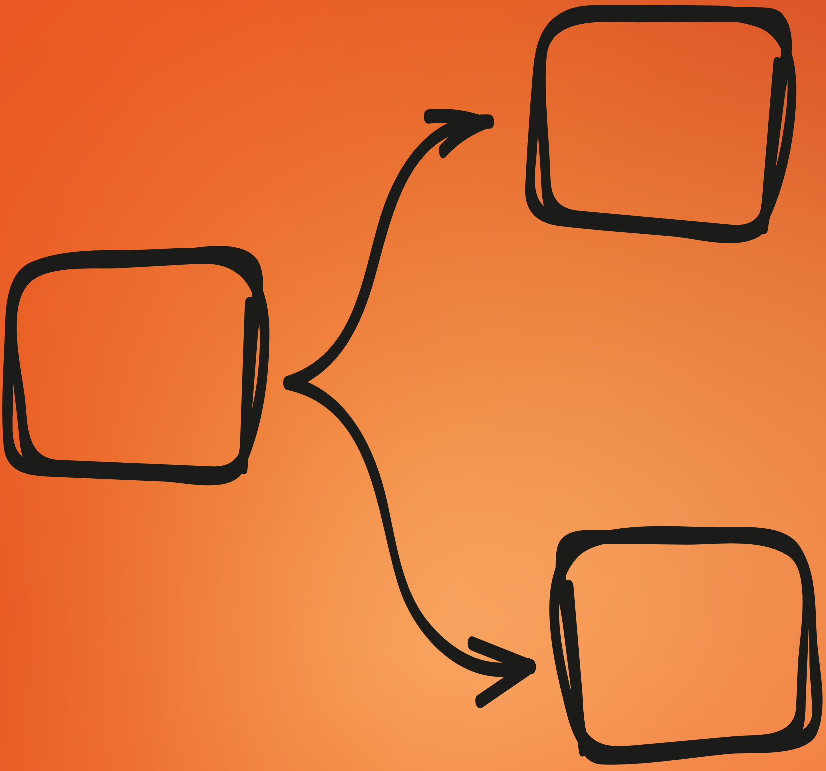


NORMAL UI

How Non-Designers Can Make Their Web Apps Easier To Use

Tony Alicea

Introduction

How do you make a web application easier to use? Note that I said “easier”, not “easy”.

Making a web application “*easy*-to-use” involves multidisciplinary expertise in areas like usability and web performance. Making a web application “*easier* to use” just means doing something that *improves* the experience.

That’s what this book teaches - **a repeatable technique** you can use to make any web application easier to use.

If you’re a developer who builds web applications (e-commerce, SaaS, intranet apps, or anything else you’re building), this book is for you. Designers who know more about making web apps look good, but less about making them work well, will also benefit.

Your web apps need to be easier to use. Usable software is successful software.

Where did this technique come from? Well, I’ve had an interesting career. For decades, I’ve been a developer and database architect. I’ve built many web applications that run large businesses. I’ve taught over 350,000 students to build web apps in my online courses.

I’ve also spent much of that time as a user experience designer. I’ve built design systems, run countless usability tests, designed interfaces used by thousands, and trained others in cognitive science, usability, and more.

Working heavily in both code and UX is considered rare in the industry. It’s given me, I think, a unique perspective.

This book is a result of my years of straddling the dev/UX fence. After 25 years of both building software and watching people use that software (and software others designed and built) I can say with confidence that using this technique will make your software easier to use.

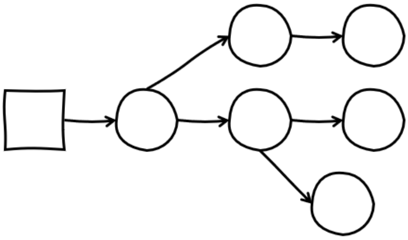
That doesn't mean I'm not standing on the shoulders of others. This technique sits on top of the massive body of work of usability engineers, cognitive scientists, and more.

I've refined and used this approach for decades, resulting in software that usability tested fantastically well, is easy for devs to implement, attracted more and happier users, and satisfied clients and stakeholders.

You don't need to be a designer to use these ideas. The technique is simple enough to be applied to any web application.

I call it **Normal UI**.

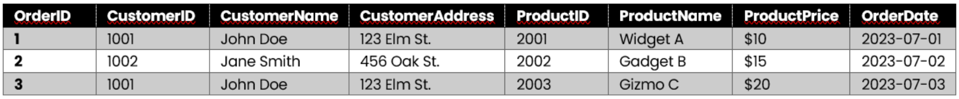
The Technique



Normalization

When I use the word “normal”, I’m referring to the term as it’s used in databases. A quick understanding of that will help you understand Normal UI.

In a “relational” database you might have a table (also called a “relation”) that looks like this:



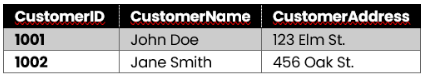
OrderID	CustomerID	CustomerName	CustomerAddress	ProductID	ProductName	ProductPrice	OrderDate
1	1001	John Doe	123 Elm St.	2001	Widget A	$10	2023-07-01
2	1002	Jane Smith	456 Oak St.	2002	Gadget B	$15	2023-07-02
3	1001	John Doe	123 Elm St.	2003	Gizmo C	$20	2023-07-03

A denormalized table, in all its glory

This table is “denormalized”. Each row contains a lot of information about the customer, the product they ordered, and the order itself. Each row of the table is doing a lot of work.

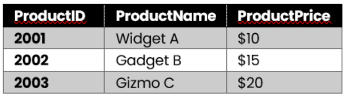
This kind of unfocused table results in data (like the customer’s name and address) being repeated, which means it’s more work when the customer moves and needs to update their personal information.

When you normalize a table like this, you split the information into multiple tables that might look like this:



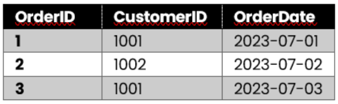
CustomerID	CustomerName	CustomerAddress
1001	John Doe	123 Elm St.
1002	Jane Smith	456 Oak St.

Customers



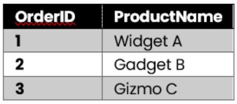
ProductID	ProductName	ProductPrice
2001	Widget A	$10
2002	Gadget B	$15
2003	Gizmo C	$20

Products



OrderID	CustomerID	OrderDate
1	1001	2023-07-01
2	1002	2023-07-02
3	1001	2023-07-03

Orders



OrderID	ProductName
1	Widget A
2	Gadget B
3	Gizmo C

Order Details

Each table focuses on one type of thing (Customers, Products, Orders, Order Details). Information is only stored in one place and referenced as needed.

When it comes to avoiding duplication and keeping each table easy to reason on, you want normalized data. But there are times (like doing a task that requires seeing all the data), when you need to pull it all together, and use a denormalized table.

In database land that's usually done via something called a SQL query, which provides a way to temporarily combine multiple tables into a single denormalized table. But that's beyond this book.

Database normalization provides the inspiration for Normal UI. In Normal UI, we think of workflows and UI screens as normalized or denormalized. For the context of this book, then, let's specify a definition of "to normalize":

> **Normalize *(verb)*:**
> 1. *In database design*: to organize data so that each table is focused on a single topic or theme, thereby eliminating redundancy and ensuring data integrity.
>
> 2. *In user interface (UI) design*: to split software workflows across different screens, so that each screen is focused on a particular task, simplifying the user experience and minimizing confusion.

For the rest of this book, we'll be focused on definition #2. How do you normalize an interface? When do you want a denormalized one? How will that make your web application easier to use?

Workflows, Screens, and Frames

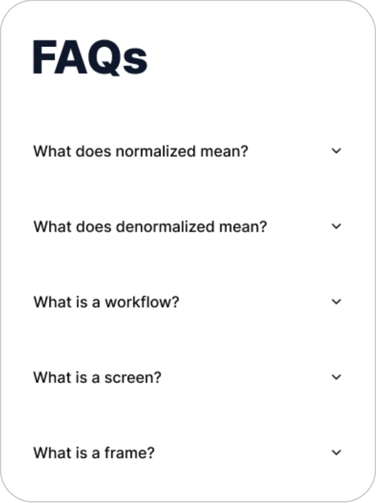
If you design or build software, you may be used to thinking in terms of "features". But that isn't helpful here. Instead, we are going to think in terms of "workflows", "screens", and "frames". In Normal UI we define these terms as follows:

A screen is *a page or a modal* in a web application.

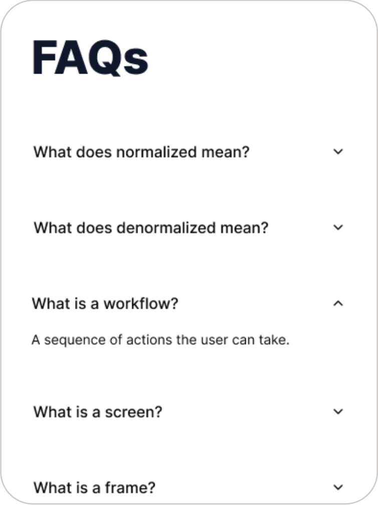
A frame is the set of UI elements that can all be accessed simultaneously by the user at a point in time. It's *everything the user can interact with, at a single moment, without having to reveal it in some way*. It represents a subset of the screen's potential content.

A workflow is *a sequence of actions that the user can take*. An action may or may not take you to a new screen but will almost always take you to a new frame.

A simple example is a set of accordions. We have a screen on its starting frame.



Let’s say a possible workflow is to click on an accordion to reveal its contents, then click again to hide its contents. Each click is an action in the workflow, and results in a new frame.



FAQs

What does normalized mean?

What does denormalized mean?

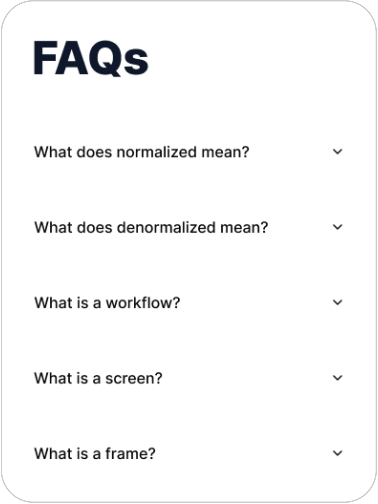
What is a workflow?

A sequence of actions the user can take.

What is a screen?

What is a frame?

Open



FAQs

What does normalized mean?

What does denormalized mean?

What is a workflow?

What is a screen?

What is a frame?

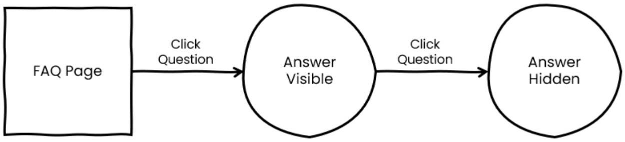
Closed again

Remember I said you didn’t need to be a designer to use Normal UI? All you need to be able to do is draw basic squares, circles, and arrows.

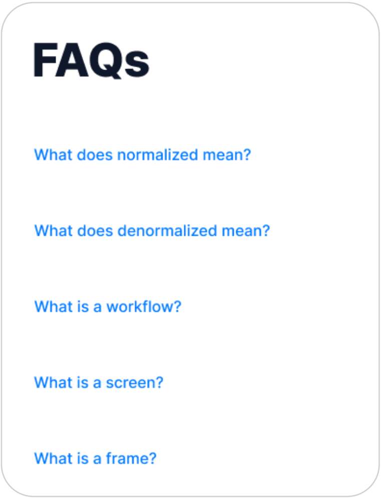
What we draw is a workflow diagram.

If you draw a square, it represents transitioning to a new screen. If you draw a circle, it represents transitioning to a new frame, while staying on the same screen. Arrows are the action the user takes.

A workflow diagram for opening and closing a single accordion looks like this:



If, instead, we moved the contents of the accordion to its own page or a modal popup, so that clicking the question took you to a new screen, then the app might look like this:



FAQs

What does normalized mean?

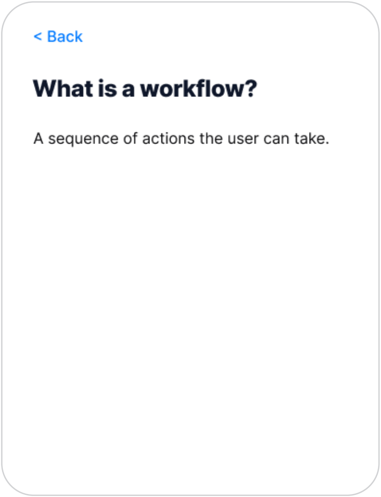
What does denormalized mean?

What is a workflow?

What is a screen?

What is a frame?

Question Page



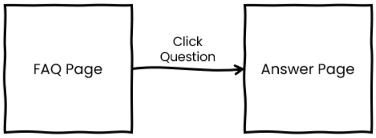
< Back

What is a workflow?

A sequence of actions the user can take.

Answer Page

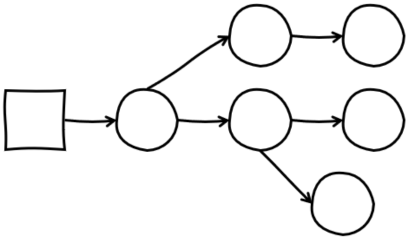
and the workflow diagram would look like this:



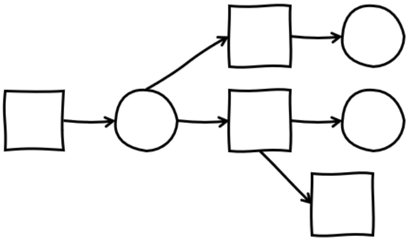
I tend to put the name of the screen in the squares, and a description of the state of the screen (the frame) in the circles.

Here we already arrive at an easy way to distinguish between normalized and denormalized interfaces. The diagram for a normalized interface has a lot of squares (screens). The diagram for a denormalized interface has a lot of circles (frames).

As you get used to thinking in these terms, you may not even need to draw everything out. But it can be very helpful when you're first starting out.



A denormalized interface
(few screens, lots of frames)



A normalized interface
(more screens, less frames)

How Do You Normalize?

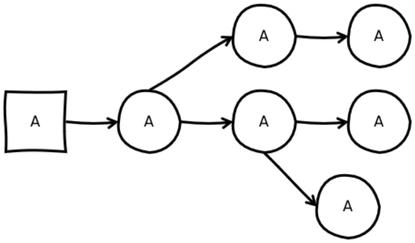
You may have noticed we've used the verb “normalize” and the adjective “normalized”. The verb “normalize” essentially means to adjust from a more denormalized to a more normalized state.

So, how do you normalize a workflow exactly? By adding screens. You either: **1) convert frames to screens**, or **2) split one screen into multiple screens**.

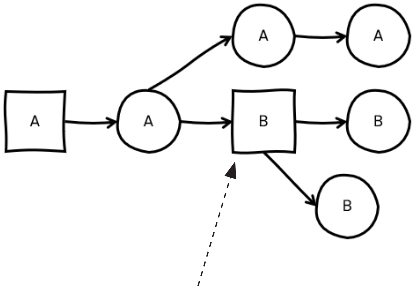
For frames, you take an action the user does in the software, and instead of that action keeping them on the same screen, it takes them to a new screen or modal.

Other times you take a portion of a page, and move it into its own page or modal.

In the workflow diagram, you might take a circle and make it a square. Or you might just add more squares.



In this workflow the user stays on screen A no matter what action they take.



We normalize an action in the workflow, taking the user to a new screen B.

If you had built our previous FAQ example, and you changed what happened when the user clicked a question from “open the accordion” to “go to an answer page”, you would have just *normalized* (verb) that “click the question” action. You would have made that workflow more *normalized* (noun) overall.

There are two major benefits to normalization, and each has even more positive side effects: 1) You reduce the user’s cognitive load at that point in the workflow and 2) You give the screen space to make that point in the workflow more understandable and guessable.

The word “cognition” means the mental action of acquiring knowledge and understanding. In other words, cognition means the work your brain does. By “cognitive load” we mean how much the user must think about and comprehend. When you split an action into a separate screen, you let the user *focus on just that action*, reducing their cognitive load.

By splitting an action into a separate screen, you also get the benefit of increased real estate. You can focus the UI elements of a separate page or modal on just what is needed to accomplish an action. You can place a few big, obvious buttons on the screen, instead of lots of small hard-to-notice ones.

We will get deeper into the implications and benefits of normalization in just a bit. But don’t think that this book is going to tell you to split your software across lots of screens. In fact, sometimes a denormalized workflow will be easier for your users to use!

What we really need, then, is a way to determine whether it’s better to build a normalized workflow or a denormalized one. We need a way to measure a workflow in our software and use that measurement to make decisions.

There’s a word for “a method of measuring something”: it’s called a “metric”.
Surprisingly, I’ve found you only need two metrics to decide between normalization and denormalization: frequency-of-use and complexity.

Get The Full Book

Thanks for reading these sample chapters!

You can get the full book, along with a companion video series at normalui.com. Plus, I'm in the video forums answering questions.

You can also find me, my social media links, and my courses on my website: tonyalicea.dev. Or drop me a line at hey@tonyalicea.dev.

- Tony Alicea