QB2.0 : *why*

The Future of Software Creation in 5 Probabilities

quickbase

The Future of Software Creation in 5 Probabilities

Low-cost (~2 existing resources), proposal to explore a “Quickbase 2.0,” not a “headless QB,” but a new head.

I believe that for the future of software creation, there is a significant chance that a no-code platform, offering back-end database architecture combined with custom front-end building, will facilitate and drive the expected 10x growth of software applications, with the potential to capture a percentage of the total software market.

The Future of Software Creation

R&D *(Research & Design)*

Comparisons with the gaming and CMS industries suggest a no-code software platform could create a double-*(or triple)*-digit Billion dollar company.

Most of the functionality exists, between QB database API's and a FE builder with API's. Therefore we can design this product with limited (.2) developer resources. Key steps will be to demonstrate a proof of concept, integrate a capable FE builder, discover technical gaps, and integrate the capabilities with UX.

TOC — 5 Probabilities

PART 1 — WHY	*CONFIDENCE*
1. 1 million ISV's by 2027	**90**%
2. Software creation will be AI assisted	**90**%
3. No-code will be dominant	**70**%
4. LLM will augment UX, not vice versa	**60**%
5. Software will have a GUI	**90**%

1 Million Independent Software Vendors (ISV's)

Level of confidence — 90%

I estimate there are more than 100,000 software companies (ISVs) today...
– up from 10,000 only 10 years ago. I wouldn't be surprised, with the **level of hyperspecialization new buyers are demanding**, to see that number grow to 1 million by 2027.[1]

Jay McBain, FORRESTER®

SAY MORE

...not to mention digital transformations / internal software applications, software creation will need to be radically easier

Sources: 1. Jay McBain;

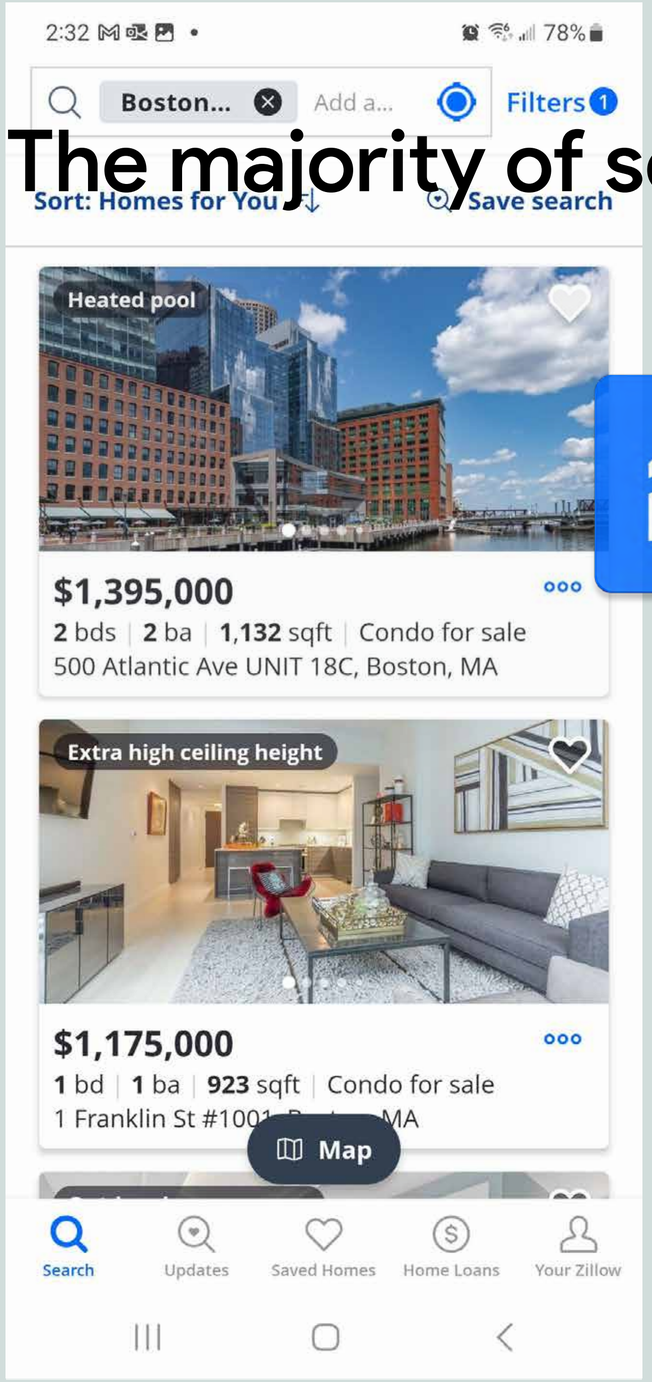
Hyperspecialization

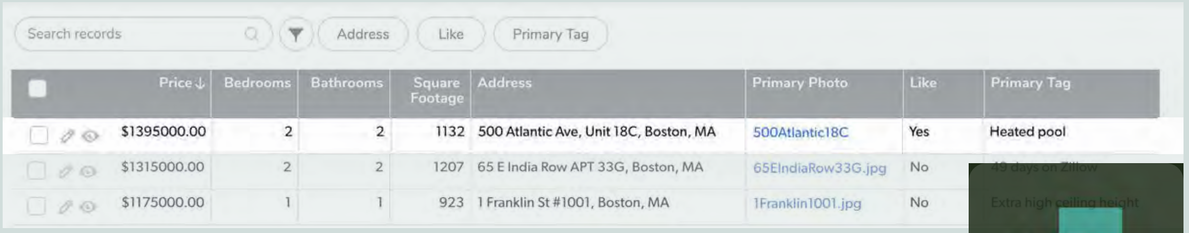
The problem ⇋ opportunity for 1 million ISV's

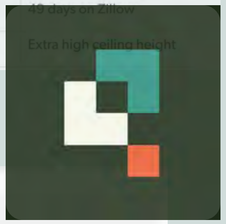
The massive increase in software applications and software vendors may mirror the rise of Quickbase, which **democratized database applications**

- **Problem & opportunity** - a two-way causality, where before there was not as great a need for bespoke, work-management solutions, and now there is the potential for a solution and an increasing complexity of work. The two forces will drive and enable each other, while also giving rise to many niche players
- **Most software is database software** - therefore Quickbase API's can go a long way towards powering it. It just needs a front-end and logic builder

The majority of software is database software







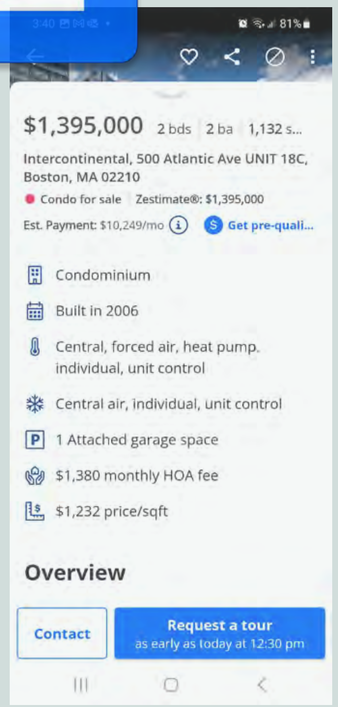
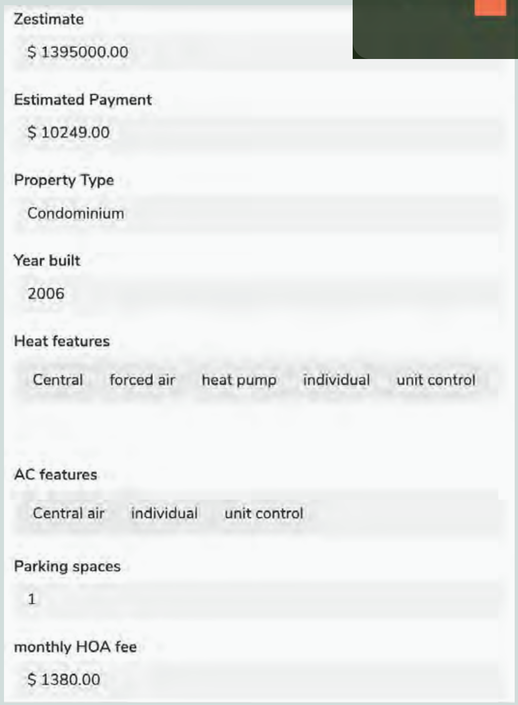
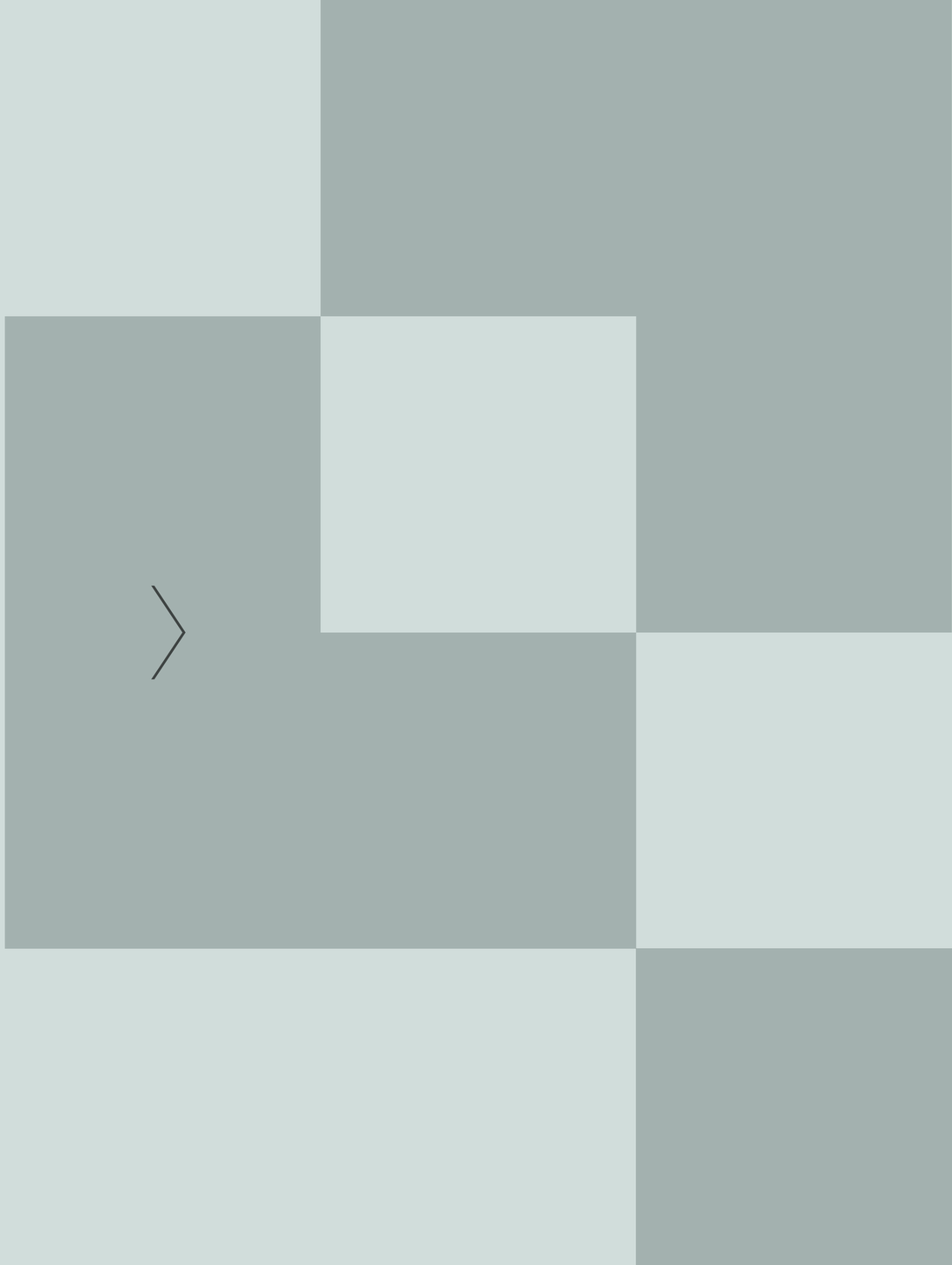


table reports
&
records



Radically easier

The capabilities for this 10x increase exist, but are fractured and too difficult for non-expert creators who will necessarily drive the growth



Software Creation will be AI Assisted

Level of confidence — 90%

- AI will reduce barriers to entry
- AI may reduce costs, enabling smaller isv's to survive
- AI may reduce available tech positions, prompting startups
- AI can help new creators achieve hyperspecialization
- **Customization and control will require a human (or company) in the driver's seat**

The operative word is "assisted," there will still need to be a builder environment or IDE

RACI in the age of gen AI

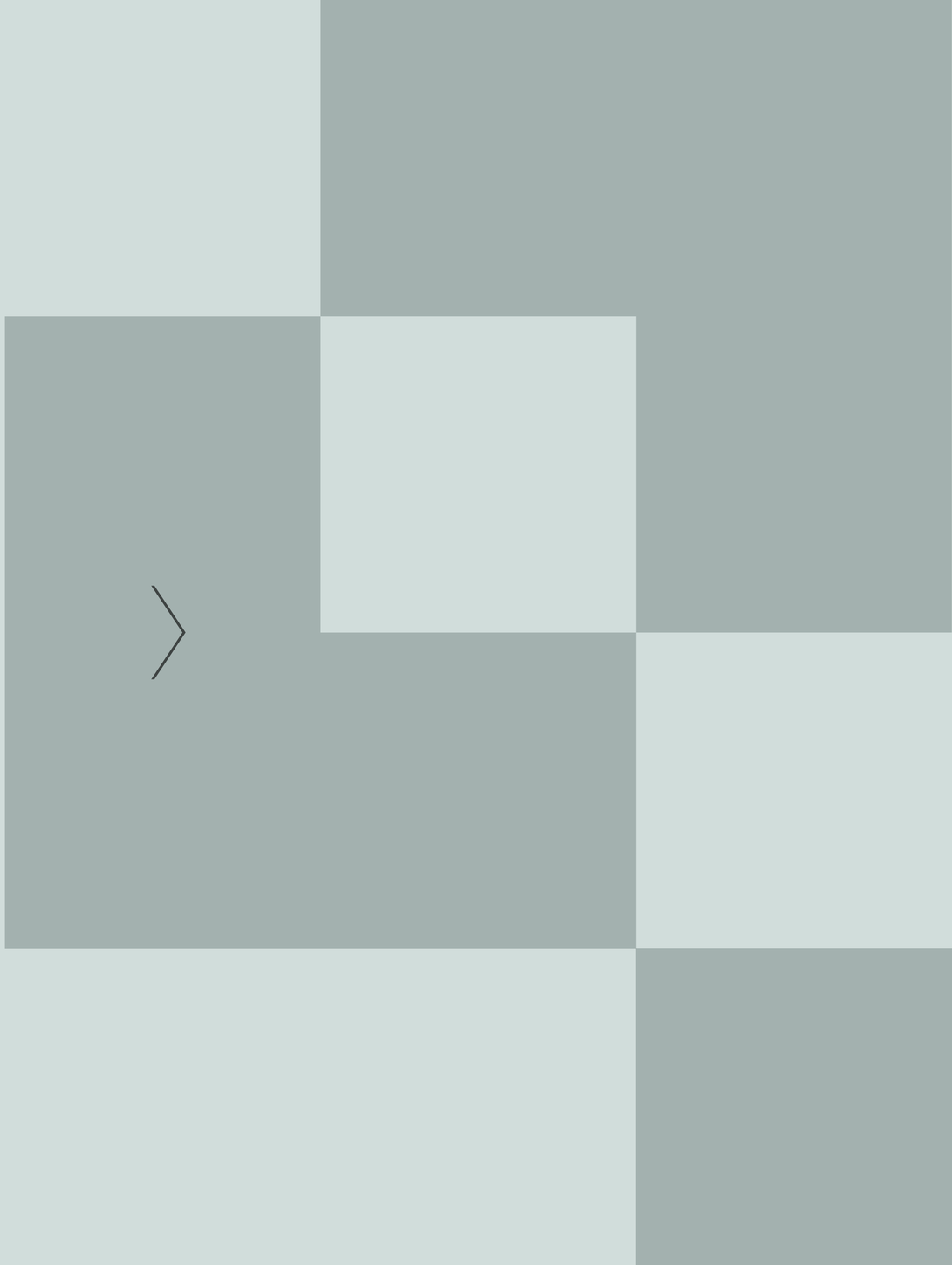
Even IF gen AI could handle end-to-end software creation

Let's say, hypothetically, that by the end of the decade, gen AI can be 100% **Responsible** for software creation

- **Accountable** - there will be bugs, security breaches, and liability, which OpenAI et al will disclaim. The human developers or platform owners will be accountable and therefore must be able to review the software in the build environment
- **Consulting** - for the foreseeable future, humans will want to edit the AI output. This is as much a limitation of human expression as LLM understanding
- **Informed** - even in the case of flawless AI execution, humans will need to review for QA, security, compliance, ALM, etc.

To code, or not to code?

With a 10x increase in ISV's and software applications , will the majority be built in code or no-code?



Proofs of Concept

Illustrating possibilities and discovering limitations

Manually create a web application from scratch
Use a front end builder or CMS, delivering data via Quickbase API's, with functionality from 3rd-party software, to replicate an existing software.

Discover technical limitations
Between QB, CMS, and 3rd-party functionality, what is missing to make functional software? Likely if-then logic (beyond QB pipelines), and dynamic addition of front-end modules to the UI.

No-code will be dominant

Level of confidence — 70%

By 2025, 70% of new applications developed by organizations will use low-code or no-code technologies, up from less than 25% in 2020.[1]

Gartner

No-code platform(s) will also need to be much easier to enable a 10x increase in software

Sources: 1. Gartner

The end of expertise

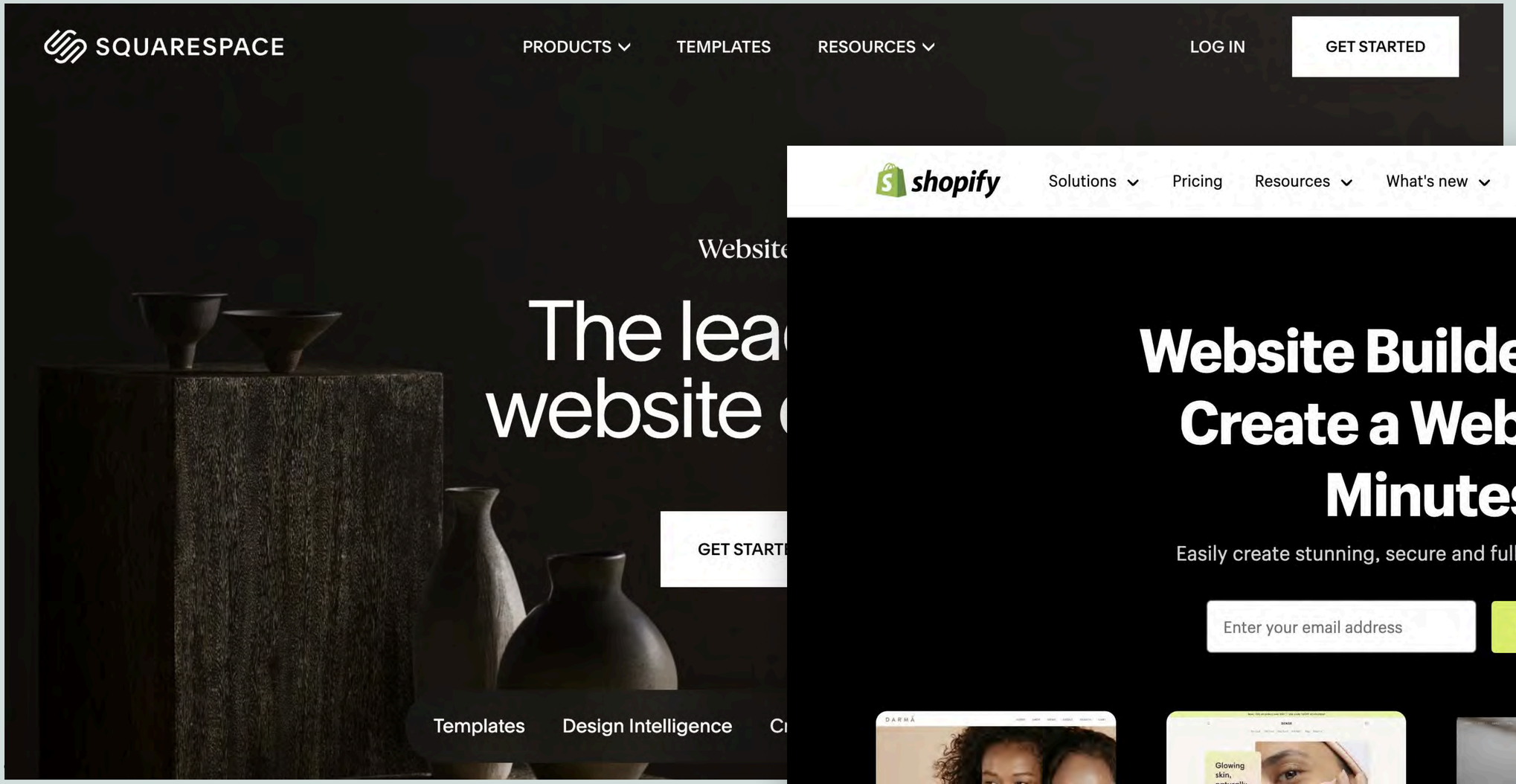
There aren't enough SWE's, nor will there be

Some argue that code development will become easy enough, with AI assistance, for non-experts to build software, but this doesn't hold water

- **Expertise required** - as explored above, there will need to be expert humans to review the output of any software creation environment
- **Moral hazard** - non-senior-level developers who rely on AI assistance will not reach the level of expertise to ensure code is deployable
- **Modest growth** - the number of new enrolees in SWE education shows modest growth, not enough to staff a future that relies on code development

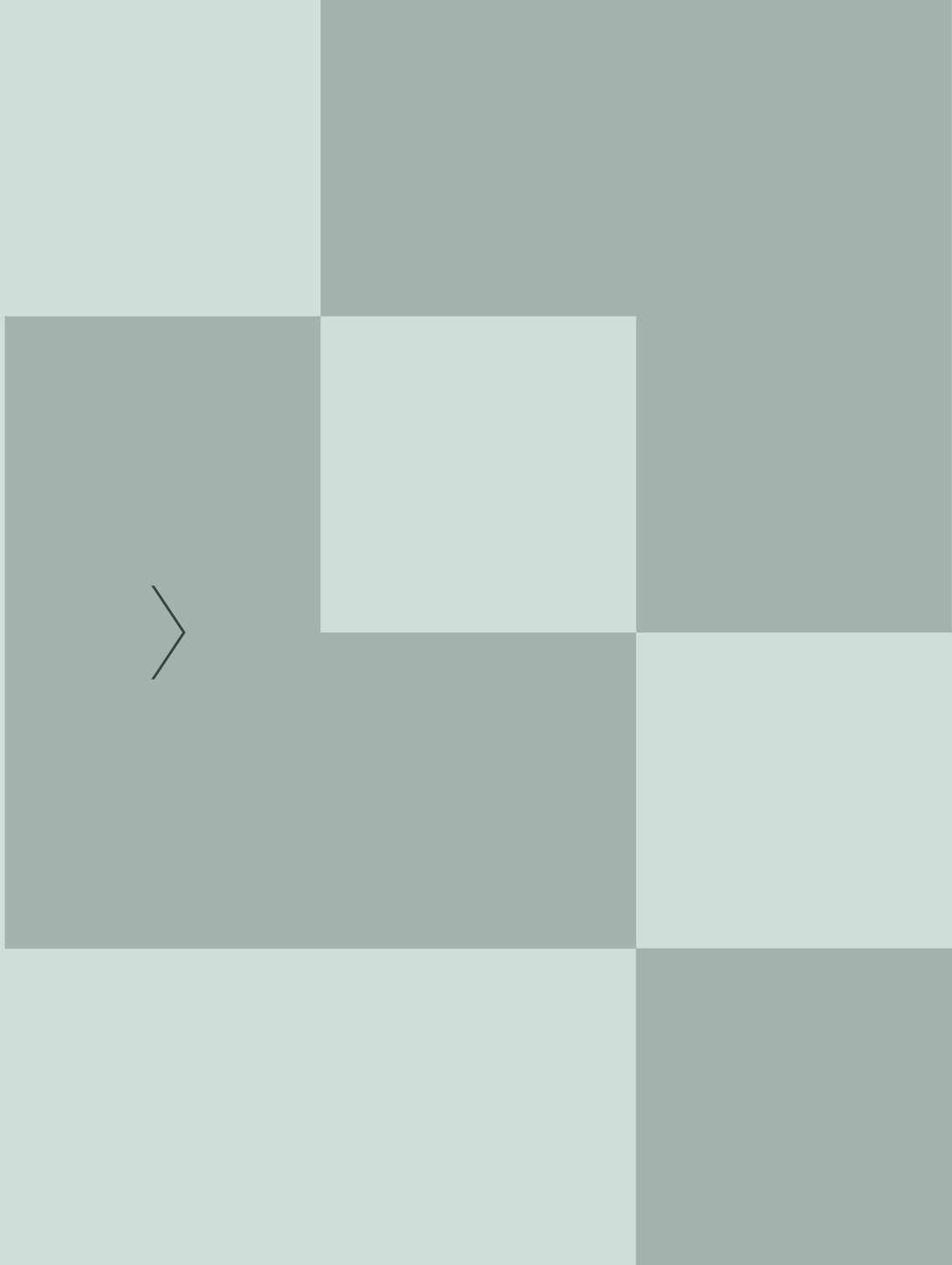
A similar trajectory

Web development is 99% no-code



Web builders rely on UX

As a predictor, front end web builders rely on UX over LLM, as users are able to choose from menus and options. Will software follow the trend?



No-code software will rely more on UX than LLM

Level of confidence — 60%

Across industries, fully 81% of all customers attempt to take care of matters themselves before reaching out to a live representative.[1]

Harvard Business Review

The fundamental challenge is that **users don't know exactly what they need**

We assume that, just as software creators will build within a visual interface, they will also primarily deploy a user interface

Sources: 1. HBR

Faster horses, faster

The nature of LLM based gen-AI

The initial limiting factor of a “pure chat” experience is that users don’t know what to ask for

- **Understanding their own needs** - what is the problem they’re trying to solve?
- **Knowing what’s possible** - without knowing the capabilities of a platform, users may not ask or may ask for something that’s not possible
- **Sacrificing specificity for speed** - the fundamental fact of using AI to generate something **complex** from a **simple** prompt is that the AI makes many assumptions. This may be fine in some use cases, like image generation, but when user needs are complex *and* specific, gen AI will inevitably be wrong more than it’s right

It’s hard to explain

Why templates are ubiquitous

Customers often have a vague idea of what they are looking for, but struggle to put it into words. However, they ”know when they see it”

- **Web design & development is dead** - website building relies almost entirely on template-based platforms
- **New trial onboarding data** - shows that the UX strategy was sound: presenting the user with a human-made template
- **Past the blank slate hurdle** - it’s easier for a user to see what they *don’t* like, and want to change, rather than start from scratch. The UX challenge is then to make **editing** as easy as possible, which will almost always be tactile

Easier done than said

vs. “switch the horizontal alignment of the product image with the text...”

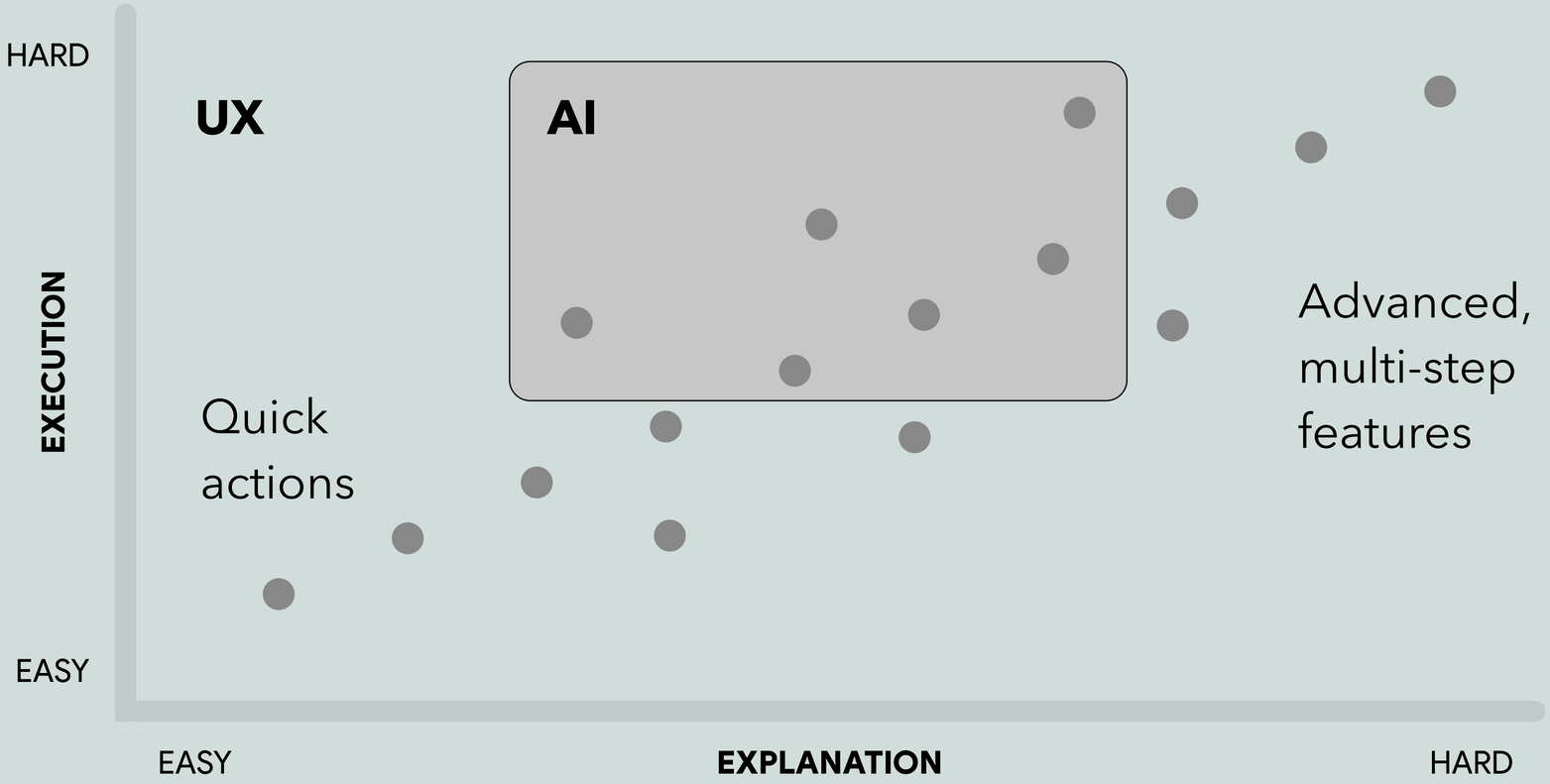
Green Water Bottle

$29.00 ★★★★★

16 oz., stainless-steel insulated bottle keeps your beverage cold or hot for 12 hours straight. The plastic coating offers is made from 100% ocean plastic and offers a comfortable grip.

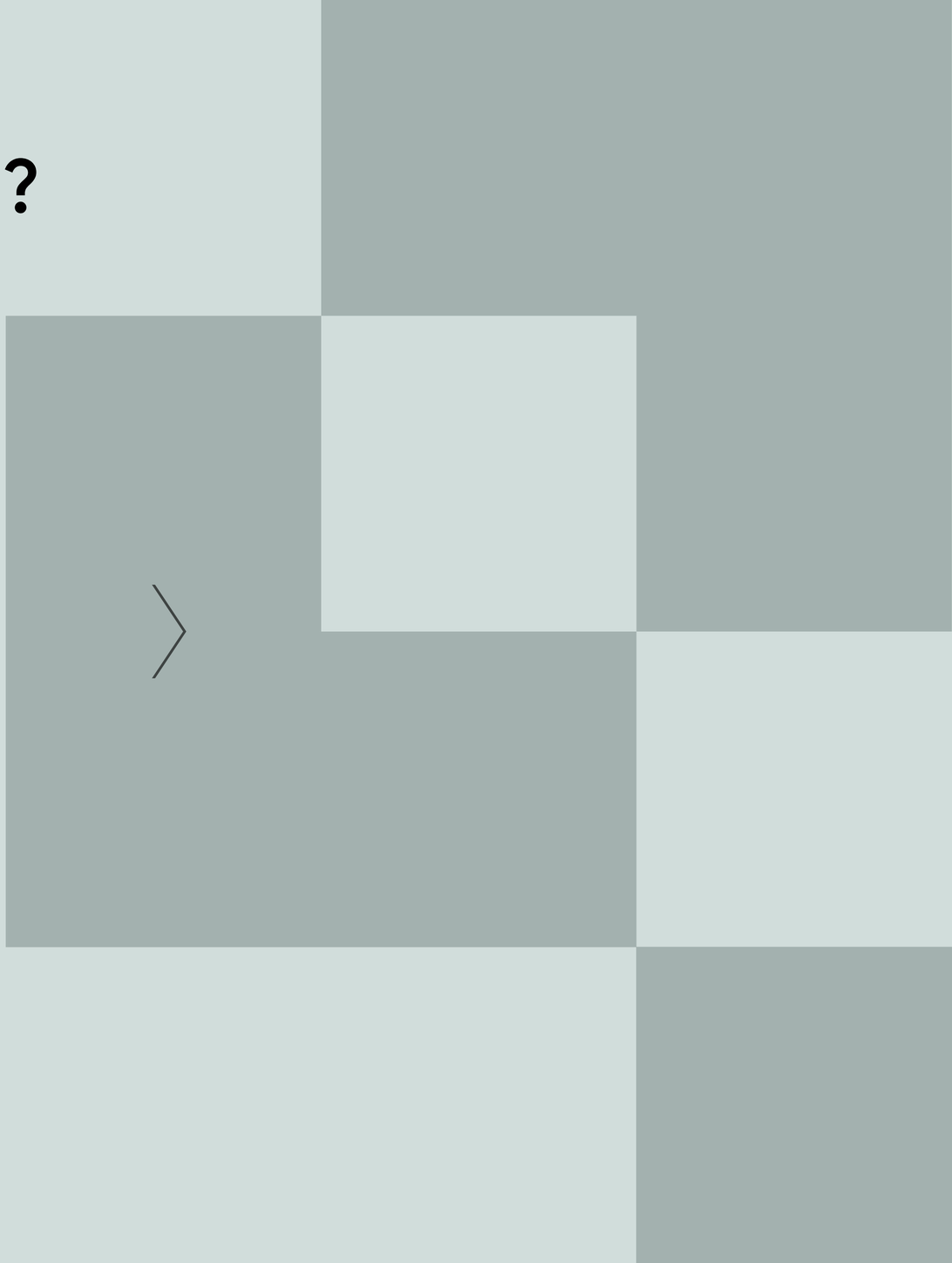
Gen AI sweet spot

Ease of explanation vs. ease of execution



How will software be deployed?

For similar reasons that software building will require a GUI and UX, end-user software will also require a front-end experience



End-user software will need a front end

Level of confidence — 90%

Non-visual UI's, such as voice, face several challenges:

- **Complexity of tasks** - multi-step or data-rich actions require visuals
- **Context and precision** - such as selecting items from a list
- **User preference** - many prefer the tactile and immediacy of visual UI's

Ease of creation and specialization suggest increased demand for branding, customization, and polish

That front end will need to be deployable to custom URL's or native applications (i.e. it will not be in a realm)

In summary

Level of confidence — 30.6%

1 million ISV's	90%
AI-assisted	x 90%
No-code	x 70%
UX > LLM	x 60%
Needs FE	x 90%
	30.6%

I believe that for the future of software creation, there is a **30.6%** chance that a no-code platform, offering back-end database architecture combined with custom front-end building, will facilitate and drive the 10x growth of software applications.

I believe that this platform will require not just the technical capabilities, but also a UX that radically simplifies software creation, including an understanding of the logic that drives the interplay of back-end and front-end for the end-users

Market (rough estimates)

The global software market is $737B in 2024 and expected to reach $2.25 trillion by 2034 [1]

Precedence RESEARCH

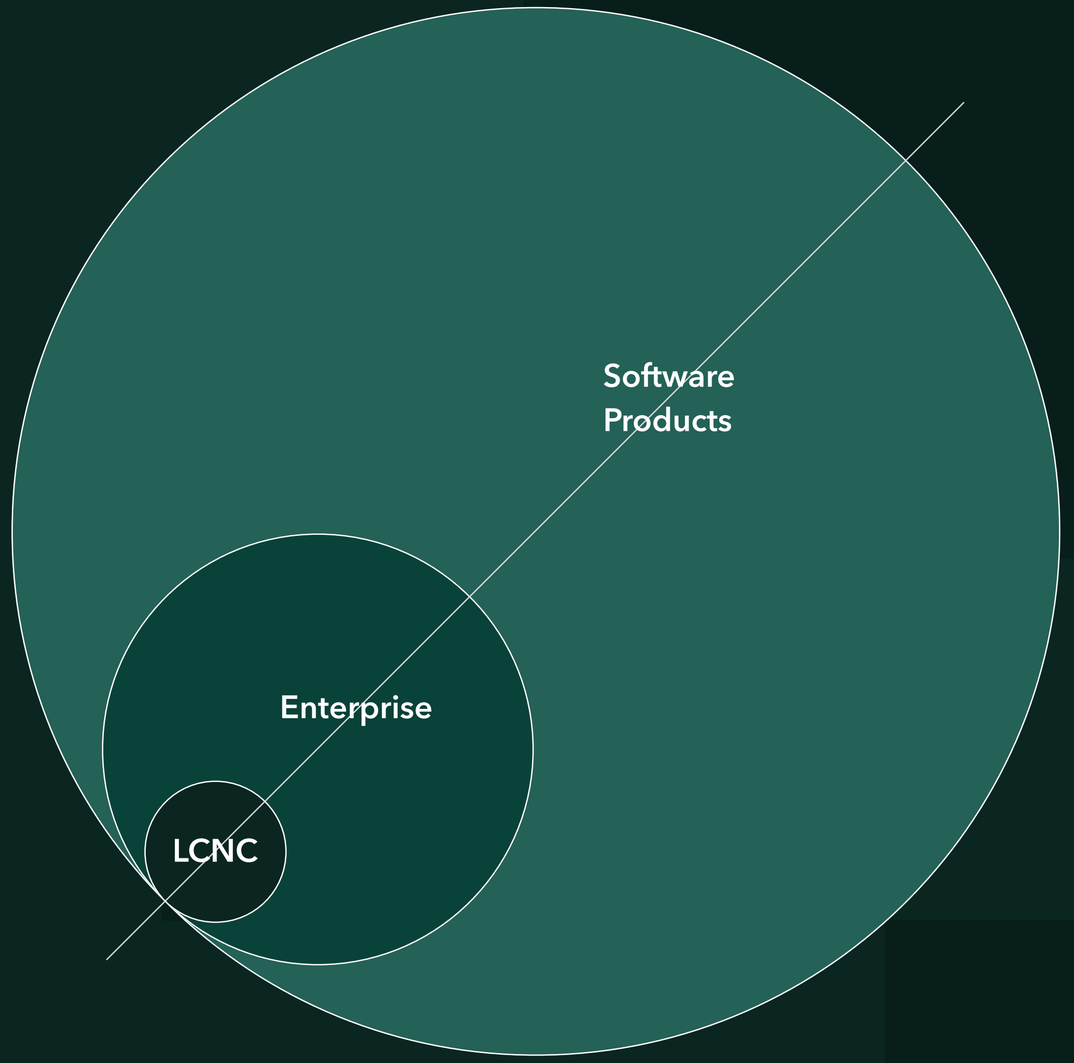
Enterprise valued at $294B in 2024 [2]

statista

Low-code platform market $32B in 2024, forecast to $65B by 2027 [3]

statista

The gaming software industry offers a useful comparison



Sources: 1. Precedence; 2. Statista; 3. Statista

Gaming comparison

PC games revenues are estimated a $42B in 2024, 23% of the total $188B market [1]

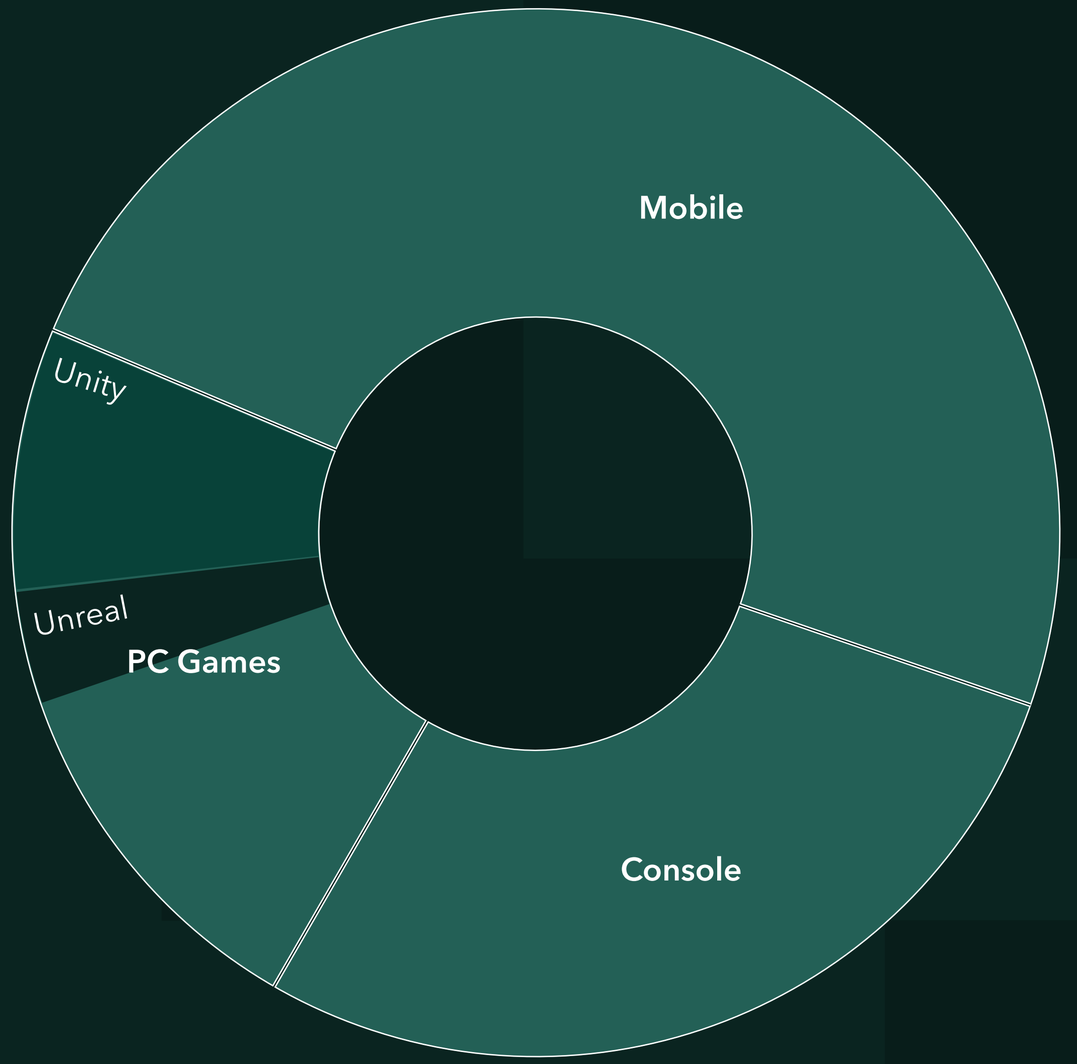
newzoo

60% of game developers use a game engine, of which 38% use Unity and 15% Unreal [2]

/DATA

This 32% slice of the PC game development market gives Unity (U) $2.2B in revenue and a market cap of $8.6B, while privately held Epic Games, with more of a walled garden ecosystem, has $5.8B in revenue [3] and a value of $22.5B [4]

statista Forbes



Sources: 1. Newzoo; 2. Slashdata; 3. Statista; 4. Forbes

Content Management Systems

Another angle for market sizing

In 2011, 75% of websites were hand-coded, today ~75% are built with a CMS or web builder [1]

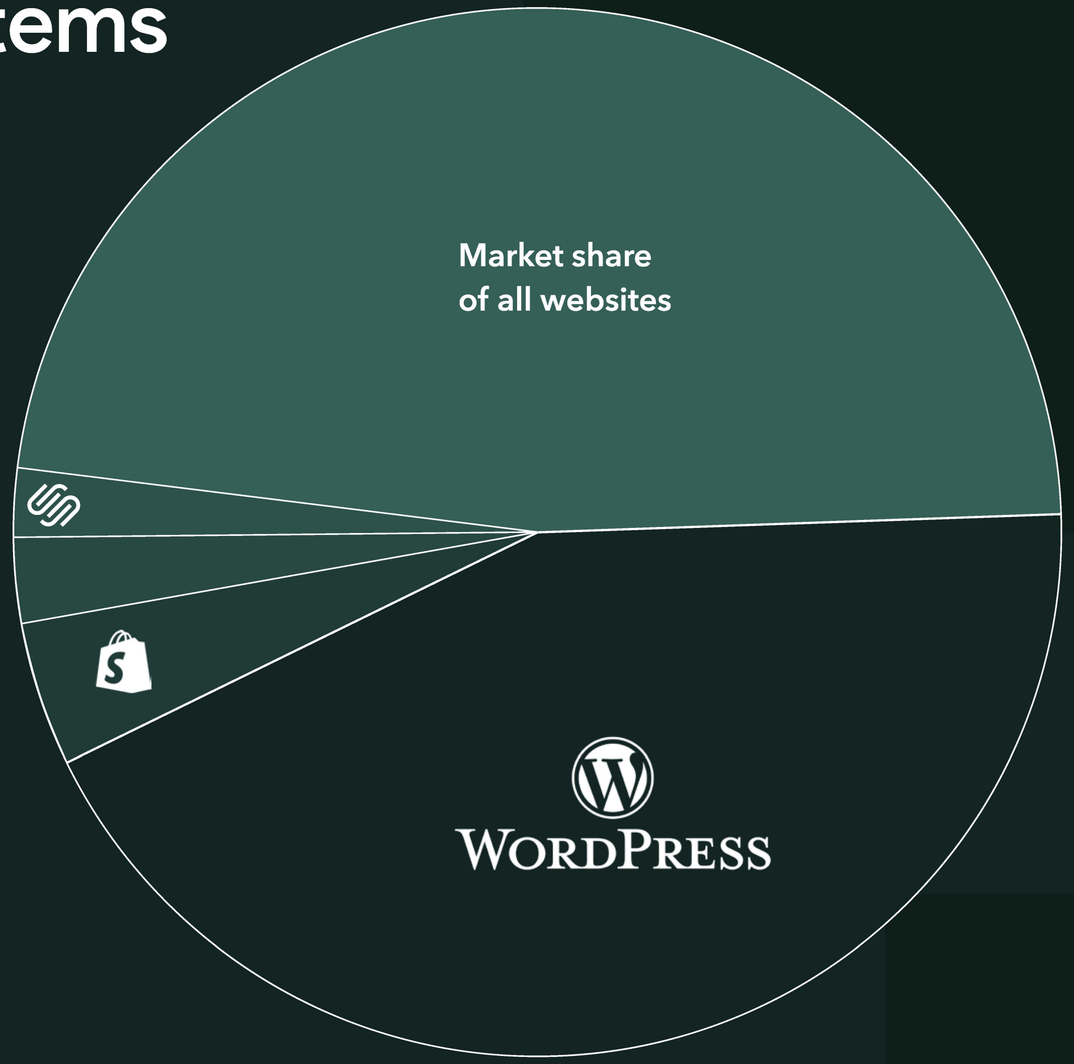
SQUARESPACE (SQSP) has a 2.1% market share of all websites and a $6.5B market cap

(WIX) 2.7% of market and $9.1B value

[redacted] (SHOP) 4.4%; $104B market cap

Wordpress accounts for 43.3% of websites on the internet and has an estimated value of **$600B** [2]

WP engine



Sources: 1. wpbeginner; 2. WP engine

Potential

Level of confidence — a good bet

Could a QB2.0, *a software engine,* powered by existing QB API's, supplemented with a front-end builder, capture a similar 19% of the multi-trillion-dollar software market of the next decade?

Software to create software, with no code, accessible to the layperson, would be the holy grail.