Teen Hacker MBA

Table of contents

Who is this for?

This document is for teenage hackers who are already great at technology but lack and want to learn the business side of software. These business skills include:

- *Product idea* — finding ideas, distinguishing good from bad ideas, validating ideas, etc.
- *Market research* — finding competitors, estimating competitor revenue, marketing channels, etc.
- *Product development* — product planning, product positioning, product pricing, etc.
- *Marketing* — SEO, content marketing, organic marketing, paid marketing, sales, etc.
- *Product sales* — selling software internationally, legal stuff, collecting VAT, paying taxes, etc.

- *Product management* — getting user feedback, feature prioritization, planning, releases, retiring features, etc.
- *Running a business* — needed roles, hiring, firing, vision and mission, funding, scaling, etc.

Why read this?

Because it is easier for hackers to understand business than it is for business people to understand technology. Most successful software companies are run by ex-programmers, engineers, physicists, and mathematicians. Some examples are:

- *Bill Gates* — Gates was programming since high school and programmed the first version of Microsoft BASIC (wiki). Gates dropped out of Harvard after the first year and never finished it.
- *Jeff Bezos* — Bezos got a bachelor's degree in electrical engineering and computer science (wiki), before Amazon he was working as a quantitative analyst at a hedge fund.
- *Larry Page and Sergey Brin* — Both were computer science Ph.D. students at Stanford when they founded Google (wiki).
- *Jensen Huang of Nvidia* — Huang got a master's degree in electrical engineering from Stanford (wiki).
- *Steve Jobs and Steve Wozniak* — Both Steves were hackers. Jobs dropped out of Reed College and worked as a technician for Atari before founding Apple (wiki).

If you are a hacker who knows programming, math, and thinks analytically, then learning the business side of software is not hard. You can use online resources and you don't need to go to a business school. Schools are tailored for an average student and they teach general topics. Knowing accounting standards will not help you create a successful tech startup.

License

About the author

This document was created by Zeljko Svedic (LinkedIn, blog), for educating top students from Croatian IT competitions (website).

Zeljko has been fascinated by computers since primary school. During high school, he got into competitive programming, which culminated with medals from a few competitions. As side projects, he developed an MS-DOS 3D space simulation game, an incremental parser, and a joke-sharing website. After getting a degree in computer science in 2002, he worked for a few years at Microsoft and Infragistics.

In 2005, he moved to the dark basement of his parents' house to start his own company. Notice how in the United States, everybody starts a business in a garage, while in densely populated Europe, basements and attics are more popular startup choices. Although he didn't see much sun, that proved to be a good business move. The first product he created, GemBox.Spreadsheet, is still selling quite well eighteen years after its creation. He hired some people, and they created GemBox.Document, which achieved similar revenue in just a few years. After that, he decided to venture into the recruitment testing area, and co-founded TestDome, a service for automated testing of programming skills.

All above mentioned are nice and dandy, but Zeljko had his own share of failed projects. For example:

- Unprofitable components:
    - GemBox.CompoundFile — A .NET component to read and write legacy container formats of office files. It made very little money and was retired. See the 3rd party archive.
    - GemBox.Spreadsheet.Java — A Java version of GemBox's popular .NET spreadsheet component. It made very little money and was retired. See the retirement blog post.
- Tools for pay-per-click (PPC) advertising:
    - GemBox.Ppc — A .NET component to programmatically manage PPC campaigns. It made very little money and was retired. See the old Visual Studio marketplace page.
    - WisePPC Editor — A Windows application to edit Google AdWords campaigns. It made no money and was retired. See the old tutorial video.
    - WisePPC Integrator — A cloud service to sync e-commerce platforms with PPC campaigns. It never passed the MVP phase.
- Other projects:
    - Zagreb Cohousing — Experimental coliving community in Zagreb. It closed the door after a year, and it never covered the full cost of house rent. See the Zagreb Cohousing FAIL article.
    - Various random video chat ideas — Multiple MVPs based on the idea of random video chat, but for filtered audiences with shared interests. All MVPs failed, same as Sean Parker's AirTime startup.

As you can see, Zeljko had more failed projects than successful ones, and that is OK. Each failure is a lesson. This document condenses lessons learned from both successes and failures.

In summary, Zeljko prefers:

- Bootstrapping over venture capital.
- Lean startup over stealth startup.
- Organic marketing over paid marketing.

Zeljko's views share many similarities with books by 37signals written by Jason Fried and DHH (founders of Basecamp and Hey), and with the writings of Joel Spolsky (founder of Stack Overflow and Trello).

General resources

Start with these general resources. Create a Google Document with study notes, write down what you have read, watched, found, and what questions you have. Later you can discuss those study notes with Zeljko.

Website

Hacker News (HN) by Y Combinator (YC) - This ugly website is the most important news and article source for hackers and startups (wiki). "News" in the name is a bit misleading, as the most valuable HN content are articles about technology, programming, startups, business, UX, and anything that "gratifies one's intellectual curiosity". Many famous startups posted their first announcement on HN, e.g. first DropBox post in 2007. YC

incubated startups like Dropbox, Airbnb, Stripe, etc., and HN started as their internal news and article-sharing website.
It is easy to develop an addiction to checking HN every few hours and then spending hours reading interesting articles. Zeljko is a recovering HN addict, so now he only checks HN's best recent articles section every few days.

Book

The Lean Startup (LS) by Eric Ries - This is the startup bible. Everybody in the startup community knows it, and it started a lean startup movement with members in every country. Zeljko summarized the book's principles in the first LS Croatia talk: Lean Startup i Minimum Viable Product
If you want to get a physical copy of the book, no problem — reply to Zeljko and he will bring you a physical copy to the camp, free of charge.

As a side note, Eric is a cool guy and often speaks at conferences and poses for pictures :)

Videos

Startup School Winter 2020 playlist (20 videos) - YC has two schools: physical seasonal batches (e.g. YC winter 2022 batch - W22) and YC Online Startup School. Zeljko and Mario applied with TestDome to the YC winter 2014 batch but were not accepted. But in 2017 they applied to YC online startup school and were positively surprised that YC had posted videos of their physical batch lectures for free on YouTube. So now you can watch the same talks as elite startups invited to Silicon Valley!

Ideas: basics

Common misconceptions

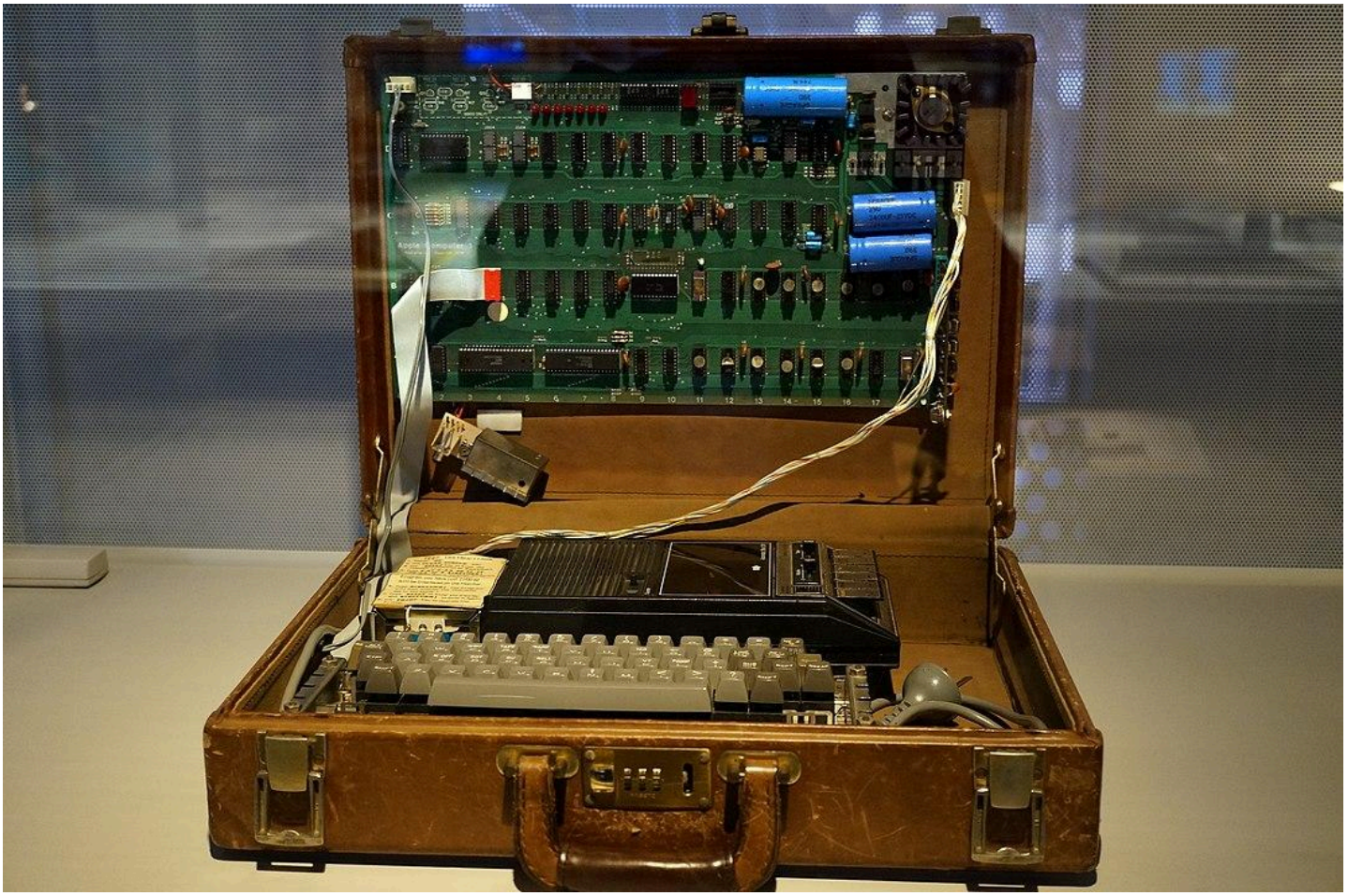
Finding a good idea is a big problem for new entrepreneurs. Unfortunately, they often have a completely wrong understanding of what makes a good idea. For example, many new entrepreneurs think like this:

> "*I need to invent something completely new. I shouldn't make a product that already exists, because there are established competitors. My new thing should be so original that I can patent it. After I patent my genius idea, nobody will be able to steal it from me. If the idea is not*

> *patent-protected, I will ask people to sign an NDA before I tell them. Even better, I will have a stealth startup. Only when everything is done will I reveal it to the world in a spectacular presentation.*"

It is easy to recognize such entrepreneurs at startup events. When asked what they do, they get evasive or speak in very general terms. They want investors to sign an NDA before hearing a pitch. They work on their precious idea for years without releasing anything.

I can understand where these views come from. There are many tales of great inventors who made money on their inventions. Apple would work in privacy on the next project until the grand reveal by Steve Jobs. What those stores miss is what influenced famous people, who were their competitors at the time, and the many failures they had before making a big success. For example, the invention of the radio is disputed between Marconi, Tesla, and Hertz. They all build on the work of each other. The first company of Bill Gates, Traf-O-Data, failed. Steve Jobs said "Good artists copy; great artists steal"—and at the beginning of his career he studied Sony's product design and marketing. Steve Jobs is commonly presented as a genius who created new, beautiful, minimalistic products out of thin air. But his first 1976 Apple 1 computer looked like this:



In other words, laymen often think inventions and entrepreneurship is a matter of inborn genius. In reality, famous people worked in context, had many failed attempts, and learned through the years.

What works

Eric Ries has a great summary about ideas in the Lean Startup book:

> "If you really believe someone will steal your ideas, take one of your ideas, and try in every way imaginable to MAKE someone to steal it: create a PR article, publish everywhere, call people from companies, send emails, and see if anyone will really steal your idea - they just won't.

Companies have enough ideas. What they are struggling with is prioritization and delivery. They just don't care about your ideas."

So, instead of having one genius new idea and believing in it, try the opposite:

- *Have many ideas, not just one*.
  You wouldn't buy the first car you see at a car dealership, would you? In the same manner, before committing to a project for the next few years, do an informed idea "shopping". Research at least a dozen ideas before you commit to the best one.

- *Ideas that are "completely new" and "have no competition" are usually not good, they are bad*.
  They are a few reasons for this:

  - *Why is nobody in the world doing that?*
    It may be because you are the first genius who thought of it, or because it is a bad idea. Out of those two, what do you think is more likely?

  - *If you are right and the completely new idea is great, how will you market it?*
    Digital cameras were invented in 1975 but didn't really take off until 20 years later. Portable digital audio players were invented in 1979 but didn't become popular until Apple iPod in 2001. It is very hard to convince people to buy something if they don't know it exists.

  - *If it takes 10 minutes to explain what your software is, you are not going to make money*.
    Digital cameras and mp3 players took off only when an average consumer knew what "digital camera" and "mp3 player" is. The same goes for software, where popular software categories are: "CMS", "CRM", "ERP", "antivirus", "VPN", "e-commerce", etc. Having a short category name is a sign of market maturity: there are thousands of companies, customers know what it is, and you can target marketing for those keywords (e.g. "ERP for manufacturing").

  - *Maybe the market is too small?*
    Maybe an idea solves a real problem, but there is no competition because the market is too small. The joke in the startup community is that a startup "has only one client in the world"—the startup founder themselves.

  The above reasons explain why completely new ideas fail more often than they succeed. Still, there are counter-examples: Netflix invented DVD delivery and movie streaming, Amazon invented the cloud, Tesla made the first electric sports car. But those were made by experienced founders: Reed Hastings first made debugging software, Jeff Bezos first made a web bookshop, and Martin Eberhard first co-founded and sold the Network Computing Devices company.

- *Don't get attached to an idea, be critical instead.*
  I've seen many people waste years pursuing an idea that doesn't make sense. Common advice that "you need to believe in your idea" is misleading. Ideas are not religion, you don't have to "believe" in them. Instead, you should be critical. If the new information shows that the original idea was bad, modify the idea or find a new one.

- *It is OK to copy ideas from others.*
  Many people have a big problem with this. They think that copying an idea is cheating, an easy path. It is not. The hard parts are:

  - *Knowing what to copy.*
    The world is full of bad ideas, even well-established ideas that are bad.
    For example, before the dot-com crash, people thought that:
    A. Yahoo-style directories were the future,

B. Generic domain names are like prime real estate, and
C. TV commercials are needed to get market share.
There were thousands of startups based on those premises. Companies wanted to be a directory for pet supplies (A), spent millions on generic names like "pets.com" (B) and spent even more on Superbowl commercials (C).
They all went bankrupt in the dot-com crash. These premises were dead wrong:
A. Directories concept died and was replaced by search,
B. Biggest search engine became "Google", an invented name (because Sergey and Larry didn't have money for a "proper" domain).
B. People buy pet supplies from "Amazon", same as all other stuff, no separate marketing is needed.
You had to be very smart in 1999 to know whom to steal from. What is obvious in hindsight was not obvious then.

- *Knowing how to copy something.*
  What is harder: creating an original painting, or creating an exact copy of the Mona Lisa? Of course, copying the Mona Lisa is harder, but you may argue that is because of the special techniques used. Then consider Hemingway's The Old Man and the Sea. The book is short, the plot is simple, and there are just a few characters. Hemingway uses plain words and short, simple sentences. Read a random page from the book, close it, and then try rewriting it in your own words. I bet text copied from your head will not be worth a Nobel prize in literature. The same is for startups, it takes a lot of knowledge to copy success. In the process of copying, you will probably change it so much that people will not recognize the original you copied.

- When you know *what* and *how* to copy, you will probably *copy and combine from multiple sources*.
  As soon as you combine more ideas into one, it is not stealing anymore, it is remixing. The original iPod player was a mix of existing ideas: a digital music player, Walkman form factor, and portable 2.5-inch hard drive. Steve Jobs introduced the iPhone as a mix of iPod, phone, and Internet device.

- *Talk with everybody about your ideas, to get feedback.*
  If you keep your ideas secret, you will never get feedback. Each person you talk with can give you different feedback. Programmers will suggest a technology. Marketers will suggest a target audience. VCs will talk about monetization. Experienced users will tell you what app they are using instead. Old granny will tell you why she would never use it.

- *Do market research homework*.
  The Internet has infinite resources you can use to research an idea: Google search, keyword research, LinkedIn company profiles, company tax filings, business directories, product review sites, discussion forums, etc. Structure what you find in a document, so you can share it with others. You wouldn't believe how many times I heard "this is a unique idea", only to find an app that already solves that problem within 30 seconds of a Google search. One time I watched founders present the "first app for blood donors in Croatia", only to find an existing app, that the founders had never heard of, before they finished the pitch. People sometimes like their idea so much that they hide it from the real world, and are unaware of what market research would uncover.

- NEW: Base your idea on real-world pain points. Tell a story about Macola.

In short, have many ideas, don't be afraid to copy and combine, talk with everybody, and always do market research. Ideas can come from your pain points, from projects you do for others, from looking at other successful companies. People who are naturally curious often have many ideas. Keep piling ideas and market research documents until you find one that stands out from the rest. Only then can you allow yourself to get attached to the idea, and only until the real world proves the idea wrong.

Venture capital approach

Some advice I give is contrary to advice given by venture capitalists (VC is a short for both venture capital and venture capitalists). For example, VCs will say to think big, grow while generating losses, pursue completely new ideas, enter winner-takes-all markets, and exit early.

I think VCs's advice makes sense for VCs, but not necessarily for you. Out of many articles written on that topic, I recommend the following:

- Harvard Business Review: VC Funding Can Be Bad For Your Start-Up
- NY Times: More Start-Ups Have an Unfamiliar Message for Venture Capitalists: Get Lost

Top-voted HN comment for the above NY article sums it up well:

> "VCs win if enough of their bets make it big enough to offset the ones that go under. Naturally the big hits are few and the ones that fail are numerous. That means the big hits need to be huge and the failures need to have a certain cap. The latter also means you can't run a company for 10 years in slowmo until they get profitable. And the big hits need to be huge which means they need to take over a nice chunk of a market which you can only do with good funding and moving faster than everyone else. VCs were able to control the startups they invested in to a pretty big degree.
> What does that mean for founders? It means that you are putting all your eggs into one basket. A founder does not found 20 startups at the same time in the hope that one succeeds. A founder puts his effort usually into only one. So founders take a larger risk than VCs. They make it big or go bust. And the vast majority goes bust when running in supercharged mode.
> What does that mean for startup employees? Well, they got the worst of all worlds. High risk and little to no upside. Their eggs are also in one basket as they work for only one company AND they don't have the huge upsides that VCs and founders have.
> The whole game favors only VCs and founders who like big bets. But for the vast majority of people (which includes employees), the VC game is not a great game to play."

Think about it this way:

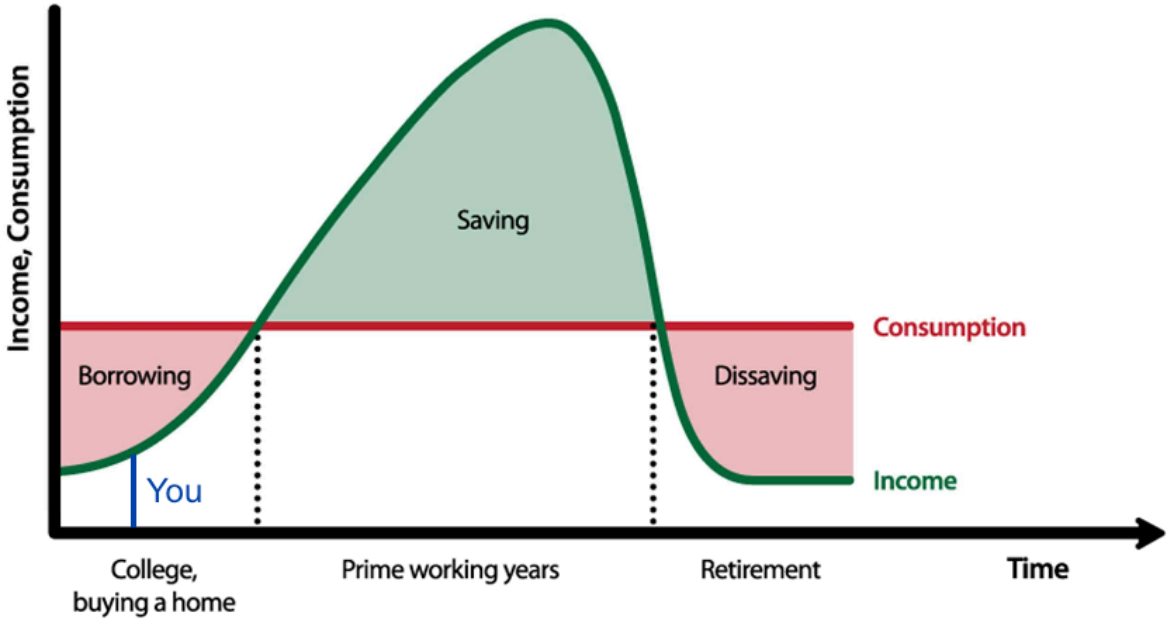
> Would you rather have (A) 20% chance of becoming a millionaire, or (B) 0.02% chance of becoming a billionaire?

Total estimated earnings are equal (20% x 1M = 0.02% x 1B). VCs push to make big bets (B). But I suspect most of you would prefer a higher chance of success (A). That is why this guide is focused on bootstrapping and going slowly. VCs will not tell you to bootstrap and go slow, because then you don't need them.

Choose projects that increase your practical competitive knowledge

Many students have money-making side gigs at Fiverr, Upwork or some small company. That is great for learning different languages and technologies, but be careful not to get seduced by money. It is a mistake to get a full-time job early if it blocks your learning.

That is because an average salary during lifetime is a hill-shaped chart:



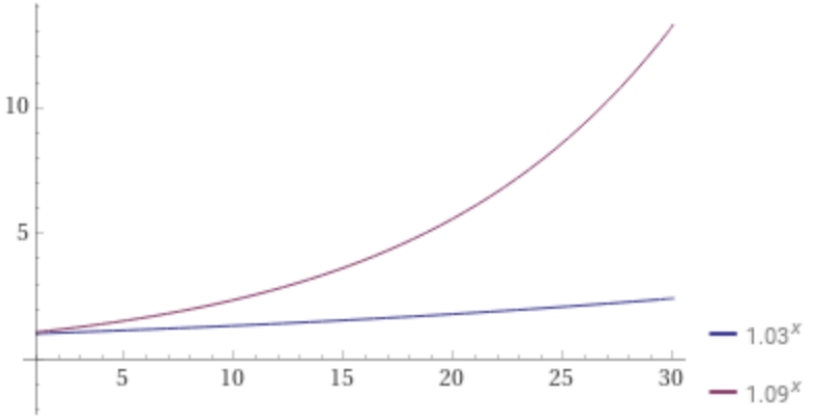
Young people have negative income while in school, and take money from parents, government, student loans or scholarships. After they start working, average salary increases with each year in the job, until prime employment years. Late in career, average salary drops as it is hard to keep pace with new technologies and trends. Older workers also prefer shorter work schedules and have more health issues. In retirement, income is again negative and retired people need savings, pensions, rental income, etc.

As a student you are at the very beginning of the chart, on the blue line. Note that:

A. You are at an exponential part of the salary curve.

B. The area below the curve is total income over the years, and the area to the left of the blue line is much smaller than the area to the right of the blue line. That means the total income you made until now is quite small compared to the total income you will make during your lifetime.

Because of that, your main goal now should not be to make money, but to increase exponential growth of your salary. Even small growth changes can make huge differences compounded over the years. For example, let's say person A's salary grows 3% per year, and person B's salary grows 9% per year. The amount increased each year seems small (6%), but over 30 years that results in a huge difference. A's salary is barely keeping with inflation, while B's salary skyrocketed (see Wolfram Alpha):



Over 30 years, total income of A is 47 base salaries, while total income of B is 141 base salaries. In the short term, it makes sense to stop education and just get a job. In the long term it is a bad decision, because learning is what makes your salary grow.

That said, formal education is not a guarantee you will learn useful things. My computer science degree required knowing how to solve integrals in the complex plane and knowing at which location an international prototype of the kilogram is stored. University taught me that I can't rely on government education and that I need to learn computer science by myself. As a high-schooler I taught myself algorithms, and as a university student I taught myself Smalltalk, Windows GUI programming, and programming web pages.

The fastest way to learn programming is via programming projects, and the fastest way to learn business is by running a business. So you should choose your next projects and jobs based on the knowledge you will get. In the startup world, the following are considered great learning experiences:

- Education at prestigious universities: Harvard, MIT, Stanford, etc.
- Work experience at Big Tech companies: Google, Amazon, Apple, Facebook, Microsoft, etc.
- Significant contributions to hugely popular open source projects: Linux, Bootstrap, PyTorch, etc.
- Getting accepted to physical Y combinator programs, like Dropbox, Stripe, Airbnb, etc.
- Running a successful startup, or having a successful exit via IPO or acquisition by another company.

Note that all of the above options are very selective. It is quite hard to get accepted to MIT, get a job at Google, or get your commit accepted to the Linux main kernel. This is the general rule:

> The more selective the learning experience is, the more valuable it is, because less people can learn it.

Working online at Fiverr or Upwork is good for a start, but is not considered a good long-term learning experience. Customers on those platforms usually have simple and repetitive tasks, and are not willing to pay for expertise. Anybody can join such platforms and claim they have the expertise. That creates something known as the lemon market. If you want to work on custom software projects, you can find a digital agency with high acceptance criteria and good employee ratings. Then learn how to ace their interview and get a job. There you will be working with different technologies and on different projects, but you will also learn teamwork, project management, UX and UI, cloud technologies, etc. If you are good, they will promote you to a team lead and then you will learn how to manage teams.

Market research

We will use a wide definition, where a market is a subdivision of the economy in which buyers and sellers engage in monetary exchange. Subdivisions can be made by geography (e.g. US market), language (e.g. French-speaking market), industry vertical (e.g. healthcare), or anything else. Economic books make a distinction between physical markets (e.g. stock exchange), geographics markets (e.g. countries), and market segments (e.g. housewives); we will use “market” as a colloquial cover-all term. The important thing is that one group of customers want to buy *from offerings in that market*. Each business is a part of multiple markets because people buy for different reasons.

For example, a Vietnamese restaurant is part of the following “markets”:

A. “Vietnamese restaurants market” — buyers want a Vietnamese restaurant.
B. “Asian restaurants market” — buyers want an Asian restaurant.
C. “Neighborhood restaurants market” — buyers want a restaurant they can walk to.
D. “Delivery food market” — buyers want their food delivered.

In my experience, most Vietnamese restaurants in Berlin try to please all four groups: they have Vietnamese food (A), but also Japanese sushi (B), they are in walkable neighborhoods (C), and they offer delivery (D). But that makes them non-distinguishable, with similar prices. Other restaurants serve a narrower market with target customers that will pay more and travel further, e.g. a vegan fine-dining restaurant.

It is your job as an entrepreneur to decide which markets you want to focus on. Not only that, you need to decide where you will be in that market, also called market positioning. Picking your target market and positioning inside that market is a part of brand management.

Bad markets and good markets

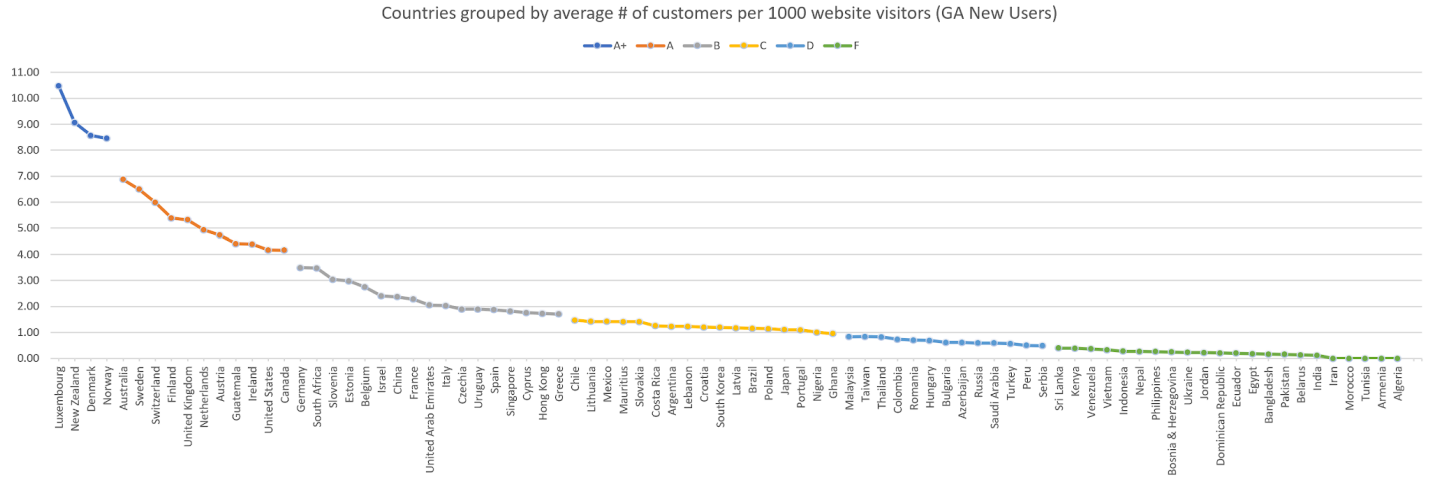
The most important general principle about markets is:

> A good idea for a bad target market is still a bad idea.

The reason is that *good markets are many orders of magnitude better than bad markets*. An excellent product in a bad market is still going to perform worse than a mediocre product in a good market. In startups, we measure "better" or "worse" by the ability to monetize. But the principle is universal, it holds for other domains where the measure is different: popularity, impact, awards, etc. For example, if you are an academic, publishing a paper in a trendy academic field will result in many more citations.

People intuitively know what markets are different, but it is very hard to comprehend cumulative orders of magnitude. Let me give you some examples with data.

Two software companies I founded, GemBox and TestDome, both sell internationally to a worldwide internet population. Both companies have trial versions. They have signups from every country of the world. I suspected that conversion rates from trial to paid version differ between the countries, so I analyzed two years of sales data. These are the cumulative B2B SaaS conversion rates for 84 countries:



The results are shocking.

Luxembourg, a small country with a population of 602,005, had 2,363 visitors and 27 paying customers, giving a visitor-to-customer conversion rate of 10.48 customers per thousand visitors.

India, a large country with a population of 1,343,890,000, had 746,893 visitors and 87 paying customers, giving a visitor-to-customer conversion rate of 0.12 customers per thousand visitors.

That means Luxembourg is 87 times better than India (10.48/0.12) if you look at website conversion rates. But if you look at the number of customers per total population, Luxembourg is 693 times better! Note that our

website and software are in English, which is an official language in India, but not in Luxembourg (where the official languages are Luxembourgish, German, and French).

And this is not a statistical fluke. These small countries all give us more customers than India:

- New Zealand (population of 4.9 million)
- Denmark (population of 5.8 million)
- Norway (population of 5.3 million)
- Sweden (population of 10.2 million)
- Switzerland (population of 8.5 million)
- Finland (population of 5.5 million)

And not to be unjust to India, there are even worse countries for our sales, where we didn't have even a single sale.

That is a harsh reality of software sales:

> The majority of English-language software is sold in the so-called "western world": the USA, Canada, North and Central Europe, Australia, New Zealand, and similar. Developing countries don't buy software. Some developed non-western countries (e.g. Japan) probably buy software but in the local language. Out of all countries, the USA is the most important for English-language software.

Because of that, I strongly suggest following this principle:

> All software we write will be in *English* and all the prices will be in *US dollars*. This targets the best market in the world and is a de facto software standard. If we need to localize, the first supported locale is the USA.

I understand when people object to this. I also thought it is unfair that I need to sell software in US dollars when I live in the EU. The first company had a pricelist in EUR, and I learned my lesson the hard way. I got a flood of emails asking me to give a fixed price quote in USD. After a year of trying to make a EUR pricelist work, I gave up and converted to USD. Emails stopped and my customers were happier. Even Iran, which hates the USA, sells their oil in USD (they tried other currencies but failed).

The difference between bad markets and good markets doesn't end with language and currency. There are many more dimensions, which all follow the general rule:

> To make money, *you need to be close to the money*. If you are far away from the money, it is a bad market.

For example, academia is many steps away from money. The research done now will become commercialized in many decades, if ever. The students studying now will hit the prime of their earning potential in 20 years. It is not fair that Albert Einstein had a smaller salary than an anonymous Wall Street broker. Einstein contributed so much to humanity, while Wall Street brokers of his time contributed to the Great Depression. But Einstein's research was many decades away from practical use, while a broker is close enough to other people's money that they can always get their broker fee. It is not fair, but such is the reality of life.

This is important because many young hackers start from a completely different perspective. Here in Croatia, they want to write software in the Croatian language. And since they are in school, the first ideas they get are connected to school activities. That includes software for school administration, making schedules, student communication, and student classifieds. That is all nice, but will not make money: Croatia is a small country of 3.9 million people, with poor purchasing power, high use of illegal software, and in the above country analysis has mediocre conversion rates. Schools in Croatia are so stripped of cash that they ask parents to donate

toilet paper (personal experience). Students will have no money for years to come. What good is your brilliant idea and code if you can't make a living out of it?

Gaming is another difficult market. Many developers want to be game devs, but to make money in gaming you need big hits. Hits get old fast, so you need to produce even more hits. I know spectacular coders in game development companies that make less money than average programmers in a local bank. They still do it because they love making games.

Developer tools are also a difficult market. Many developers want to develop their own tools and become famous with other developers. There are endless programming languages and frameworks to choose from, all open source, and maintained by volunteers. It is very hard to make money like JetBrains.

That is why I advise teen hackers to develop B2B (meaning business selling to business) or SaaS (software as a service). Businesses are close to money—they literally generate money for everything else in the economy: salaries, taxes, vendors, owner profits, etc. Businesses have a lot of problems, and if your software solves their problem, they will pay for it.

You wouldn't believe how crappy business software can be, and how much money it can make. Next time you are at a bank, shop, or admin office, take a look at the screens of employees' computers. You will easily find old, ugly software, written for MS-DOS, Windows XP, or mainframe; running in an emulator, remote desktop, or text terminal. Yet, that ugly old software makes millions. For fixing one single bug, the year 2000 bug, the US alone spent $134 billion (without inflation). But companies don't have much choice. Business software is not sexy, and they need to pay somebody to do it.

The goals of market research are:

- *Estimate how good the target market is*. How many companies are in that space and how much money are they making? How hard is monetization? How demanding are the customers?

- *Map the market landscape*. Which companies are leaders and which are falling behind? Why are some companies better than others? What clicks with customers? Is there an unfulfilled gap, where some customer demand is not satisfied?

- *Decide on your positioning*. How will you differentiate yourself from others? What will be your marketing channels and pricing? Why would someone buy your product instead of the competition?

Most ideas will fail in the market research phase, and that is fine. Actually, it is great, because it will save you years of working on the wrong thing.

Estimation

In school, you learn that only the exact answers are the correct ones. If the correct answer is 87, then 70 and 100 are wrong answers. Unfortunately, you will have to unlearn that.

We can't get exact estimates for other companies or market size no matter how hard we work. If we try to do that, we will end up in analysis paralysis. But, it doesn't matter if we misjudged the revenue of a company by two times or even four times. In startups, having an estimate in the same order of magnitude is better than having no estimate at all. The differences between markets and companies are measured in thousands, so even with an error of one zero, we will still see which market is better and which company makes more money.

What matters for market research is speed. If it takes two hours to do research, you can do it every time you have an idea or notice a cool company. If it takes twenty hours, you will postpone the research for the weekend. Then for the next weekend. And then you will never do it.

You will get a better picture of the market if you analyze 10 competitors in 10 hours, than if you analyze one competitor for 10 hours. Yes, 10 hours will lead to more precise estimates. But, with approximate figures for 10 companies, you will still be able to rank them and get a better picture of the market.

Remember, startups don't fail because they have 550 customers instead of an estimated 950 customers. They fail because they have zero customers. Orders of magnitude matter more than exact numbers.

Market Landscape

The first step is to map the market landscape: *which companies* are serving *which customers*? To do that, you need to go to the *marketplace* they use to match with each other. In the old days, that meant going to a trade fair.

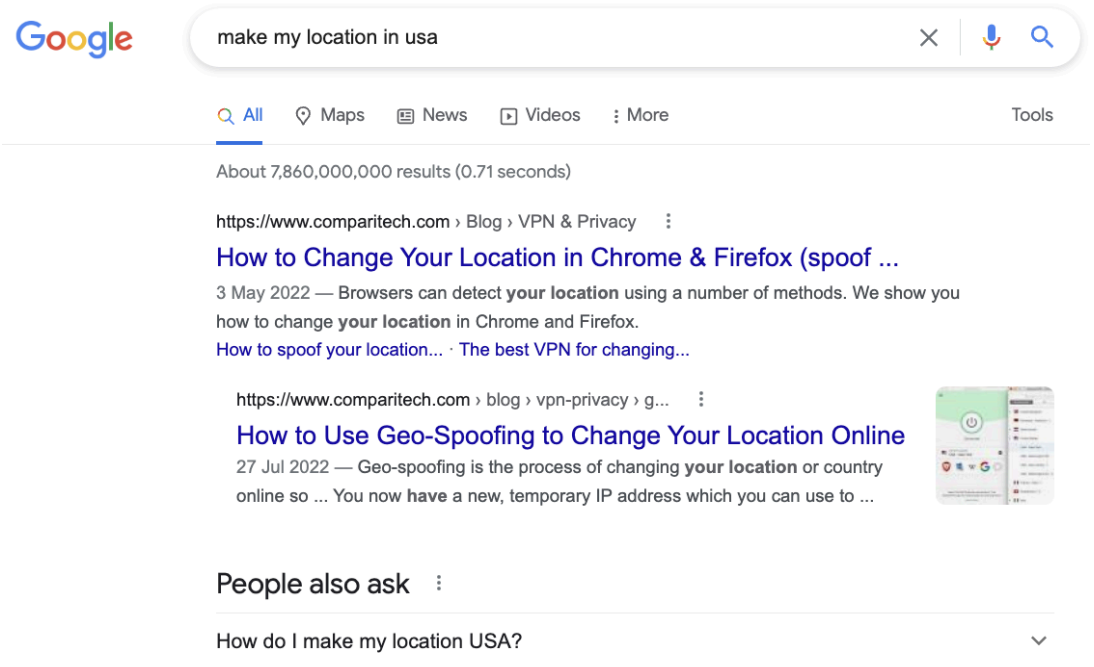
For example, if you wanted to develop a new combine harvester, you would buy a ticket to an agricultural machinery trade fair. There you would talk with both attendees (who are looking to buy a harvester) and vendors at their booths (who are selling different harvesters). You would listen to experts talk about the current trends and the future of combine harvesters. You could notice that the market is moving to self-driving GPS combine harvesters and that many customers and vendors are worried how they will switch their old machines to new technology. You could then decide to not develop the whole vehicle, but just the self-driving GPS kit that can be installed on the existing machinery. You develop the kit, rent your booth at the fair next year, sell your kit to both customers and vendors, and become rich and famous in the agricultural community.

However, visiting fairs in the old days was both time-consuming and expensive. Fortunately, those days are gone, and trade fairs are slowly dying, especially for software.

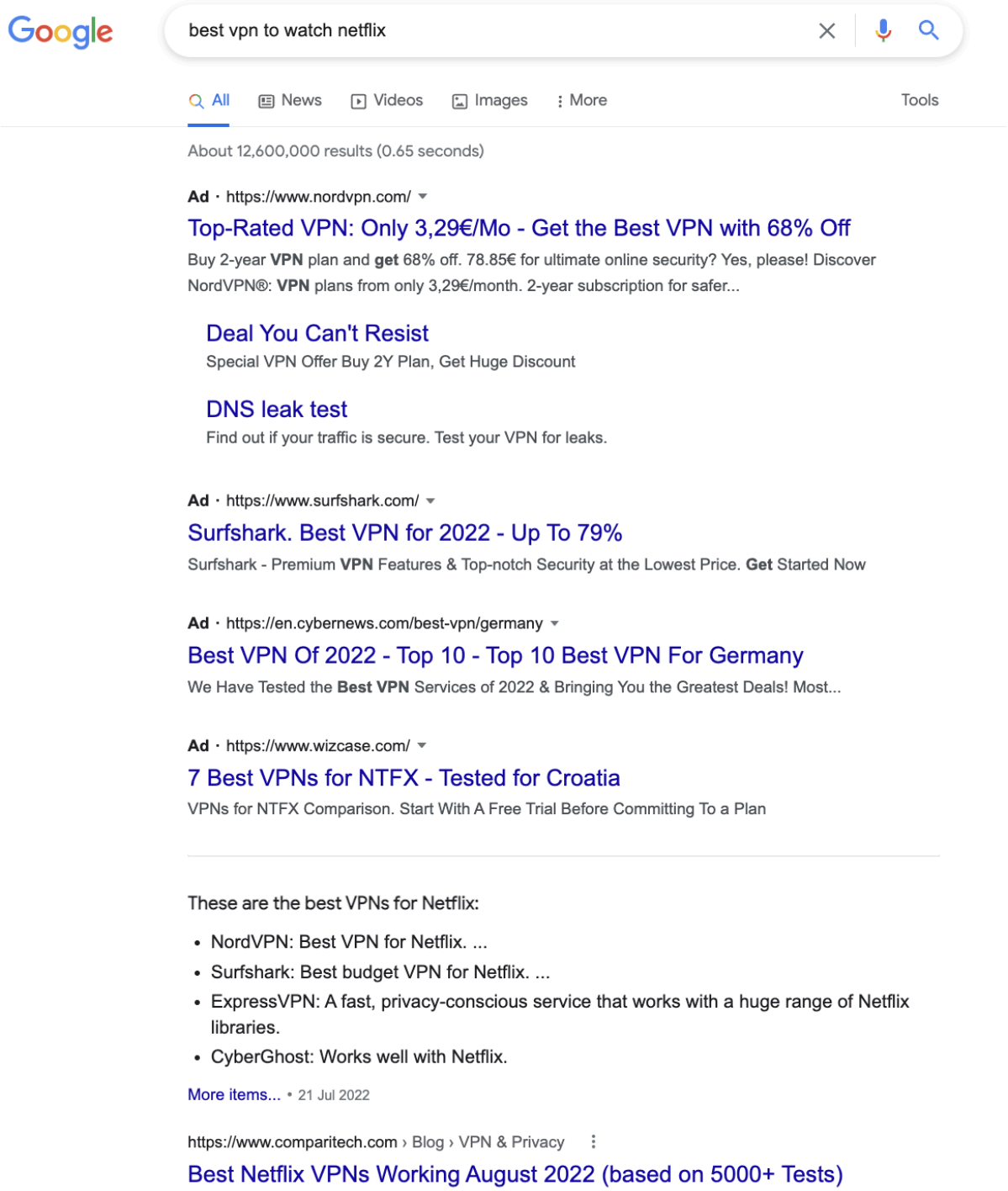
The new marketplace where customers and vendors meet is the Internet. You can research it for free from your home.

Keyword research

Start with a Google search and try to identify the keywords that connect customers and vendors. Good search keywords will show products, companies, reviews, and articles on how to solve the problem. Bad keywords will show unrelated things in the first 10 Google results. For example, the keyword "make my location in usa" doesn't connect anybody with anything:



While the keyword “best vpn to watch netflix” looks like a gold mine, with four ads, reviews, and list articles (aka listicles):



Note that if you want to see all ads, you should use VPN to teleport to the US IP, because there are more ads targeting US consumers.

To make things easier, there are many keyword research tools. Some of them are:

- SpyFu (free): e.g. GemBox summary
- Google Ads Keyword Planner (requires registration)
- Ahrefs Keywords Explorer (paid)
- Semrush Keyword overview (paid)

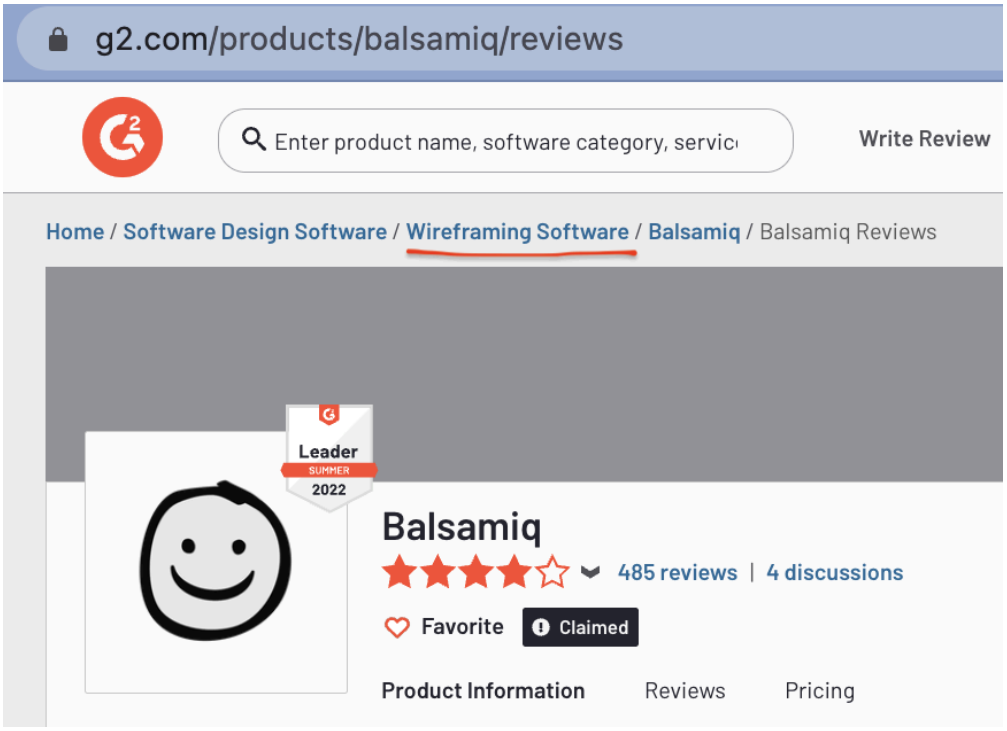
All the above tools give keyword search estimates. A good practice is to use the US as the location for every estimate, so we can easily compare numbers in the future. For some reason, all tools (including Google) give estimates that are too low, but that is not a problem because we will use estimates just for comparison.

Save good keywords to a worksheet (e.g. in Google Sheets). They should at least have a search volume column, but can also have columns for minimum ad bid, number of ads, etc.

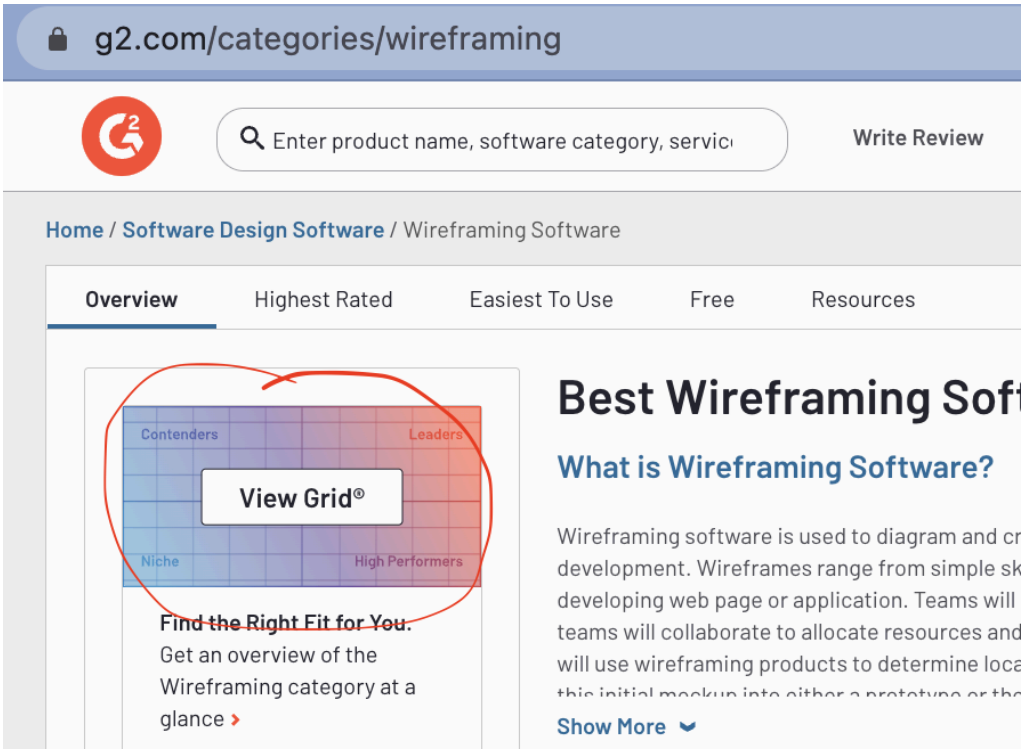
Competitors research

Save relevant companies you find to a separate worksheet. Relevant companies are the ones who offer the exact solution you are researching, or offer a *replacement* solution. For example, for the combine harvester example, relevant companies are the ones who offer GPS self-driving solution kits for harvesters, but also the ones who provide GPS self-driving solutions for tractors, because they have similar controls. Customers don't care how the problem is solved, they just care the problem is solved.

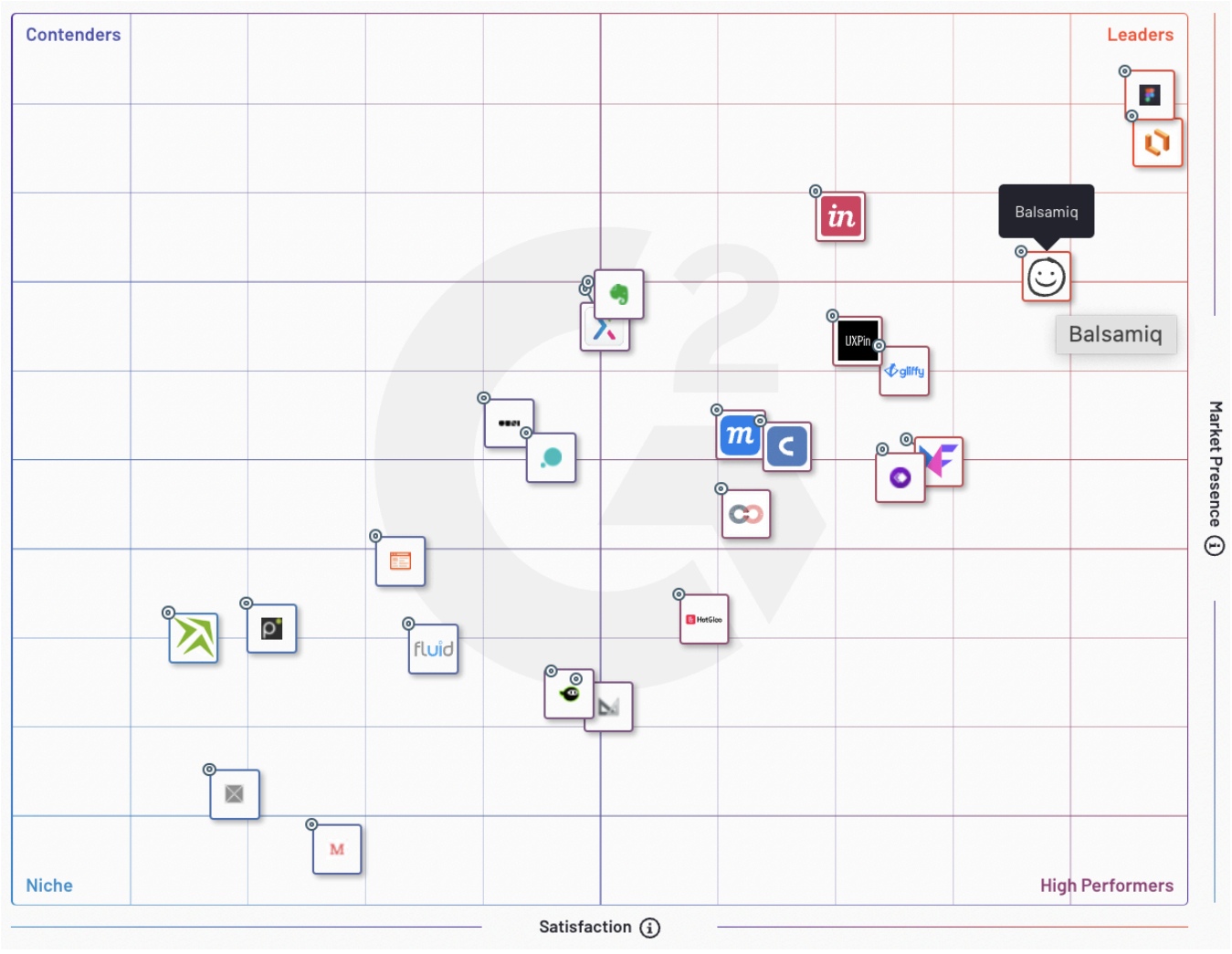
To quickly find competitors, use software comparison and review websites like Capterra and G2. G2 especially has a useful chart feature. For example, let's say you are researching a market for software mockup drawing tools. You find Balsamiq, and want to find more competitors like that. Go to Balsamiq's G2 profile, and notice that G2 has a parent category for "Wireframing Software":



Great, now you know that type of software is called "wireframing software" and that customers use that keyword to find vendors. When you go to the wireframing category page, notice the "View grid" button:



Selecting it will show a nice chart of all major players sorted by satisfaction and popularity:



There we can see that the market leader is probably Figma and not Balsamiq. Notice how fast we can get this grid with a competitive landscape. What before required going to a trade fair in Kansas or reading expensive trade publications, now requires just a few minutes of your time!

Company research

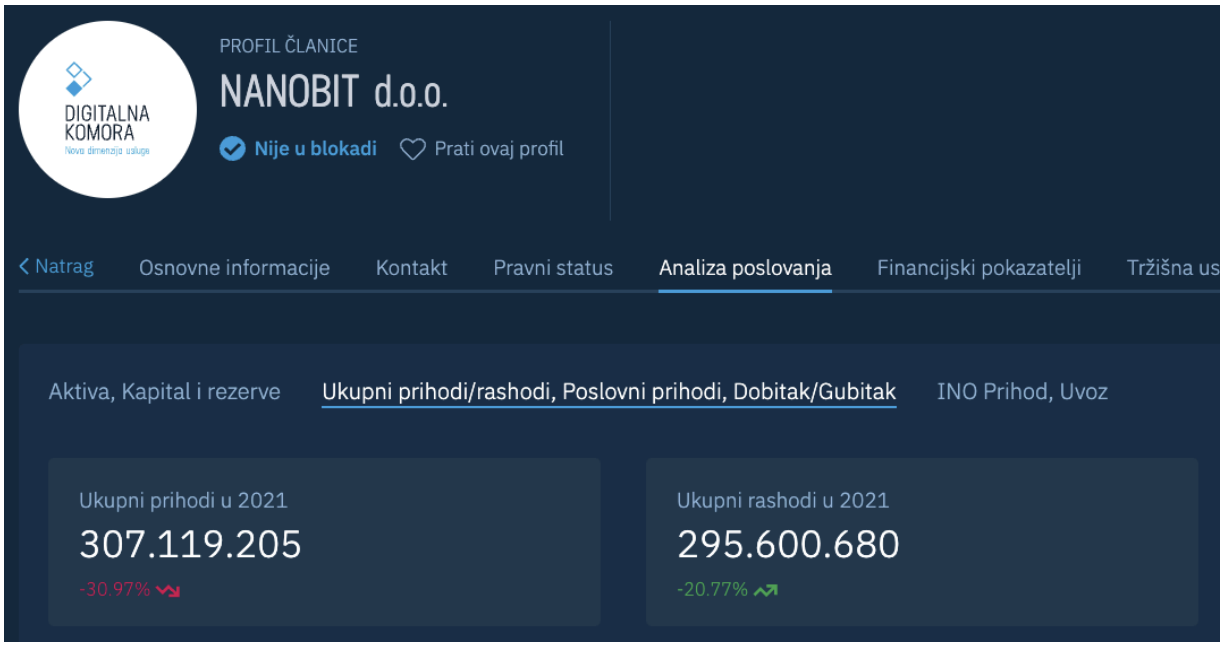
With a list of companies, we can estimate how much money both leaders and niche players make. There are a few ways to estimate company revenues:

A. If a company is publicly listed, revenue, profits, and costs are all available in annual reports. Wikipedia lists that information in an info box on a company page. For example, we know that Atlassian, the maker of JIRA, made US$2.8 billion in 2022:

Revenue	▲ US$2.8 billion (2022)
Operating income	▲ US$101.6 million (2021)
Net income	▼ US$–696.3 million (2021)
Total assets	▼ US$2.9 billion (2021)
Total equity	▼ US$294.9 million (2021)
Number of employees	8,813 (June 2022)

Unfortunately, this method is not very useful to us because as young entrepreneurs we are not competing directly with publicly listed companies. Our ambitions are smaller, and most of the companies we find will be private companies.

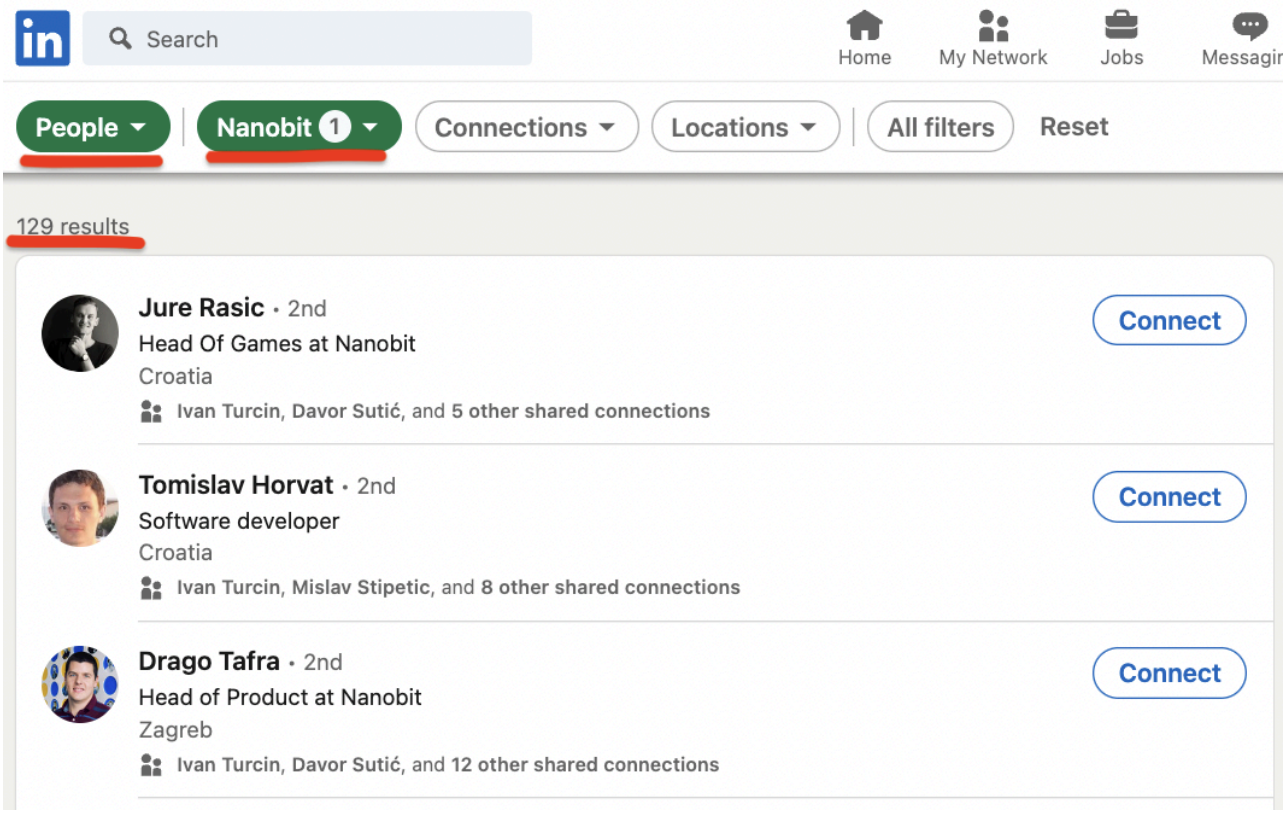
B. If a company is private but residing in a country where company tax reports are public, we can still get an annual tax report. Unfortunately, the US is not one such country.
But Croatia is. You can use either FINA or digitalna komora (both require a registration) to get a tax report for the previous calendar year. Nanobit is a popular game company from Croatia, their report shows that their revenue for 2021 was 307 million HRK (~42 million USD):



Some countries are even more open. For example, in Norway all salaries are public.

C. If a company is private and tax reports are not public, we can estimate revenue via LinkedIn. Most companies and employees have LinkedIn profiles, so we can estimate the number of employees in different countries. Multiply that with the average salary for a country and a sector (available on Google) to get an estimate of the total salary cost. For a business sector, the share of salaries in total

costs is similar between companies, so we can use that to estimate revenue.
For fun, let's do a LinkedIn estimate for Nanobit, and we can see how big the error is. LinkedIn people search says Nanobit has 129 employees, the majority of them in Croatia:



The average net salary in IT in Croatia in 2022 was ~15,000 HRK, which is 26,791 HRK gross salary, or $3655 per month. The average annual cost per employee is then around $43,860, for 129 employees that is $5,657,940. But, what is the ratio of salary costs to revenue for a typical game producer? That is a bit hard to find, but this Quora answer claims that out of $36 paid to a game producer, $10 goes for salaries. Therefore, we multiply $5,657,940 by 3.6 to get the final revenue estimate of $20,368,584.
From the previous point, we know actual Nanobit revenue was 42 million USD, so we underestimated it by a factor of two. Not bad, as we made an estimate, based on an estimate, based on an estimate, but we got a cumulative error of only two!
Of course, we can improve our numbers by estimating how many employees are not on LinkedIn, and calculating salaries for employees in different countries and in different roles. But most of the time that is not worth the effort. We just want to quickly see which companies and ideas to research more.

D. Check public investment data by searching "CompanyName investments". If there was an investment, sometimes a press release will say how much the annual revenue was. But more importantly, large investments can make unprofitable businesses look like stars. They will have nice offices, many employees, and elaborate products—but they can still go bankrupt because they don't have enough customers. We want to estimate *revenue from organic customers*. Investment is not revenue, and you can "buy" customers if you have a large investment. For example, you can offer $25 of free credits to everybody who signs up for your service (read about the MoviePass failure).

Good markets have many competitors, but each competitor positions differently, and the market is not a winner-take-all market. That means money is being spent, and customers are shopping around to see which solution suits them the best. You can be one of those solutions.

Features and positioning research

Each company uses marketing, features, and pricing to position itself in a different segment of the market. Take different car brands for example. Fuel economy and total cost of ownership are important for economy

cars, performance is important for sports cars, cargo volume is important for vans and pickups, range for electric cars, ease of parking for city cars. Some car makers go for a small number of expensive cars (e.g. Mercedes), while others go for a large number of affordable cars (e.g. Toyota). It makes no sense to have a 4x4 drive on a city car, a 500 HP engine on an economy car, or a leather interior on a commercial van. Marketing materials will tell you how the company positions a product.

The most important thing about market positioning is:

> Always try to *position in a single category*. It is very hard to position a product in multiple categories at once.

Many new entrepreneurs make the mistake of wanting to conquer multiple categories. They think their idea is so generic that everybody can use it. The problem is that even generic ideas still have a dominant target group. Snapchat is a generic messaging app, but predominantly teens use it. Excel is a generic tool, but the first spreadsheet users were accountants. Remember:

> By saying that users of your software are *everybody*, you are marketing to *nobody*.

Positioning in multiple categories is so hard, that car companies have separate brands for mass-market and for luxury cars. Even if the mass-market and luxury cars are made in the same factory, they appear as different companies to an average consumer. In the US, Honda's luxury cars are branded Acura, Toyota's luxury cars are Lexus, and Nissan's luxury cars are Infinity. VW has eight different car brands, while GM has nine.

The same applies to software. You can see how a software company positions itself by reading its marketing materials, list of features, and pricing. Sometimes it is hard to compare products just by looking at the website, because nowadays marketing materials often present benefits instead of features ("improve your sales by 20%"), give a software demo account only if you register, and hide prices behind a sales contact. Here are a few tips for research:

- When visiting a product website, look for "features" and "pricing" pages. Often a pricing page will give a better explanation of how the product is different than the homepage. But, the homepage is still important, because its messaging tells you what target group they are going for.

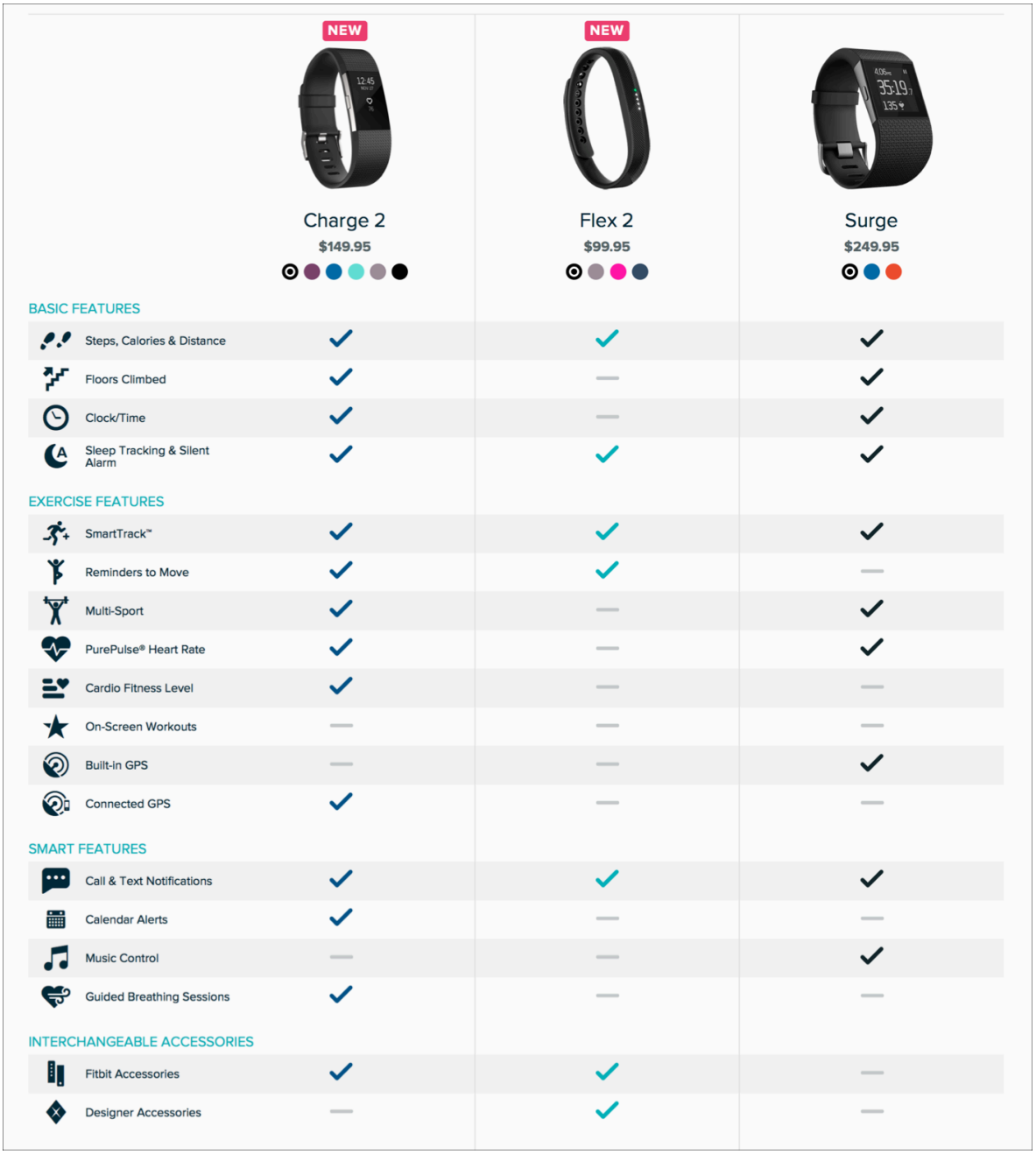
- Have a secondary email you can use to sign-up for accounts, demos, or contact salespeople to ask for prices. It is better if that secondary email is not from a popular free service (e.g. Gmail or Hotmail) because some companies reject emails from free services. You can also use your primary email to register for a service, but then prepare to be overwhelmed with promotional emails.

- If information is not listed on a website, you can often use the website's pop-up chat to get to a support person immediately.

- You can check user reviews and forums to get information on popular vendors. For example, Optimizely has a very generic homepage, saying "Unlock digital potential", and their prices are hidden. But, independent blogs will explain it is primarily an A/B testing tool, explain the features, and reveal pricing options (e.g. $36,000 a year, for 200,000 visitors or more).

- Write down your findings in a simple, structured format.

One way to document your findings is to have a spreadsheet with columns representing companies, and rows representing features, prices, marketing messages, etc. The comparison table format is popularized by many product review and software review websites (below it compares products, not companies):



For example, for GemBox.Document, we had a Google Sheet comparing different components on the market. You can also put your notes and conclusion in the comparison table.

The analysis will show where each product positions. Old market players will often have many features (sometimes too many), while newcomers will specialize in certain areas. Some companies will sell expensive licenses to enterprises, while others will try to reach small and medium businesses.

Your goal is to find an underserved niche market (a subset of the market on which a specific product is focused). This is the hardest part of market research. Estimation is routine work, but for detecting an underserved market, you need to understand the domain.

For example, Snapchat found a large, unsatisfied market of people who don't want their messages to persist forever. Their killer feature was that messages disappear, and that made the product hugely popular with teenagers.

A good example of researching and riding market trends is Amazon. Jeff Bezos had a well-paid job as a quantitative analyst at a hedge fund. But he noticed that the new thing, called the Internet, "was growing at 2300% a year". He wanted to jump on the opportunity, and decided on e-commerce, as *it was close to the*

money. He researched 20 different products to sell online and noticed the book market is *underserved*. The problem with books is that there are 100+ millions of books in the world, and any physical book store can only have a very small fraction of them. A lot of people in the US didn't live close to a large book store, so their choices were very limited. Jeff realized that without a physical store, he could also have lower prices. Books are also easy to ship in small packages. So Amazon bookstore had five things going on for it:

1. Riding an explosive Internet growth in the 90s.
2. E-commerce is close to the money (percentage of every sale).
3. Underserved market of people far away from large book stores.
4. More books than in physical stores (as warehouses fit more).
5. Lower prices (as a large warehouse costs less per unit than many small stores).

Market positioning

Over the course of years, I noticed students making the same mistakes when doing market positioning for their product. Let's cover two common mistakes.

Start with niche market, not the broad market

This seems counterintuitive. For sure, starting in a broad market that is worth billions is better than starting in a niche market worth just a hundred millions? It is not. That is because of the following rule:

> Target a niche segment of a good market where you can be somewhat competitive in six months.

For example, spreadsheet software is a much larger market than annual tax filing for dental offices. But, to develop an alternative to Microsoft Excel or Google Sheets you will need hundreds of people working for years, and high risk that users will not like it once it is released, for whatever reason. But you could develop an app to simplify tax filing for dental offices in less than six months, and it will be the best in that niche segment. You could get your first customers, get feedback, iterate your product and expand to other niches.

Many startups failed with a good idea because their ambitions were too big. With big ambitions, they need many employees. To pay those employees, they need large investments. With every investment, the founder's share in a company is diluted. Ambitious deadlines get extended, a product is developed for years, and often receives a lukewarm reception on release. Often, a company runs out of money before a full vision is realized.

Think of starting in a niche market as your advantage, not your disadvantage. Google, Microsoft and Apple can't afford to go after a small market (e.g. annual tax filing for dental offices). Every product they offer must be a hit, otherwise it will be a rounding error in their annual statement. Large companies sometimes abandon perfectly fine products that make millions of profit, because management doesn't want to bother with millions when they are focused on making billions. But you wouldn't mind a business that makes a few millions of annual profit, would you? The smaller you are, the more niche market you can target, with less competition.

Starting in a niche segment may sound like it contradicts the previous advice to always start in a good market. For example, one might say "Hungarian software is a niche market, with less competition than English software." So why develop software in English? Confusion is caused by the word "good" in "good market" and "good niche" meaning two different things:

- "Good market" is not the bigger one, but the one with better conversion rates. "Bad market" is one with bad conversion rates.

- "Good niche" is not one with less money, it is one where users are not satisfied with options (i.e. have a pain point), and you can develop a better option. Meaning your software will have good conversion rates.

My advice is therefore to develop software in English because of good conversion rates, but target a niche where you can make a difference (also for good conversion rates). But, for example, you can also start in the German-speaking DACH market (Germany, Austria, and Switzerland).

Take my companies, GemBox and TestDome, as examples of where I am coming from.

I started developing GemBox's first product on January 1st, 2005. I finished the first version, samples, marketing materials, pricelist, reseller agreement, and a basic GemBox website in six months. On June 11, 2005 I sold the first two licenses, one to Robert in the US and another to Marc in France. That is 5 months and 11 days from the first line of code to the first two customers. Niche was very specific:

> C# programmers who want to read/write XLS files using a native .NET component.

A small niche, but in a good market, because C# programmers easily spend money on components. GemBox could expand by adding features to that product, or by creating products for other file formats. If I had targeted Java programmers, it wouldn't work. Java is a bad market for paid components because Java programmers prefer open-source components. Later GemBox tried making a Java component in 2018, but that failed. Today GemBox has five components, all for C#/.NET.

TestDome started development in May 2013. We launched the website in February 2014, and sold our first testing pack to Luke in the UK on March 12th, 2014. TestDome was a more ambitious project than GemBox, with four people working on it for 10 months before the first sale. Zel was the initial investor who paid salaries for the first 18 months. Still, the niche was very specific:

> Technical people doing programming interviews in Java or C#, that prefer language-specific, short, work-sample tests.

Other platforms at the time had generic programming tasks, with academic questions about algorithms and data structures. We focused on specific language features and work-samples. From that niche we could expand by adding more questions to a specific skill, or by adding new skills. Today TestDome covers more than 90 skills.

Lower price is usually a bad competition strategy

A few students suggested a low price as a competition strategy. A long time ago, I had the same idea: "I will offer a much lower price than competition, and everybody will buy my product." Unfortunately, that is another common wisdom that doesn't work. Low price only works when customers don't differentiate by other criteria.

For example, low price is a good strategy for commodities: copper, oil, corn, rice, etc. By definition, commodities are treated "as equivalent or nearly so with no regard to who produced them." Because of that, customers buy the cheapest commodities. If you find a way to produce corn 20% cheaper than everybody else, congratulations, you are going to be a billionaire.

Customers will also differentiate by price if they are price-sensitive or if they can't see a difference.

For example, airlines clearly don't offer the same service. Some airlines offer free check-in luggage, have larger carry-on baggage allowance, offer meals, have better service, better in-flight entertainments, bigger space between seats etc. But, when you are searching for a flight, you don't see all that. Most prominent are flight time, duration and price, and you choose by that criteria. Even if advanced search is available, most travelers are price-sensitive and just want to get from A to B. Low-cost airlines dominate this segment. No matter how much travelers complain about Ryanair, they still fly it because it is cheap.

It is interesting that for business travelers price is less important. They learn which airlines offer better service and they are not price sensitive because their company pays a ticket. Frequent business travelers pay more to fly on favorite airlines, in business class, and have tickets that can be easily rescheduled. As a result, despite business travelers being only 12% of passengers, they generate "as much as 75% of profits"!

In short:

> Most software is not a commodity, and the price is not the primary differentiator for most customers.

E.g. customers buy Windows despite having free Linux, Excel despite having free OpenOffice.Calc, and Photoshop despite having free GIMP.

Most software is not a commodity because:

A. Software products are *inherently different*. E.g. professionals working in Photoshop can't easily replace it with GIMP.

B. Most of the cost in software comes not from the software price, but from *time needed to learn and customize the software*. E.g. it takes years to learn Photoshop, Photoshop plugins mostly don't work with GIMP, and GIMP has limited support for existing PSD files.

C. Software products are *sticky*. Once you learn something and you make it work, you don't want to switch. E.g. some large organizations are still using COBOL after 60 years.

D. Software often has a *network effect*. The more users software has, the more valuable it becomes. Microsoft Office is valuable because everybody knows how to use it and every PC can open Office file formats. The value of a social network is proportional to the number of users.

When I was a student, I thought software should be cheaper, because I didn't have money and I had plenty of time. But I was not the typical software buyer—I was a student using free or illegal software. Now I understand that a typical software buyer has money and they want their problem solved with low effort on their part.

There is another reason why VC's and startup mentors don't like "being cheaper" strategy—it shows that you lack other differentiating ideas. "Being cheaper" is a generic strategy that you can stick to any software, from spreadsheets to mobile apps. If you did your homework and analyzed the market, how come the best insight you have is to be cheaper? Knowing what exact problems your customers need solving always sounds better in a pitch.

There is a more philosophical explanation why you should choose to work on software with a high profit margin. As a teen hacker, you probably want to make an impact in the world. Profit margin is capitalism's way to tell you which projects have the most impact. If the problem is already solved, there is a very slim profit margin (e.g. email hosting). If the problem is not solved, the profit margin is high (e.g. cure for cancer). Although "big profits" has a pejorative meaning in common language, it often means you are solving a large problem for many people (unless you are scamming them, which we will never do). Of course, you can decide to dedicate your life to the benefit of mankind and work only on open-source software. Then just replace "high profit margin" with "high usage" and this section will still hold true. Why work on open-source that benefits hundreds when you can work on open-source that benefits millions?

But, if you want to be in a market where price matters, you can still find such markets in software. In my modest opinion (without first-hand experience) these two are similar to commodity markets:

- *Gaming market*. Of course, every game is unique. But most games solve the same problem: *how to be entertained for X hours*. A gamer can spend their free evening playing a first-person shooter, a

real-time strategy, or even watching a movie. Because different ways of entertainment are interchangeable, entertainment is often the winner-take-all market. In other words, everybody plays the same hit games, while 90% of other games lose money. On top of that, game buyers are price conscious and often wait for seasonal discounts and bundles. The pressure to be cheaper is so high, that most new games use a free-to-play model.

- *Outsourcing market*. Most companies decide to outsource software development to other countries *exactly because it is cheaper*. Outsourced developers are often treated as commodity, as a head count (e.g. “two senior and three junior developers”). Because developers are treated as non-differentiable commodity, in a war to discount hourly prices, winners are countries with low average wages (e.g. India, Philippines, or Ukraine). If you want to discount your work, be aware that is the market you need to compete in.

For each Electronic Arts in gaming and Infosys in outsourcing, there are many other gaming and outsourcing companies that barely make any profit.

Homework: market research

Now that you know what makes a good idea, a good market, and how to do basic market research, it is time for a practical exercise.

For each task, make a single Google Doc in English. If you also need Google Sheets for tables, embed them in the main document, or link to them from the main Google document. Then share the finished documents with Zeljko.

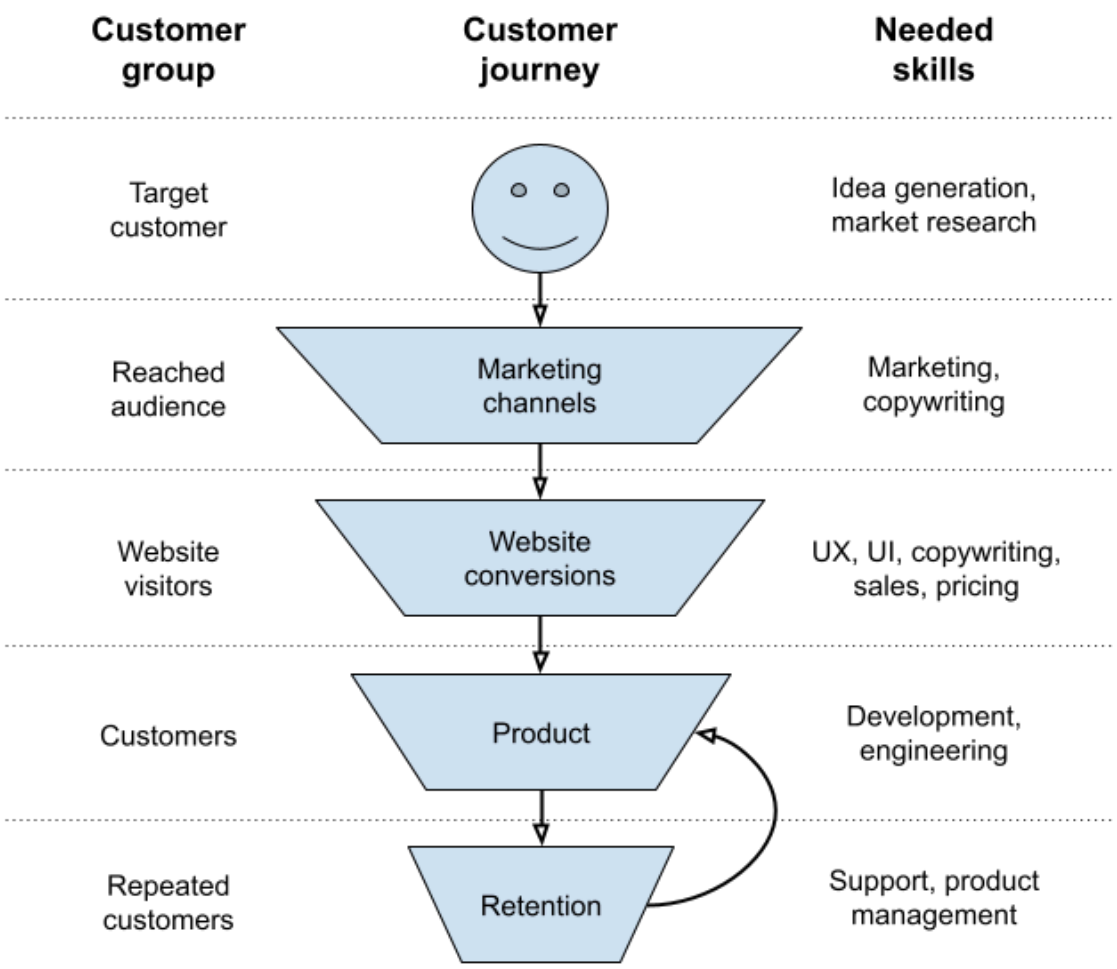
Homeworks is given in separate documents. Zeljko will give you more details.

Marketing

Hacker, hustler, and hipster

Saying that “a successful startup needs a hacker, a hustler, and a hipster” is both funny and packed with wisdom. Let's unpack it.

More than 95% of startups fail. Startups are hard because completely different skills are needed in order to succeed. Take a look at a typical software customer journey and the skills needed to execute each step:



The steps of the customer journey are similar to links in a chain. “A chain breaks at the weakest link” is also true for startups:

> A startup will fail at the weakest part of a customer journey—often the part that founders understand the least.

It doesn’t matter that your software has great code and engineering behind it if the idea doesn’t stick with target customers (e.g. Google Wave and Google Stadia). It doesn’t matter that your marketing is great if your product doesn’t work (e.g. Theranos and Nikola Corporation). It doesn’t matter that your website and product are great if users churn after the novelty wears off (e.g. Quibi and AirTime).

“A successful startup needs a hacker, a hustler, and a hipster” because that roughly covers needed skills:

- Hacker: development, engineering, technical support
- Hustler: market research, sales, pricing
- Hipster: marketing, copywriting, UX, UI

I guess that some of you are thinking: “Wow, that is a lot of different skills! Let me develop a product first and then I will find a cofounder who is an expert in other domains. Or I can outsource that to a marketing agency, a design agency, developers in India, and support agents in the Philippines.” That is what I thought when I started my business. But, that doesn’t work.

Specifically, late cofounders don’t work because:

- It is hard to find an expert in complementary skills. Your friends and social circles are probably similar to you.

- Even if you find an expert in other domains, it is hard to convince them to become a cofounder. They don't know you, it is not their idea, and they didn't have any say in a product you created.

And agencies don't work for startups:

- It is hard to find an agency that is good and will care about your product. Agencies juggle multiple projects at a time and prefer to deliver standard service to different clients. Agencies rarely wait for analytics data to do incremental changes, hypothesis testing, and pivots. As soon as the project is paid, they will move to another client and forget about you.

- Even if you find a good agency that will care about your product long-term, that will cost you an arm and a leg every month. Read this quora answer about retainer fees for advertising agencies. Prestigious agencies have rich corporate clients, not struggling startups.

- You can't just add marketing or design to an existing product like it's a side dish. Design is part of the product, and a marketing strategy decides what features are important.

I know that the idea that you need to own the entire customer journey is a hard sell. If you are a techie, the ambiguity and impreciseness of marketing, copywriting, and UI looks scary. But actually, you are more qualified than you think.

The customer journey is a chain which breaks if any link breaks. What other things are like that? Mathematics is—a solution for a math problem must be correct in every step. Programming is—one wrong line of code and a program will crash. It doesn't matter if other 9999 lines are correct, the program will not work. That is why mathematics and programming are difficult for most people. If you are good at mathematics and programming, you can be good at building companies. You already have analytical skills like:

- Breaking down a complex problem into smaller steps.

- Analyzing which steps don't work.

- Using data to make decisions.

- Making a chain of logical conclusions.

- Working on parts of the problem that make the most impact (aka Pareto principle).

The difference is that programming has an immediate feedback loop: run a program on your computer and see immediately if it works. In marketing, copywriting, and UI, feedback takes more effort and time. You will need to get feedback from people or wait for a few months to get enough analytics data. But the feedback loop is still there.

I will share my experience in the hope that you will learn from my mistakes. Over the years I tried to outsource parts I didn't want to do and I failed miserably. For example, when Mario and I started TestDome, we decided to use outside agencies for SEO and for website design.

For SEO, I found both a freelance expert and a full-blown SEO agency. Both were recommended to me, they were present in the local community, and they left a professional impression. We paid them to write separate SEO articles hosted on 3rd-party domains. After their work was published, I was not impressed. Articles sounded generic and they were on websites full of generic SEO articles. Still, they were the experts, so we

paid them and waited for the "link juice" to come. And boy, did it come! One day Mario called me and said "We have a problem: we are not getting *any* visitors. " It turned out that we got a Google ban for using prohibited SEO practices (aka. "Black hat SEO"). We had to manually disavow every link those "SEO experts" created, in order to be shown on Google at all. We complained to both of them, but each blamed the other one. I found that funny, because the articles were so similar I needed to consult an excel sheet to know which article was written by which "expert". Neither of them offered to return their SEO consulting fee.

For website design, we chose a design agency with a great reputation and where we knew the founder. This experience was much better, and we ended up with a website design we really loved. But, as a startup you change and test things all the time, so you need new designs every month. The design agency doesn't want to stop the current project to work on a single page, so we batched multiple designs every few months. After a few years, the founder sold the agency. The new boss told us that they are busy with large American clients, and will have to stop working with small clients like us. We lost our designers although we paid each of their invoices immediately. Money was not the issue, they just didn't have resources for small and infrequent design tasks.

But there is an upside to both stories.

The TestDome SEO debacle forced us to think about SEO and put all our effort into it. Today SEO is an integrated part of the TestDome product. We have landing pages for different tests, skills, and publicly available questions. Our content creators need to think about SEO before creating new tests. The product team has an alert warning if a landing page's loading speed falls below Google's proposed guidelines. None of that would be possible if SEO was outsourced.

Design is now done internally, by a young employee passionate about Figma. He is available every day for advice and small design tasks needed in separate development tickets. The product team created a design book specifying the TestDome design language. I became experienced in UX and often create mockups of required pages.

That doesn't mean we don't outsource. We outsource payment processing, accounting, legal, servers, server maintenance, and event organization. We don't need anything creative there; quite the contrary. We want our accounting and legal documents to be boring and standard. But we control our customer journey.

When to think about marketing

If I convinced you to become a hipster and learn marketing, the question is when to think about marketing? After the product is done, in parallel with product development, or before developing a product? Let's take three different perspectives:

A. *Risk perspective*: We want to reduce the risk of startup failure. Since we are hackers, it is unlikely technical product development will fail. We can also develop a website, create some UI, and answer support questions. But we don't have a clue about marketing, so that is the most likely thing that will kill our startup. We should then work on marketing first, because *if marketing fails there is no point in other activities*.

B. *Feature set perspective*: Before we build a product, design screens, and create documentation, it would be useful to know what features we need. How can we know what features users want? One way is to test different marketing messages first, and *see what features users want to buy*.

C. *Idea validation perspective*: There is a risk market research was wrong and our idea is not worth a penny. What is the fastest way to invalidate an idea? Create marketing materials and try to sell it before you build anything. If you can't find customers, then you can *abandon a bad idea early*.

Three different perspectives, but the same conclusion: *you should do marketing before building a product*.

Unfortunately, most new founders ignore that advice. That is a well-known problem in startup incubators, accelerators, and corporate entrepreneurship. Most mentors say that the startup path should go like this:

1. *Ideation*: create many ideas
2. *Market research*: research each idea, choose the most promising one.
3. *Market validation*: create marketing materials, find target customers, create a mailing list of people who want to buy the product when released. If market validation fails, go back to previous steps.
4. *Minimum Viable Product*: Build an MVP, test it with interested customers, confirm the basic feature set.
5. *Full product*: Develop the full product and charge reasonable money for it.
6. *Scale up*: scale by bootstrapping or by getting investors. Getting investment is easy when you can show that real customers love your product.

Instead, as a startup mentor, you often see this:

1. A founder comes with a single idea that they "believe in".
2. Market research was superficial and dismissed alternatives with "not the thing I had in mind".
3. The founder needs an investment to develop the idea. Since they don't have any market validation, finding investments is hard and the investment terms will not be favorable.
4. The founder and new employees work on the product for years and need more investment.
5. The startup releases the product and then they start to think about marketing.
6. As marketing is not integrated in the product and there is no user base, the startup decides to buy advertising. With standard ad conversion rates for Google Ads, Facebook, or LinkedIn the customer acquisition cost (CAC) is high (e.g Casper CAC is $290 per mattress buyer).
7. High acquisition cost doesn't matter much, as most customers churn anyway.
8. All investment money is spent and the startup goes bankrupt.

It is very sad to see this story playing out again and again. Especially when it doesn't need to be this way.

Many smart people thought about how to solve this problem. I will present three frameworks, in no particular order. Although being different, they all try to do the same thing—force founders to think about the market and marketing before making a full product.

Crowdfunding

Crowdfunding is great for consumer products which have some novelty factor.

The biggest crowdfunding website is Kickstarter, followed by IndieGoGo. They operate on the same principle:

> Crowdfunding converts a customer product idea to a series of marketing materials and promises, that multiple backers preorder and receive once produced.

Marketing materials and promises include product videos, images, specifications, pricing, claims of effectiveness, early-access testimonials, etc. Some crowdfunding websites say a “working prototype” is needed to create a crowdfunding campaign, but in practice projects got founded and failed without a working prototype. For crowdfunding to succeed enough backers need to fund the project, and that becomes the initial capital.

Crowdfunding is great because it forces you to think about marketing, target customers, features, and pricing before developing the full thing. If customers crave the product, they will back it. If they don’t, you will know that before borrowing money from a loan shark.

Zeljko knows founders of two successful kickstarter projects. One is the origami folding boat that received $443,806 in backing. Another one is the Uprising board game that received $324,441. An example from Croatia is CircuitMess by Albert Gajšak that received $408,749. All those founders said that Kickstarter was the key to their success. And they are still complete owners of their companies.

Unfortunately, crowdfunding rarely works for software. And it definitely doesn’t work for B2B software, which is not as sexy as gadgets and physical products.

If you still decide to use crowdfunding, find articles and videos from people who had crowdfunding success. It is not as easy as it may look. It is recommended to build an initial community before launching a crowdfunding project. That initial community will help you refine the marketing materials, and will be ready to found your project once it is live. For example, a friend with the Uprising board game had a mailing list of more than 1000 people before Kickstarter launch.

Lean Startup

LS is the most famous startup framework and is useful for different product types. The Lean Startup book is the startup bible, you should read it. The revolutionary idea from that book is that the goal of the startup is *not* building an actual product. In Eric Ries’ own words:

> “The goal of a startup is to figure out the right thing to build—the thing customers want and will pay for—as quickly as possible.”

In Lean Startup terminology, “figuring out the right thing to build” is called *validated learning*. Validated learning can come from different sources like:

- Talking to target customers.
- Creating a website with a marketing description, generating traffic, and seeing how many visitors hit the “Buy” button.
- Pretending you have a product but using humans to do the work in the background (known as the Wizard of Oz technique).
- Creating a minimum viable product (MVP) that you can test with target customers.

Unlike crowdfunding, where a crowdfunding website promotes your project with potential customers, with the lean startup you need to find target customers yourself. That makes it harder but more flexible—you can find your target customers wherever they usually hang out: internet forums, subreddits, business fairs, meetups, google searches, LinkedIn, Facebook, Instagram, etc.

In order to validate your hypotheses you will need to find hundreds of target customers to talk to or run experiments with. What if you can’t find hundreds of target customers? Then you should definitely *not* develop the product, because you will not be able to make money out of it. If your goal is to sell 100 licenses but an

average visitor conversion rate is 1%, you will have to find 10 000 target customers. If you are not able to even find 100 customers to do quick research, how do you expect to find customers?

Amazon's Working Backwards

AWB framework originated from Amazon as a way to build new products. A usual corporate product development process would start with implementing the product, then creating marketing materials, ending with a press release and support documents. Jeff Bezos noticed problems with that order:

- It is company-centric to start with development first. In the development phase a company is inclined to cut corners and reuse existing solutions that are not optimal for customers. Instead, the process should be customer-centric, and it should start from customer needs.

- Project goals are not clear to all employees involved. Even developers are not always clear on which features are necessary and which are optional.

- The product team often develops features that the marketing team doesn't find marketable, and lacks features that would be great for marketing.

- If the price is left unspecified, the product team will often create a product that is too expensive.

Instead, Jeff asked teams to work backwards, starting from a target customer. The first thing a team needs to create is a press release for the fictitious product. Industry practice is that press releases should be 300-400 words, not longer. Since that limits a product description, only key features need to be listed. A press release also explains use cases, target customers, price, and availability. After approval by management, the entire team reads it and knows what they are working on. The next thing developed after a press release is an internal FAQ (frequently asked questions) that explains all things team members need to know but are not in the press release, for example: which services, resources, or hardware is needed, people involved, minor features, architecture, timeline, budget, edge cases, etc. The purpose of the FAQ is that all team members understand the details and commit to it. After the FAQ is approved, the product development starts.

For example, when Kindle 2 was being developed, Jeff insisted that the press release includes "Whispersync": an ability to wirelessly sync books, bookmarks, and reading progress over a GSM network. Previously, one needed a cable, a PC, a sync application, and an internet connection to manually move books to Kindle. With Whispersync, purchased books would magically appear on your Kindle after a purchase. And the press release stated Whispersync will be free with every Kindle. Jeff and the marketing team loved it. The internal FAQ explained that Whispersync will use Whispernet, Amazon's custom always-on 3G connection. That put a great burden on the product team. They needed to add a 3G modem, negotiate prices with network carriers to have an affordable 3G plan, incorporate the cost of 3G in the price of Kindle and books, and develop new syncing software. If the press release didn't insist on that feature, employees would be inclined to do what is easiest for them: keep the existing cables and sync software. The fictitious press release made solving that customer pain point obligatory. When Kindle 2 was released, journalists were delighted to discuss the new revolutionary feature.

Summary

All above frameworks have one thing in common—they are customer-centric. Customer needs are discovered via conversations, hypothesis testing, mockups, videos, press releases, etc. Product decisions are based on customer needs.

In contrast, starting a project by writing code is self-centered. It is easier for developers to write code than to write press releases. And it is a chance to learn hot new technology! But it is often disconnected from customer needs.

Being self-centered is a big problem in startups. Many think they know what customers want, and proceed with their vision even when market research shows them otherwise. I was certainly guilty of being self-centered when I developed some early products that flopped (see About the author).

The customer-centric approach works because, for most software startups, the biggest risk of failure comes from customer adoption.

But, if something else is the biggest risk for your startup, focus on that instead. Other risks include technology, legal compliance, cost, safety, etc. For example, customers loved Clear startup where frequent flyers paid $200 per year to skip airport security lines. But the Transportation Security Administration (TSA) never approved it, so their customers needed to wait in the same queues as everybody else. As another example, if you are in one of the Hyperloop startups, the biggest risks are probably technology, cost, and safety, not customer adoption.

For the rest of us, it is best to work within a customer-centered framework.

Online marketing

Marketing is a huge field that would take many books to cover. Fortunately, if we focus on software, we need just digital marketing. And we will cover just marketing channels that work for small software companies. Even better, our learning will be useful for marketing but also for market research and customer validation.

Such marketing focus seems counterintuitive. Ask an ordinary person to give examples of marketing and they would probably mention TV ads, newspaper ads, radio ads, billboard ads, team sponsorships, mail flyers, conference sponsorships or promotional email. For us, all those channels are completely irrelevant, because they are all mass-media advertising. They are good if your product has mass appeal, like Coca-Cola, but bad if you have specialty software. More generally:

> Marketing reach is useless if it doesn't find *your* target customers.

It doesn't matter that your TV ad was seen by a million people, if none of them are your target users. People ignore or forget ads that don't offer a benefit to them. An average person in the 1970's would have seen between 500 and 1600 ads per day. Today we see between 4000 and 10,000 ads per day. Advertising is everywhere: on every newspaper page, in every TV show, in most google searches. Even some of my socks have logos on the side so that others can see where to buy such lovely garments.

Now make an experiment: write down all ads you saw yesterday. How many did you remember? 5, 10, 20? Even if you remembered 7 out of an average of 7000 ads, that means you remembered only 1 in 1000. Modern humans have an incredible talent to ignore advertising. On the web, it is called banner blindness. Eye tracking studies showed that the brain unconsciously ignores everything that looks like a banner and positions eyes directly where the content is. Effective ads get noticed and acted upon because they are relevant to you. Ineffective ads can get millions of views and still get zero sales. For example, a 2008 study found that spammers need to send 12,500,000 spam emails to get a single sale!

Why do ad conversion rates matter? Because:

> Ads are directly or indirectly charged per number of views.

Advertising networks often claim they charge "per result", for example Google or Facebook charge you "per click" (aka. pay-per-click or PPC). But, if you dig deeper, cost per click (CPC) is higher if your conversion from views to clicks is low. That is because Google or Facebook algorithms maximize the money they can get from 1000 views. For example, an advertiser A with 10 ad clicks for each 1000 views will bid $2 per click, while an advertiser B with 1 ad click for each 1000 views will bid $20 per click. Both A and B pay for "the result", but after math is stripped away they both pay 2 cents per ad view.

Same applies for all other paid marketing channels. TV ads are expensive in prime time and cheap late in the night. Radio ads cost more if the station has more listeners. The most expensive print ads are on the front page.

The numbers are quick to “add up”, because they actually “multiply up”. For example, let’s say that you pay $2 per click which leads visitors to your page. Out of all visitors, 4% sign up for a free trial. Out of all trial users, 5% decide to purchase your product for $100. Your customer acquisition cost (CAC) is therefore $2/(0.04*0.05) or $1000. Congrats, you lose $900 on every customer you get via ads, and you made a perfect money burning machine.

The above example is not exaggerated or unusual. As mentioned before, even Casper that sells a mass product like mattresses has a CAC of $290. That is nothing. For enterprise software, CAC can go from $2,190 (eCommerce) to $14,772 (financial software).

That is why ads are dominated by mass market products (e.g. drinks, detergents, telecoms) or by expensive products with high profit margins (e.g. cars, watches, finance products). Mass market products have high conversion rates as everybody uses them, while products with high profit margins can cover the high cost of acquisition.

Your specialty software only has a niche appeal and you probably can’t afford to pay $1000 of ads per customer. What to do then?

We can solve that problem by using few marketing strategies:

- Pay-per-click (PPC) advertising
- Content marketing
- Search engine optimization (SEO)
- Growth hacking

That is because they all allow *niche targeting*, something mass marketing channels can’t provide.

Niche targeting

For niche software, what is the most important advertising skill? Design? Copywriting? Nope, it is niche targeting.

Targeting is finding people who are your potential customers. In mass media, targeting is very wide and unsophisticated. Usually, the market is segmented by demographic parameters (age, sex, income, education level etc.) and location (city, state, country etc.). For example:

- TV: Soccer is mostly watched by men, so ads in breaks target men (beer, cars, etc.). Daily soap operas are mostly watched by women, so ads in breaks target women (cosmetics, household products etc.). Fun fact: soap operas were created by soap manufacturers so they can target housewives.
- Radio: Local radio stations can serve ads to the local population, e.g. local business ads.
- Newspapers and magazines: Different magazines have different target audiences. Financial Times is read by business people, Runner’s World by runners, and Better Homes and Gardens is read by homeowners. By buying an ad in a magazine, you are targeting your ad to that specific audience.

That targeting sounds good, but is actually too broad for most products.

For example, let's say you are selling car bike racks. You could spend millions on TV ads displayed alongside Tour the France. But, most viewers aren't looking to buy a car bike rack right now. Some of them don't even have a car or a bicycle, and are just watching for fun. Serious cyclers often already have a car bike rack. France Télévisions claims 41.5 million viewers watch Tour de France. For comparison, the population of France is 65 million. So it seems that just about everybody watches it, regardless of whether they are bike enthusiasts or not. What is the probability of a viewer being a car owner, and a bike owner, and in need of bicycle transport, and didn't buy a car bike rack yet, and decided to buy it that month while watching Tour de France, and noticed your ad in a sea of many? Probabilities multiply up and I would bet less that 1 in 10 000 viewers are your target customer. But, as we said before, you pay the ad per number of views.

As an alternative, you could go to Google Ads and bid on the "best car bike racks" keyword. After typing that keyword into Google, your ad will be displayed next to search results. You know that the person clicking on your ad is a warm lead: they showed interest. But, you could target that ad even more. "Best car bike racks" is a somewhat generic keyword one would type at the beginning of a customer journey. As one learns division of mounting systems, they will refine their search to "towbar bike racks". Less people search for that keyword, but they are "warmer" leads, as they are further in the customer journey and more likely to buy. As one learns such racks are expensive, they can search for "towbar bike racks below X eur". Or they can search for "towbar bike racks Birmingham" if they want to find a local shop today. Guess what, there is a company targeting just that niche: Towbar Services Birmingham.

The internet revolutionized advertising targeting. Companies can now target ads in ways unimaginable before. I know one company owner who sells wedding planning software. Owner bought ads on Facebook, targeting people who just changed their relationship status to engaged:

Targeted advertising is how Google and Facebook make their billions. In 2020, ad revenue was 73.3% of Google's profits and 98.5% of Facebook's revenue. Businesses don't want to go back to targeting by sex and age when they can now target customers thousand times better.

Targeting has a counterintuitive relationship with ad design and copywriting:

> The better your targeting is, the less you need to think about design and copywriting.

If you are creating a TV ad, you need to think about every second, every word, every detail. Fast food commercials cost hundreds of thousands of dollars. But, you can recognize a McDonald's commercial in

seconds and tune out. We instantly recognize ads and ignore them. Compare that to receiving a message from a friend saying “hey, wanna grab macky d's tonight?” Your friend didn’t bother copywriting, he didn’t even bother capitalizing words. But, you are more likely to join your friend, because he targeted that question directly to you. That is why word-of-mouth marketing has the highest “conversion rate”—you are more likely to listen to your friends than to advertising.

That created a significant shift in how things are advertised. Before, ads were boxed audio or video segments that would interrupt a radio or TV program. But people quickly learn to ignore ad breaks. Instead, now advertisers find influencers and tell them to describe the product in their own words. If your favorite YouTuber says they love product X because of Y, that connects to followers better than a 30-second ad break. That is called influencer marketing. Influences need to use colloquial language when endorsing. If they start reading professionally written copy, you will notice these are not their words and trust them less. Crafted copy and professional design are a signal it was paid for and not genuine. Two other related strategies are native advertising and product placements. In both of those cases, advertising tries not to be recognized. Native ads look like ordinary content. Product placements use branded products in entertainment content like movies or series.

All that can be bad news if you are a designer or copywriter in an ad agency. But it is great news if you are a hacker with a low budget. Niche targeting is a technical skill. And once you find your target customers you don’t have to make a huge splash. Just tell them, in your own words, “To solve problem X, use product Y.” If your product really solves that problem, they will buy it.

Pay-per-click (PPC) advertising

The biggest advantage of PPC advertising is that *you can target many different niches very fast*. It only takes a few days from entering keywords to seeing the first statistics. Because of that, PPC advertising is great for testing keywords, ad messages, testing MVPs, finding your first customers, and iterating fast.

The biggest problem with PPC advertising is that the *cost-per-click (CPC) can be prohibitively expensive*. It was not always like that. Before PPC got popular, CPC could be as low as $0.15. In the last two decades, businesses discovered PPC advertising and started outbidding each other. There are more search ads now, users ignore them more, and conversion rates dropped. One analysis says that in 2006 the average CPC was $0.32 and the average conversion cost was $7.63. Ten years later in 2016, the average CPC was $2.16 and the average conversion cost was $33. Note that these are often “soft conversions” like signups, downloads, or email leads. If the cost of every signup is $33, but only 5% of signups convert to buyers, then the actual customer acquisition cost (CAC) is $660 ($33/0.05).

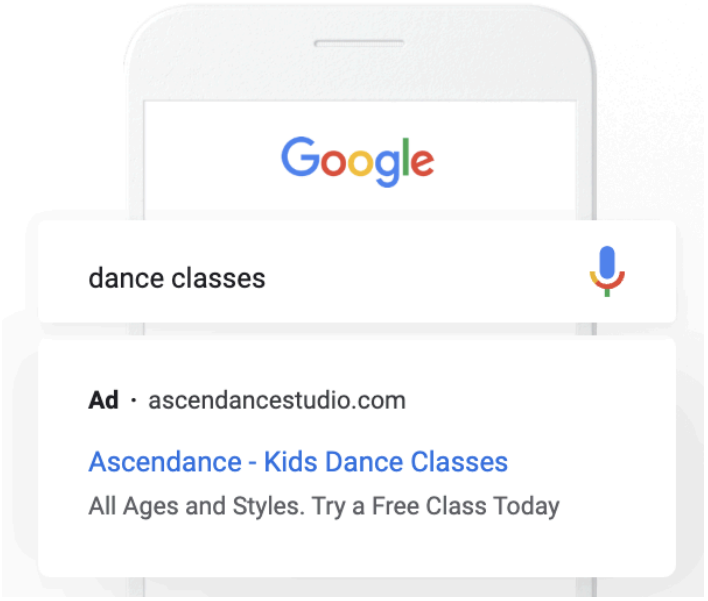
However, even if you are losing money on PPC advertising, I would recommend starting with a PPC campaign first. Why? Because you will be able to test many different ads, keywords, price points, and landing pages in a few months. PPC advertising is a fast iteration machine, perfect for the lean startup approach. Let’s say you lose $100 per customer. The first 10 customers will cost you $1000. The amount of information you will get from your first 10 customers is invaluable. If you do nothing else, you will sleep well knowing that people want to buy your product. The 2nd most ecstatic day I had in my life was when I sold my first two licenses (the 1st most ecstatic was, of course, the day my daughter was born).

After you learn what works using PPC advertising, use that knowledge to create long-term marketing, such as content marketing, SEO, or growth hacking.

Now, let’s get into the details of PPC platforms.

Google Ads

The biggest player in the PPC space is Google Ads (formerly Google AdWords). It’s easy to get started. Open an account with Google Ads, add a credit card, and start adding keywords and ads:



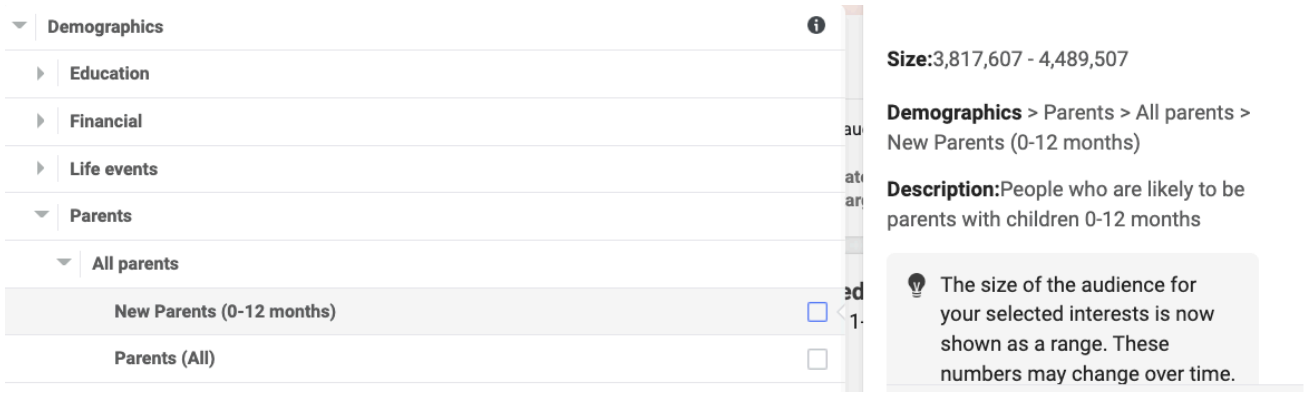
However, this is technical advertising and it gets complicated quite quickly:

- Everything is organized into ad campaigns that have one or more ad groups, which have many keywords and ads.
- There are different campaign types: search (most common), display, video, shopping, app, etc.
- You need to specify a schedule, location, and language.
- You need to specify your daily budget, bid strategy, and individual keyword bids.
- You can find additional keywords by using keyword research tools.
- Keywords can have different match types (broad, phrase, or exact) or you can even add negative keywords (e.g. “free”).
- Ads need to be short, and can’t contain trademarks or misleading statements (e.g. “free” if it is not free).
- Device targeting decides if you want to target desktop, tablet, or mobile devices.
- Ads should not link to your homepage. Instead, they should link to a landing page on your website that is optimized for that ad and keyword combination.
- Once your ad is running, you should optimize it depending on available metrics: click through rate, average position, conversion rate, bounce rate, etc.

I recommend that you open an account and then follow an in-depth tutorial, like Wordstream Google Ads Guide.

Facebook Ads

Another popular PPC advertising platform is Facebook Ads. Unlike Google Ads, here you can’t target specific keywords. Instead, you target users by demographic, interests, or behaviors. For example, this is demographic targeting of new parents:



Another big difference is that search engine users are more focused and ready to convert. Facebook users are in leisure mode, browsing updates from family and friends, and unlikely to buy serious software. Still, Facebook Ads are cheaper than Google Ads and are worthwhile if there is some customer appeal of your product (e.g. fashion, watches, consumer apps, etc.).

Other PPC systems

LinkedIn is more serious than both Google and Facebook. It is used by business people and it can be good for selling B2B software. Unfortunately, there are two problems:

- LinkedIn targeting is very generic. Google Ads allows targeting by an infinite number of keywords. Facebook Ads allow targeting by many specific demographic, interests, or behaviors. LinkedIn targeting is limited to location, company info, and profile info.
- LinkedIn ads are quite expensive. Too many companies are pouring money into LinkedIn advertising.

Zeljko used LinkedIn ads to test TestDome's landing page messages and the average CPC was $7. Unless you are selling expensive software, LinkedIn is probably not worth the effort.

Software review websites live out of software vendors paying to position higher in category lists. The two largest ones are:

- G2.com: They have three paid plans for companies.
- Capterra.com: They have Basic and PPC plans. With the PPC plan, you pay for every "lead" (visitor to your website). We have used Capterra with mixed results, as we were not sure that the ROI was positive.

Bing Ads are similar to Google Ads, but much smaller, because 92% of searches are made via Google. Still, it might be useful if you are targeting Microsoft users.

Zeljko never tried advertising on Twitter, Instagram, TikTok, and YouTube, so he can't give any advice there.

Content marketing

Once you learn which keywords and content attracts your customers, you can go to the next step. Content marketing attracts new customers by creating and sharing valuable free content: articles, blog posts, quora replies, videos etc. That solves two PPC advertising problems:

- Content marketing is free, meaning your content is shown organically without paying.
- Good content is orders of magnitude more likely to be read than a good ad, because of banner blindness.

However, there is a drawback that scares people:

> For content marketing to work, you need to have some talent for writing.

Even for video content, it needs a script, so there is no workaround around writing requirements.

Content marketing is not completely free, as you pay with your time to get a good article, blog post, quora reply, video script, or similar. But, once you put in the effort to figure out what people want, you will have a content recipe that will pay future dividends. A good piece of content will attract new users for years to come. Contrast that to PPC advertising, where as soon your credit card bounces, your marketing stops. Content marketing is a long-term investment.

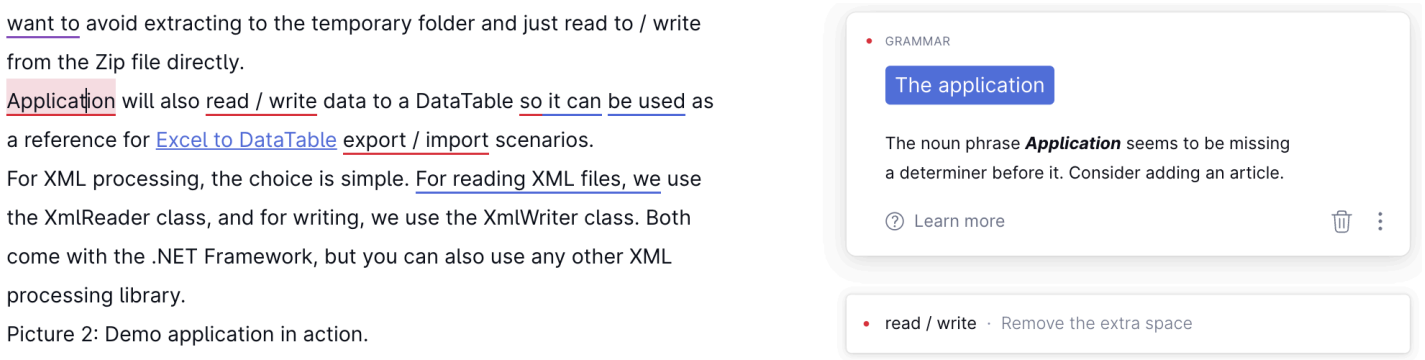
Don't be scared of writing. Writing good content is different from school writing. Even if you hated school essays, you can be good at content writing. Good content doesn't need to have literary flair, wide vocabulary, elaborate plot, or even be grammatically correct. Good content just needs to explain something the reader wants to know in the simplest possible terms. Being funny or creative is a bonus but not a requirement.

For example, one of Zeljko's practical problems became the basis for an article. In 2006 Zeljko was working on the GemBox.Spreadsheet component used to read and write XLS files. But, Microsoft announced Excel 2007 will use the new Open XML format (XLSX file extension). It took some time to implement XLSX support in GemBox.Spreadsheet, because the Open XML file format is quite complex. The full Open XML standard is more than 6000 pages of dense PDF! After implementing it , Zeljko condensed his learnings into a short article and accompanying source code. That article was published for free on five C# programming websites: CodeProject, C# Corner, developerFusion, codeGuru, and ASPAppliance (they even paid for the article!). Out of all them, the CodeProject version became the most popular:

As of 2022, the CodeProject article has 380,900 views and 13,500 downloads. What is more impressive is that all readers are GemBox target users—they are C# programmers who need to implement Open XML support

right now. To get the same targeted audience with Google Ads, with CPC of $1, would cost $380,900. Yes, it took Zeljko two weeks to write and publish the article. But two weeks of effort for $380,900 worth of marketing is not a bad deal.

Note what the article lacks. Grammarly is quick to find a lot of errors:



There are no funny bits, no plot, no characters, no drama, no twist, no smart words. It has zero literary flair. Just a boring explanation of how to process XML by a non-native English speaker. But still, 300 people bookmarked the article. Why? Because *people use the internet to solve their problems*. This was one of the first articles on the Open XML format in 2006. And it was pure poetry compared to the alternative—reading 6,000 pages of Open XML specification. The article was a 10 minute read and the accompanying source code can get you started.

Good content writing is not only less demanding, it sometimes contradicts school advice. For example, take the following advice on good content writing:

- *Write like you talk*. Simple, conversational tone is the easiest to read. For an explanation why, read Paul Graham's essay Write like you talk and the Write Like You Talk, Figuratively Not Literally article.

- *Don't use uncommon words*. Good copy is accessible to all. Most internet users are *not* native English speakers, are *not* university educated, and are often *not* domain experts. Even if they are all of that, it is often easier to read text that uses common words. The general rule is only to use the top 5,000 English words or words that are well known to the target audience.

- *Bottom line up front* (BLUF or inverted pyramid). Classic dramatic structure starts with a long introduction of characters and environment. The plot slowly develops and the most exciting and important bits come towards the end (see Kenneth Rowe's story structure or Freytag's pyramid). On the internet, no one has time for long introductions. If the readers don't find something interesting above the fold, they will close the tab and go to the next one. Content writing is more similar to journalism: start with a killer headline, then with a lead paragraph, and then continue with a story.

- *End with a call-to-action*. After a book ends with a satisfying resolution, you can close it and put it back on the shelf. Good online content ends with something still unresolved. To resolve it, readers follow a suggested action. That is called a call to action (CTA), and your last paragraph should motivate the reader to execute that CTA: signup, download, buy, contact us, etc.

- *Less is better*. Both school essays and final thesis have a minimum page requirements. You can get to a page count in school by being overly verbose, but that doesn't work for online content. The shorter you can explain something, the better. 800 words is better than 1600 words. If you can explain the whole concept in a tweet, that is even better.

Good content writing is very close to journalistic writing. Stephen King said this of his first job as a journalist:

Stephen King @StephenKing · Jul 7, 2018

My first paying job as a writer was the sports beat for a small town weekly. The pay was tiny, the experience was enormous. I learned more about writing in three weeks than I ever learned in school.

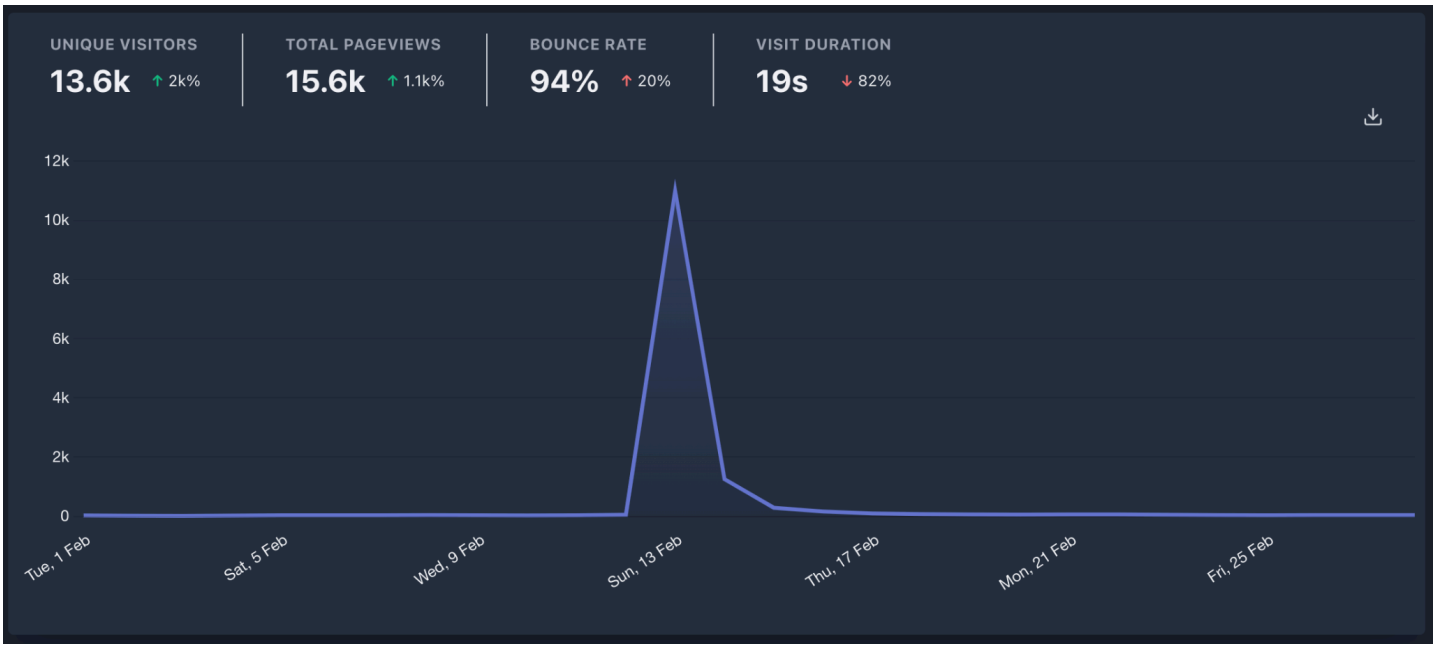
Think about Stephen's statement. He valued three weeks at a local newspaper more than his Bachelor of Arts degree. To get good at writing you need to start writing for real people.

Once you start writing content, readers will tell you how good it is. If your content gets only 15 likes/upvotes, it is not good enough to have marketing value. Even 50 likes/upvotes is not worthwhile, if most of the likes/upvotes are from your friends. Only when likes/upvotes get into hundreds and people start to share your content, you will know it works as content marketing.

Different content works for different audiences. To learn how to satisfy the Hacker News audience read Aline's article on How to write stuff that gets on the front page of Hacker News.

Once content is published, it needs to be distributed. Common distribution channels for geeks are Hacker News, Reddit, Lobste.rs and similar. Articles that become popular there commonly experience a hug of death—website crashing due to a large influx of visitors.

For example, one analysis of website traffic after the Hacker News #1 ranking has this chart:



Notice a few things:

- The author's website had zero daily visits before HN discovered him.
- Once the article reached #1 on HN, it had 10,000 daily visits.
- Five days after the peak, the website again had zero daily visits.

This shape of traffic is typical for any viral content. Content spreads in the community until most people read it, and then it is old news. After that you need to find a new audience.

Search engine optimization (SEO)

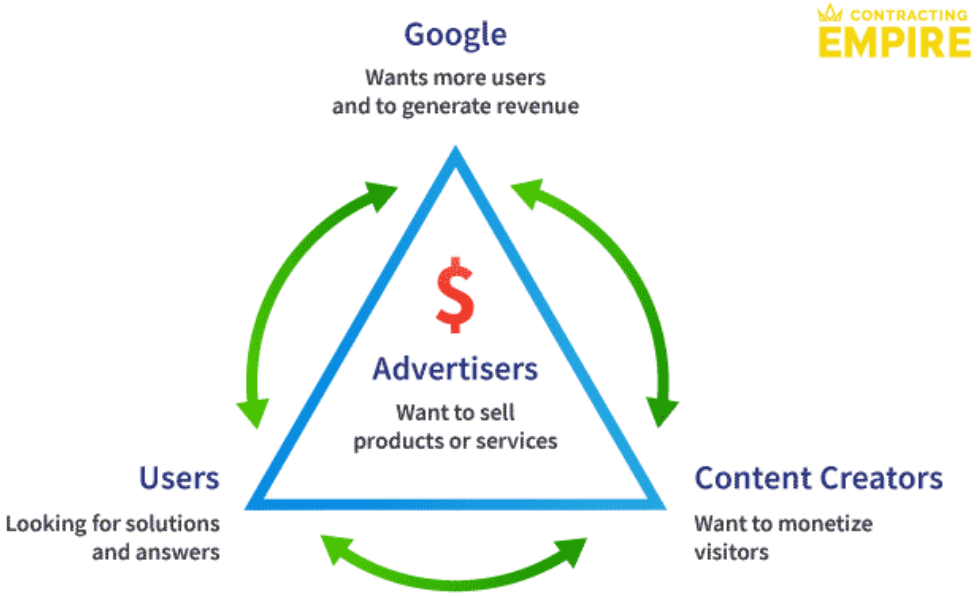
As already said, content marketing is great, but it needs repeated promotion. If the promotion succeeds, there is a spike of visitors who came because of the buzz, not because they currently have a problem. More often than not, the promotion fails, because the content is not “viral” enough. But there is a way to:

- Have *constant incoming traffic*, without promotions.
- Have visitors that come because they *need to find a solution right now*.
- Have *a lower bar for writing*, where it just needs to be good enough, not viral.

SEO offers all those benefits. Note that you still need good content, and SEO is a way to distribute that content.

Zeljko doesn’t like how many sources describe SEO. It is often presented as a way to trick search engines, or trick visitors. As described in a previous section, tricks don't work. Below is an honest SEO explanation.

First, we need to understand the holy trinity of SEO:



Users go to a search engine like Google in order to solve their problem. If they don’t have a problem, they will just keep scrolling their social network of choice. But Google doesn’t have solutions. There are an infinite number of search queries; it would be impossible for Google to have an answer to all of them. Even if they had all answers today, 15% of all search queries are never seen before, so they would need to update answers every day. Instead, Google decided to use other people’s content and make money on ads next to search results. Google needs content creators to create content for specific search queries. That is a silent agreement between Google and content creators:

> If you create content that answers a search query better than current search results, Google will be happy to put you in the first place.

SEO trinity is a triangle of dependencies. Users google in search for answers. Google needs content creators with actual answers. Content creators need users to be profitable.

In order to rank well (positions 1-5) on Google’s search engine results page (SERP), Google needs proof that your content is better than thousands of other possible pages. Google decides that algorithmically (see PageRank), so there is a strong incentive to “cheat” the algorithm. That cheating is called Black Hat SEO, and

you should never use it. Besides moral reasons, black hat SEO doesn't work today as Google switched from using purely text and link analysis to using user behavior. For example, Google knows:

- The probability of a click on a specific search snippet.
- That you are not satisfied with a page if you go back to SERP.
- That all results on SERP were probably inadequate when you go back and change the search query.

That is why the best SEO advice is:

> Don't optimize landing pages for Google, optimize them for website visitors and their search intent.

Of course, a landing page should also have text with important keywords, and primary keywords should repeat in text, title, headings, and other parts of on-site SEO. Without on-site SEO, Google will not even consider your landing page. But, that part is a technical routine and is easier to learn. The hard part is creating a page that is engaging to visitors.

The SEO topic is large and gets subdivided in different ways. Common SEO division is:

- On-page SEO — The content of your web pages: blogs, copywriting, articles, etc.
- Off-page SEO — External stuff that influences rankings, e.g. backlinks.
- Technical SEO — Technical stuff needed for rankings, e.g. sitemap, site speed, crawler visibility etc.

Another division is by primary SEO strategy:

- *Programmatic SEO* — using automatically generated or user-generated content (UGC) to create landing pages at a large scale. E.g. Booking.com, Zillow.com, Pinterest.com etc.
- *Editorial SEO* — using high-quality, editorial, long-form articles focused on topics related to your audience. E.g. HubSpot Blog, interviewing.io blog etc.

That is SEO in a nutshell, now it is time to go deeper. I will not go into details of on-page SEO, off-page SEO, and technical SEO, because there are already many great resources for that. I will also not go into details of editorial SEO strategy, because it is not my expertise, and I think it is unlikely that hackers like you will choose an editorial-heavy strategy.

What hackers really get interested in and find quite natural is programmatic SEO. I suggest you get into programmatic SEO by reading PDFs in the Programmatic SEO folder.

Start with Winning at SEO - Issue 34 - Lenny's Newsletter.pdf (article ends at page 12) by Brian Ta (SEO lead for AirBnB, Strava and AngelList). It is from the popular Lenny's Newsletter which is a great startup resource.

Then read the following SerpDB guide (unfortunately, some images are missing from the archive):

- Programmatic SEO (1) - How to Do Keyword Research at Scale.pdf
- Programmatic SEO (2) - Competitive Analysis of the Search Results at Scale.pdf
- Programmatic SEO (3) - Creating Landing Pages at Scale.pdf

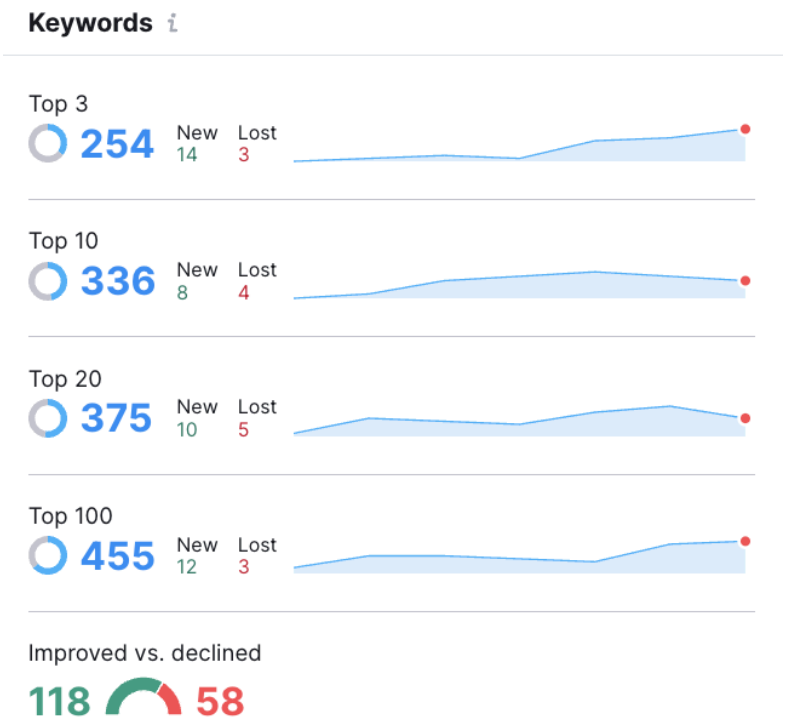
We have used both editorial SEO and programmatic SEO in my companies.

At GemBox, we used editorial SEO in the form of articles explaining how to import data from or export data to various office formats. An article mentioned before is Read and write Open XML files. But the GemBox website has many more articles. Each component GemBox sells has online examples, and each example is an article with executable code, e.g. Create and write Excel files in C# and VB.NET.

At TestDome, we tried using editorial SEO by writing blog articles, but we failed to get noticeable rankings. However, what really worked for TestDome is programmatic SEO. We noticed that both jobseekers and companies search for specific tests (e.g. “C# online test”), although for different reasons. Jobseekers want to practice solving questions, while companies want to see how relevant the questions are. Since they both want to see questions, TestDome decided to have 30% of all questions available for free, directly on landing pages. For example, the C# and .NET Online Test landing page lists 21 public questions you can solve directly in the browser, without signing up. Why give away all this content and maintain all evaluator machines for free? Because visitors love good free content, and that makes Google love you. Below is a screenshot from SEMRush position tracker:

Keyword	Intent	SERP	Pos. Nov 28	Pos. Dec 4	Diff
testdome c# programming test	I		1	1	0
c# programming test programming test	C		1	1	0
c# coding test programming test	C		1	1	0
c# programming test for interview programming test	C		1	2	↓1
c# practice test programming test	I		2	2	0
c# online test programming test	I		4	2	↑2
c# test programming test	I		3	3	0

At TestDome, we currently have landing pages for 132 different tests. As a result, 254 of our 714 tracked keywords rank in the top 3:



Note that these results are nothing spectacular. TestDome does not even have the biggest SEO audience in the small city of Zagreb. A quick check on SpyFu shows that Matija Babic's TasteAtlas ranks for an order of magnitude more keywords. But the principle is the same: programmatically arranged quality content. TasteAtlas claims to have 10,000 dishes, drinks, and ingredients, like for example Scottish Haggis.

Growth hacking

SEO is great, but it has one limiting factor. Even if ranking for all niche keywords, you are still limited by the total search volume of those keywords. People who never type "C# programming test" into Google will never learn about TestDome's C# tests.

Growth hacking allows you to grow bigger and faster. Growth hacking covers multiple topics, but I will focus on *incorporating a marketing growth hack directly in a product*.

Famous examples of product growth hacks:

- Hotmail was the first to add a signature line to every outgoing email, inviting email recipients to get a free account. That single line was all it needed for super-fast growth to 12 million users (or around 20% of the email market at the time) in 18 months:

  PS: I love you.Get your free e-mail at Hotmail

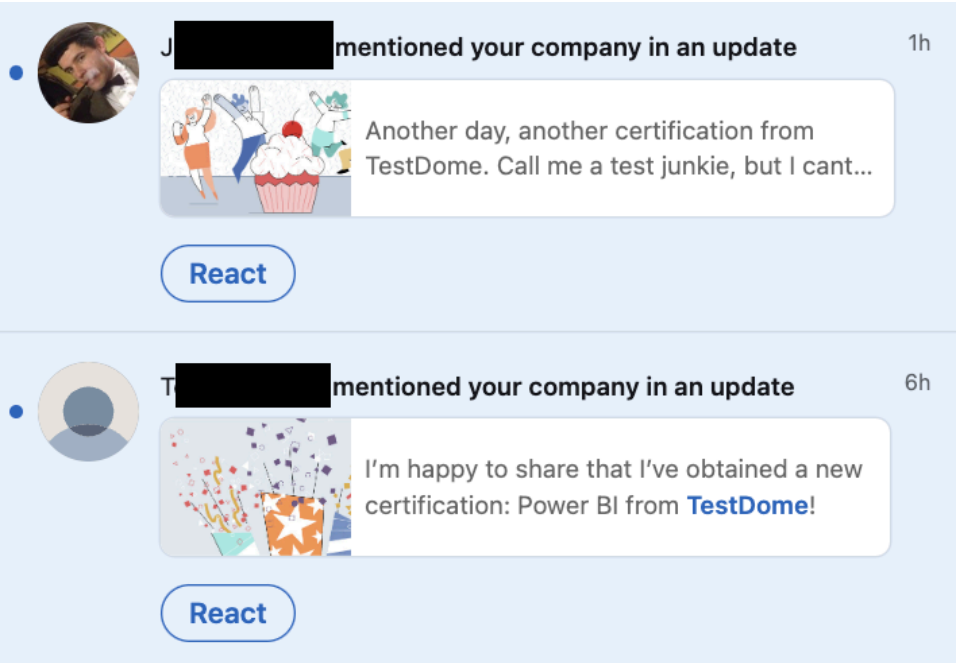
- Dropbox offered bonus storage to users who referred their friends. That was just one of many gamified rewards:

Camera Uploads	Bonuses for using Camera Uploads	25/8/2012	3 GB ✔
Getting Started	Completed!	2/8/2012	250 MB ✔
Simple Tasks	Posting <3 to Twitter	31/7/2012	125 MB ✔
Simple Tasks	Why you love Dropbox	31/7/2012	125 MB ✔
Simple Tasks	Linking with Facebook	31/7/2012	125 MB ✔
Simple Tasks	Linking with Twitter	31/7/2012	125 MB ✔
Simple Tasks	Follow @Dropbox on Twitter	31/7/2012	125 MB ✔

- Airbnb realized people often looked for alternative accommodation on Craigslist, so they added a one-click "Post to Craigslist" feature.
- Facebook asked users to add their contacts, and then they would email those contacts if they were mentioned or tagged on Facebook.
- Shazam is interesting because their growth hack worked in real life. People used Shazam in bars and clubs to identify the current song. They would raise their smartphone up in the air, and then show the song name to their friends. After that impressive demonstration, people nearby also wanted to use Shazam.

Notice that in all above examples the product was designed with intentional virality.

At TestDome, we created jobseeker certificates as a growth hack. Jobseekers naturally want to solve tests to check their knowledge. If they get a good score, we will issue a gold or a silver certificate (like this one) they can share on LinkedIn, Facebook, Twitter, or their CV. Tens of thousands of certificates were shared, and every day we we get multiple notifications from people proud of their certificates:



Summary

I hope I convinced you that online marketing is not hard, can be done on a limited budget, shares similarities with programming, and should be thought of before creating a product.

To summarize:

- Pay-per-click (PPC) advertising is great when starting. It is used for testing keywords, testing MVPs, finding first customers, and iterating fast. But, depending on your prices and conversion rates, it can get too expensive.

- Content marketing is free and has better conversion rates. But, you need to be able to create good content and know how to promote it.

- Search engine optimization (SEO) is a way to constantly distribute content to visitors looking for a solution. But, total traffic is limited by niche keywords you can rank for.

- Growth hacking is a free way to have constant exponential growth. But, your product needs to have something worth sharing.

There are many other marketing topics we didn't cover, but I don't think you will need them right now. For example:

- Sales — if your product costs a few thousand dollars, you will need sales skills. People don't buy $5000 software just based on a Google search result or an article. The more expensive your software, the more meetings, documents, presentations, and customizations you will need. Enterprise software can take years to sell.
  But, I don't think you need to know sales practices, because I doubt your software will cost more than $500. Even if you decide to be a salesman for your software, you will be at a big disadvantage because you are too young and you are not a native English speaker. If you sell software for a few hundred dollars, customers will just buy, without sales presentations. That is your advantage, as nobody will know from your professional-looking website that you are one teenager from a small country.

- Branding — if you sell a customer product for masses, you need to think about branding. That includes brand values, messages, colors, logo, etc. For example, if you are making a new ketchup for US supermarkets, branding is everything. There are dozens of ketchup brands, you need to be distinctive.
  But, I suspect you will not work on customer products for the masses. If you make a small niche software that solves a problem, you don't need to put much effort in branding. Focus on solving the problem, not what your company colors are. For the first year of TestDome we didn't have a logo, we just had text "TestDome" in a blue font.

- Social media marketing — if your product is in the lifestyle category, you will need social media marketing. Lifestyle products are products seen as expressions of one's style, values, and way of life: clothes, footwear, travel, cars, watches, food, movies, music, video games, etc. Lifestyle products are naturally shared and endorsed on social media.
  But, I suspect you are not in the lifestyle category, and niche software is not sexy enough for successful social media marketing. Remember, 20 likes from your family and friends is not social media marketing, it is just a vanity metric.

The four methods that we covered (PPC, content, SEO, growth hacking) are the tools you need to get started in a niche software business. If you follow the given suggestions, you will be able to sell software to customers that are oblivious to the fact that there is a single youngster behind the company. I was in that situation — my first two customers from France and the US didn't have a clue I was a single guy working from my parent's basement.

Homework: marketing

Now when you have a general idea of online marketing, it's time for marketing homework. In preparation for homework we can role play in class.

Homeworks is given in separate documents, Zeljko will give you more details.

Startup idea generators

Generators, new paradigms, and trends

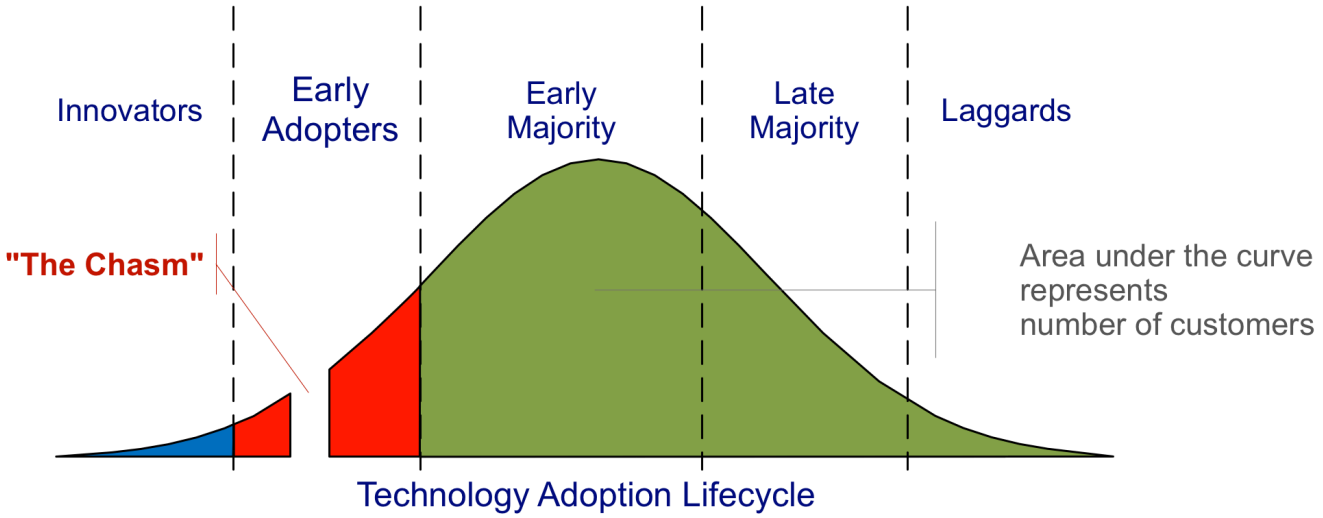
Coming up with good new ideas or becoming a domain expert is really hard. Fortunately, there are shortcuts. Sometimes ideas get "recycled" into new forms. In the lack of better word, we will call such recycling systems "idea generators":

> *Idea generator* is a system for converting old, successful business or product ideas to new, potentially promising ideas.

There are general idea generators, which work across different domains, and domain-specific idea generators (e.g. computer-related idea generators). Domain-specific idea generators are born when there is a significant change in the domain:

- New paradigm — *new technology* enables new ways to solve old problems.
- New trend — general *sentiment of consumers* in a market changes.

Note that a new paradigm comes first, but that is often noticed only by early adapters. Only later the paradigm becomes obvious to most consumers, and we see a market trend. Classic book that explains that is Crossing the Chasm by Geoffrey A. Moore. It explains how a product or technology is used by different customers in each stage, and how to transition from each phase to the next one:



Take cell phones for example. First mobile phone was the Motorola DynaTAC (a.k.a. "brick"), invented in 1973. At that time only a few telecom engineers understood this was a new paradigm. The phone and the first network were released 10 years later, in 1983, but were not a hit. The phone weighed 1.1 kg and cost $4000 (equal to $11,000 in 2023). Because of the weight and cost, in 80s cell phones were sold as pricey addon for luxury cars:

First owners of cell phones were not praised for being innovators. They were often ridiculed by the public, who couldn’t afford such phones, as self-important individuals who couldn’t wait to get to an office to call someone. Then in the 90's cell phones became cheaper and smaller, so the same public started buying cell phones in masses. By the end of the 90's, everybody got cell phones and people who didn’t have one were ridiculed. Then the market trend was to have the smallest cell phone possible, because each new generation was lighter and thinner. You were made fun of if you had a large phone.

Then in 2007 Steve Jobs launched an iPhone and a new paradigm of smartphones was born. Again, at first, people who bought the first iPhones were made fun of because iPhones were expensive, large, lacked a keyboard, and the battery would only last for a day. As smartphones became cheaper, people started to realize the benefit of having a big screen and apps, and started to call people with feature-phones “dinosaurs”. Soon smartphone apps needed bigger and bigger screens, and each new generation of phones became bigger and bigger:

Let's think about this. In the past 40 years, you were made fun of:

- First, if you *had a cell phone at all*.
- Then, if you *didn't have a cell phone*.
- Then, if *your phone was large*.
- Then, if *your phone was a smartphone*.
- Then, if *your phone was not a smartphone*.
- Then, if *your phone was small*.

That is the difference between new paradigms and new market trends. New paradigms here were *cell phones* and *smartphones*, and they were products of visionary engineering. Everything else were marketing trends, byproducts of people's psychology, which is not always rational. People can deny an invention is great for more than 20 years, until you lower the price of that same technology to $299. As soon as they and their friends can afford it, they "discover" the benefit of technology. Note that dominant market players have no problems following new trends but often fail to follow new paradigms. Nokia was the biggest cell phone manufacturer until smartphones came, then it went bankrupt.

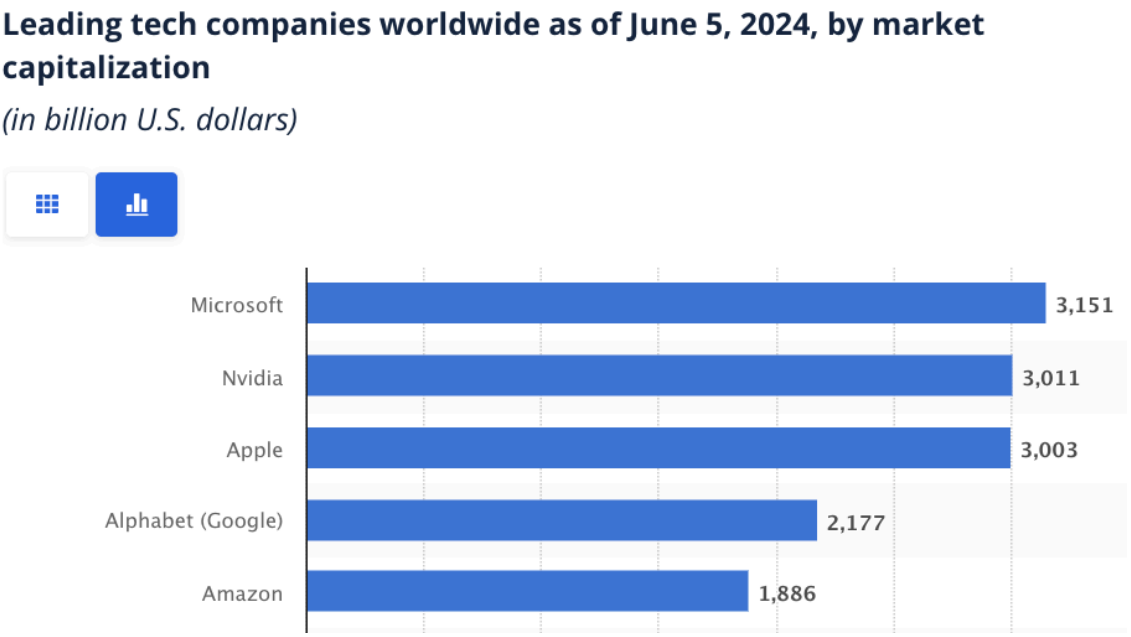
As hackers, be aware that a new paradigm that is obvious to you may take a long time to become obvious to customers. Some new paradigms adopted by early adopters have inherent adoption problems and never become market trends. In retrospect, we sometimes call them "hypes" or "bubbles". For example, most companies from the dot-com bubble failed to make a market impact.

As of 2024, people are endlessly debating if the following new paradigms will change the world or become remembered as hypes:

- VR / AR / Metaverse

- Blockchain / Web3 / Bitcoin
- AI

What do you think, which of the following will become a market trend, and which one is just a hype? If they become a market trend, when will your parents start using it? Would you bet your career or your company on any of the above? Remember, biggest tech companies today were all-in on new technologies:



Apple was betting on the microcomputer market with Apple I. Microsoft was betting on software for microcomputers with MS Basic. Nvidia was betting that GPUs and not CPUs will do the most of the future compute and that other people will write GPU applications if they standardize CUDA.

Can you remember what Google and Amazon were betting on (we mentioned that before)?

General business idea generators

Most of the business idea generators in this category are timeless, but specific examples are often related to a specific market trend.

ToDo: the sections below are still being written. To suggest another startup idea generator or to provide feedback, make a comment or send an email to Zeljko.

Free service, monetized via ads

Idea:

> Take an existing product X, which is paid, and make a copy which is free but monetized via ads.

Examples:

- Newspaper, radio, and TV
- News portals, blogs, online publications
- YouTube, Facebook, Instagram, TikTok

This idea is old as advertising. As soon as newspapers were invented, publishers realized that they could charge extra for printed ads. But printing was expensive and readership small, so newspapers still made most of the money by selling newspapers to readers. As technology progressed, it was cheaper and cheaper to print

for more people, and ads became more lucrative. Then someone realized they could offer discounted or even free newspapers in order to get more readers, and make more money with ads.

The same system then got applied to radio and to TV, and that is the reason today we have free radio and TV programming. But note that there is a big difference in the ad format: newspapers have *non-interrupting ads*, you can just ignore them when reading an article. Radio and TV have *interrupting ads*, you need to watch the ad before the program continues. Interrupting ads are more effective for advertising and pay more. But, users *hate* interrupting ads, and they may switch to other channels which are ad-free or have less ads.

If you want to make a profitable ad-based business, it is important to make a realistic ad revenue calculation based on the ad format your users will tolerate. In practice, that means you need to have both low costs and millions of daily views. Just as reference:

- Facebook charges display ads only $0.10 – $0.60+ per 1,000 impressions. But Facebook is a billion dollar company because it has both *reach* (billions of people use FB and Instagram) and *frequency* (people use it many times a day).
- YouTube cost-per-view is $0.010 – $0.030, meaning 1000 views are $10 – $30, much bigger than Facebook.

But, you don't have either the brand or targeting tech of Facebook or Google, so you would need to use an ad network such as Google AdSense. If interested, take a look at Google AdSense income calculator. My advice for most readers is to forget about this model unless you have millions of views per month.

Paid service without ads, monetized via subscription

Idea:

> Take an existing product X, supported by ads, and make a premium subscriber version without ads.

Examples:

- Cable TV packages (e.g. HBO)
- Digital satellite radio (e.g. Sirius XM)
- Online music streaming (e.g. Spotify, Deezer, Apple Music, YouTube Music)
- Online video streaming (e.g. Netflix, HBO, Disney)
- Paid ad-free versions of ad-supported products (e.g. YouTube Premium)
- Paid newsletters (e.g. Lenny's Newsletter with 325,000 subscribers)

ToDo

Bundle

Idea:

> If products P1, P2... PN are often used together, make similar products and bundle them in a package. Offer that package with a more affordable price and better integration than separate products.

Examples:

- Microsoft Office (replaced WordPrefect and Lotus 1-2-3).
- Adobe Creative Cloud (includes Photoshop, Illustrator, Premier etc.)
- Google Workspace (includes Gmail, Contacts, Calendar, Meet, Drive, etc.)

- Figma (replaced Adobe InDesign and Sketch, but bundle was not the only benefit)

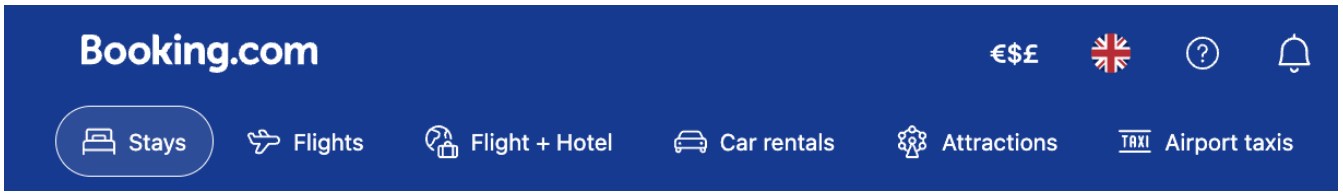
ToDo

Bundling features

This is a special case of bundling. Instead of bundling products (P1, P2... PN), bundle features (F1, F2… FN) that are commonly used together in order to cover a use case. Users then don't need separate apps and data transition is seamless.

Examples:

- Instagram boundled three features used together when posting to a social network: taking a photo (Camera app), applying a filter (photo editing app), and posting to social networks (FB, Twitter, etc.). That is how they kickstarted the network effect — Instagram app was useful for posting to other social networks even when there were no users on Instagram.
- WeChat is a Chinese super-app because it provides: text messaging, hold-to-talk voice messaging, broadcast (one-to-many) messaging, video conferencing, video games, mobile payment, sharing of photographs and videos and location sharing.
- Booking.com and Expedia provide five separate search features that are used together when booking a trip: accommodation search, flight search, rental car search, airport taxi search, and attraction tickets:



Unbundle

Idea:

> If a popular bundle of products/services/content P1, P2,... PN has a program/service/content PX that can be used on its own, then offer PX as separate products/services/content. People will buy it either because it is cheaper to buy only PX, or because your version of PX has advanced features.

Examples:

- Music albums -> single MP3s
- Newspapers -> Blogs, classifieds, news
- University -> courses

ToDo

Unbundling features

Idea:

> This is a special case of unbundling. If a popular product has features F1, F2... FN but feature FX has value on its own, then extract feature FX into a product and make it much better for target users: more options, faster, more configurable, better platform support etc.

Common pitfalls:

- Low price is usually not enough to be better. Your target users already have a license for product X, no matter how low the price is, you are extra cost to them. But, if they use product X to make money, power users will be willing to pay extra.

- Your product based on functionality FX needs to be well integrated in their workflow. That means easy import and export. If you are not integrated, your product is another obstacle in their workflow.

Examples:

- Craigslist was an immensely popular classifieds website in the late 1990s and early 2000, mostly free. But its interface was rudimentary and generic. With years, separate successful companies popped up by unbundling different types of classifieds (see Unbundling Craigslist). Most famous of them was probably Airbnb, which sought to directly replace Craigslist's "vacation rentals" category. Notice that Aribnb did such a great job that today they are more valuable than Craigslist—Craigslist's estimated value is $3 billion in 2017, while Airbnb is a public company with a capitalization of $78 billion in July 2023.
  Notice that Airbnb avoided common pitfalls:
  - Airbnb is not cheaper than Craigslist, it is actually more expensive. Craigslist was mostly free, while Airbnb charges around 14% or rental fee. But that extra fee includes professional services, like professional photography, money back guarantee, and damage insurance.
  - Airbnb was well integrated with Craigslist. In their early days, they had a "Post to Craigslist" button. You would create a great listing on Airbnb, click that button, and you would have listings both on Airbnb and Craigslist (which would link to Airbnb).
- Excel is immensely popular. But it is generic and specialized to any domain. Experts in any field start developing small applications in Excel but often hit limits of the tool. Over the years, hundreds of startups made good money by unbundling different parts of Excel. See The SaaS Opportunity Of Unbundling Excel.

Homework: keyword research

We still haven't finished with idea generators, so we will do another marketing homework on keyword research.

Computer-related idea generators

Most of the computer-related generators below are new paradigms and market trends that were associated with a time period. But, that doesn't mean they don't work anymore. Some of them evolved into modern trends, and some business domains still didn't catch on to those trends.

Digitalization: convert paper processes to digital form

Idea:

> If a process is currently done with paper (forms, letters, documents, post-its, paper cards, etc.), convert that process to a digital form, done on a computer / smartphone / specialized device.

Examples:

- Mail → email
- Checks → electronic payment

- Paper worksheets → computer spreadsheets

- Paper blueprints → CAD (Computer Aided Design)
- Question: how many times was payment digitized?

ToDo

Automating things that couldn't be automated before

Idea:

> If a process that required human work can now be automated, solve the problem on a computer / smartphone / specialized device.

Examples:

- Human translation → machine translation
- Manual video transcription → automated transcription

ToDo

Personal computer revolution (80's)

Idea:

> If a program only exists on a large computer (mainframe or mini-computer), develop the same program for a personal computer (microcomputer).

Examples:

- Microsoft realized the paradigm shift and first developed BASIC and then MS-DOS. Larger computers already had BASIC interpreters and operating systems that were much more powerful (e.g. HP Time-Shared BASIC, z/OS).
- Autodesk realized the paradigm shift and developed AutoCAD for PCs. Larger computers already had advanced CAD/CAM software (history of CAD in pictures, wikipedia).

ToDo

Internet (90's)

Idea:

> If a program / service / content could benefit from online collaboration or zero-deployment, then create a version of program / service / content on the internet.

Example:

- Program: email client → Hotmail (zero-deployment).
- Service: newspaper classifieds → online classifieds (zero-deployment, speed, online communication).
- Content: paper newspapers → online news (zero-deployment, speed).

Multiple paradigms /trends:

- Early internet: client-server, server side
- Responsive: Javascript, JSON
- Client side: Web assembly

ToDo

Components, controls (Visual Basic in 1991) and APIs

ToDo

Integrations

ToDo

Social networking (MySpace in 2003)

ToDo

Smartphones (iPhone in 2007)

ToDo

List of artillery video games - Wikipedia

- 1979: Human Cannonball (video game) - Wikipedia

- 1991: Scorched Earth (video game) - Wikipedia
- 1995: Worms (1995 video game) - Wikipedia

- 2009: Angry Birds - Wikipedia

Bela - Play Store
- Market?

Rain Alarm - Apps on Google Play
- Market?
- Which business idea generator?

Who makes more?

Gig economy or “Uber for X”

ToDo

Cloud computing

ToDo

Blockchain and Web3

ToDo

VR, AR and Metaverse

ToDo

Deep Learning and AI

ToDo

Survey

TH MBA Survey 2025