YEAR 4 / ENTREPRENEURIAL ESSENTIALS

Minor Digital Skills

From idea to a link you can share.

Build something new. Add something useful to your business.

0403EE | 7 EC | 196 hours

WELCOME

What you will leave with



An idea. A useful first version.

A link you can share.

Start something new.

Or add to your current business.

Let someone try it.

Small enough to build today. Useful enough to show someone.

Why this is valuable for your venture



Explain an offer more clearly.

Make one customer task easier.

Test before building more.

Your business already exists. What useful extra could you add?

Today’s roadmap: idea, tools, share

01

Pick an idea

New project or one extra for your current business.



ChatGPT / Claude

02

Build with tools

Choose a small task.

Make a working first version.



Codex / Claude Code

03

Share the link

Publish it. Let a peer try it.

Improve one thing.



GitHub / Vercel

Understand enough AI to choose well. Learn by making.

Meet your instructor: Victor Cazacu

FOUNDER / FORWARD DEPLOYED ENGINEER

I build digital ventures and the systems behind them.

Explore my work



Portfolio example: LEGIT AGENTIC

Build. Test. Improve.

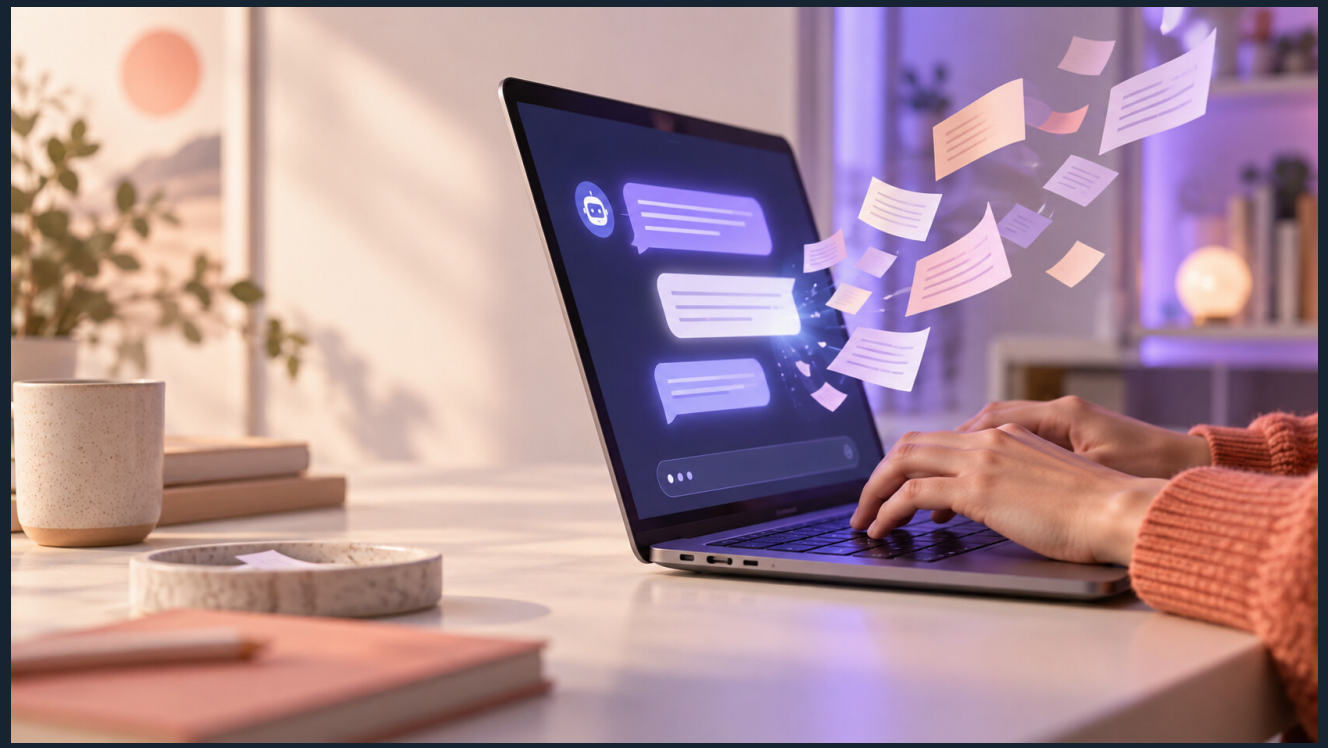
How often do you use AI?

A quick show of hands

Used AI today?

Use it almost every day?

Still getting started?



Which AI tools are already in the room?



Which tools are in the room?

Paid ChatGPT?

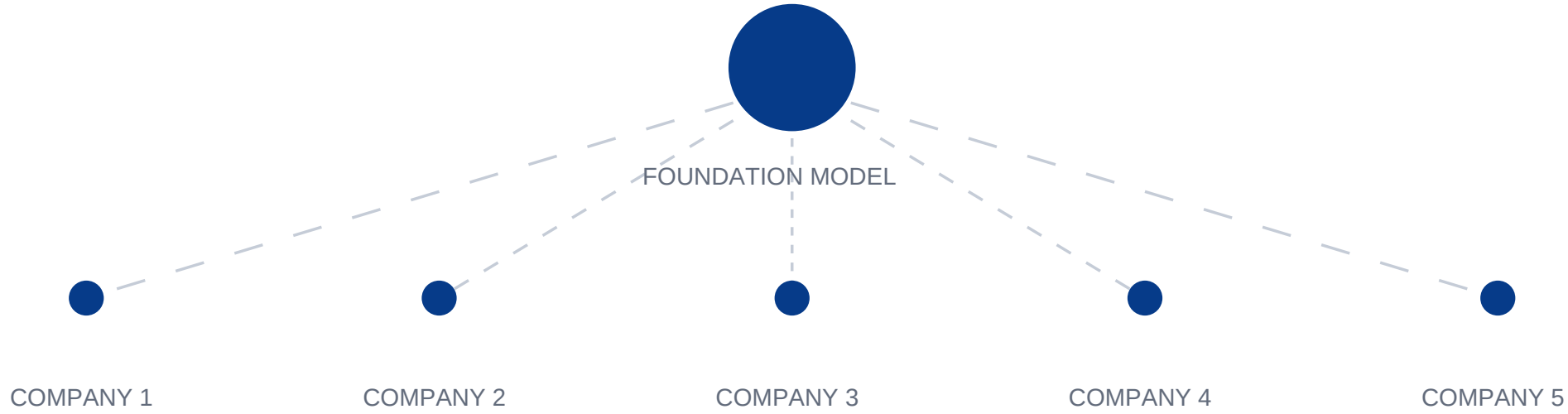
Paid Claude?

Free access, another tool, or neither?

THE BUSINESS LESSON

Everyone can now access intelligence.

The same foundational AI capabilities are available to your competitors.



Shared model access is the starting point for this strategic argument.

WHAT MAKES YOUR BUSINESS DIFFERENT?

Intelligence alone is no longer the moat.

When everyone can access it, access itself does not set you apart.

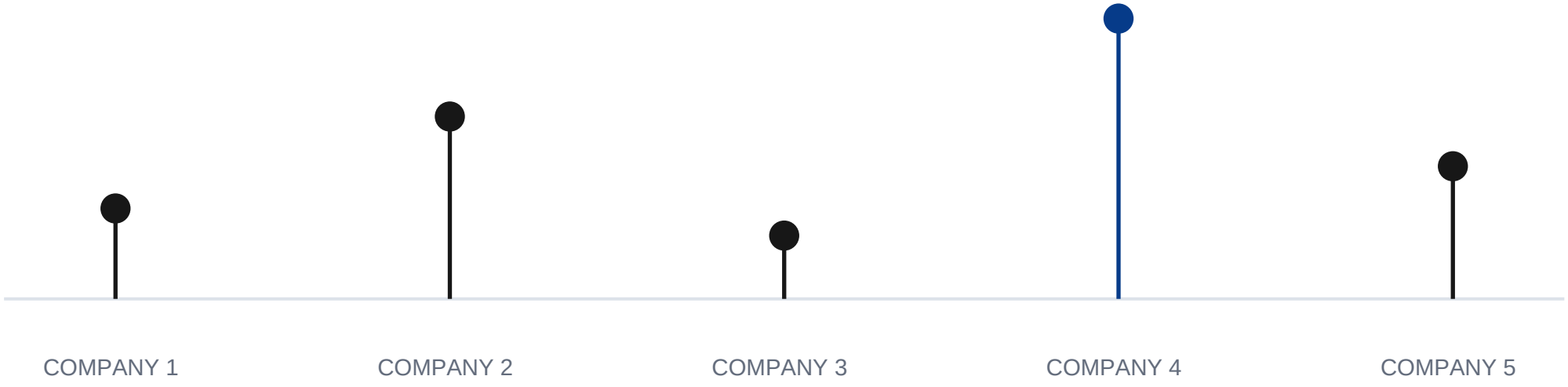


A moat is a competitive advantage that others find difficult to copy.

SO WHERE DOES THE ADVANTAGE GO?

Into deployment.

The advantage is where, how, and why you put intelligence to work.



Apply AI to a valuable problem, inside a useful workflow.

Deployment means real-world application. Diagram is illustrative, not measured results.

Forward deployed engineering



Bridge the customer and the code

Understand the real workflow.

Build inside its constraints.

Deploy, observe and improve.

The shift came when ideas met data and compute.



Geoffrey Hinton • 2013



Ilya Sutskever (left) • 2014

1943 The neuron as a mathematical model.

2012 AlexNet scaled deep learning on GPUs.

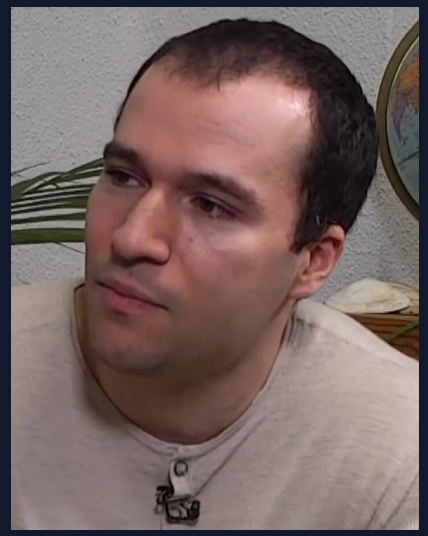
The research did not begin with OpenAI.

History: McCulloch & Pitts (1943); AlexNet (2012). Photos: E. Bach and S. Jurvetson, Wikimedia Commons.

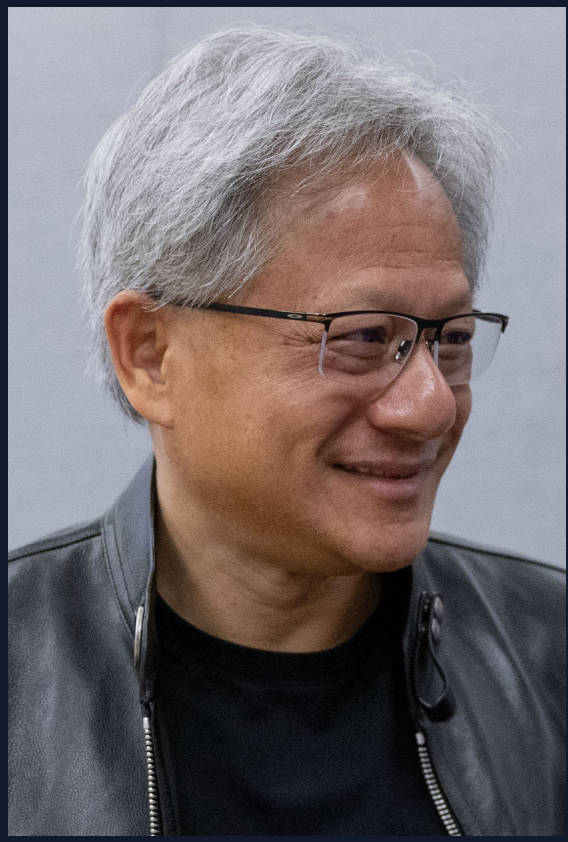
OpenAI meets accelerated computing.



Sam Altman



Greg Brockman



Jensen Huang

OpenAI was introduced in 2015.

In 2016, Huang delivered an NVIDIA DGX-1 to the lab.

OpenAI (2015); NVIDIA (2016). Photos: Alexlcory, Simulation, P. Dasilva / EC; details in notes.

The ideas were older. The doorway was new.

2017

TRANSFORMER PAPER

Attention Is All You Need.

2018

GPT

Pretrained language models.

2022

CHATGPT

AI you could simply talk to.

A research breakthrough became a public tool.

SAME FIELD. A MUCH EASIER WAY IN.

Vaswani et al., Attention Is All You Need (2017); OpenAI GPT (2018); ChatGPT launch (2022).

SOTA = state of the art

Best at what, and when?

SOTA means state of the art.

A benchmark is one test.

Try the same real task yourself.



SOTA map: OpenAI and Anthropic / Claude

OpenAI

GPT models

App: ChatGPT

Coding agent: Codex

API: connect models to your product

Anthropic

Claude models

App: Claude

Coding agent: Claude Code

API: connect models to your product

Same task. Compare quality, speed, cost and reliability.

The Top 50 Consumer AI Apps, by Monthly Revenue

1. OpenAI	11. Midjourney	21. descript	31. Meshy.ai	41. EVOTO AI
2. ANTHROP\C	12. OpenArt	22. zeely	32. QuillBot	42. MINIMAX
3. Canva	13. openrouter	23. HeyGen	33. NovelAI	43. beautiful.ai
4. SUPERHUMAN	14. Otter.ai	24. jumpspeak	34. justdone	44. fal
5. Higgsfield	15. Figma	25. runway	35. fireflies.ai	45. n8n
6. replit	16. PLAUD	26. KlingAI	36. sudo write	46. Cognition
7. SUNO	17. Notion	27. TopazLabs™	37. Jasper	47. FaceApp
8. Lovable	18. reface	28. invideo	38. Photoroom	48. moises
9. perplexity	19. manus	29. TOPVIEW	39. Hugging Face	49. synthesia
10. IIElevenLabs	20. Speechify	30. GAMMA	40. Genspark	50. NOUS RESEARCH

Source: YipitData

Note: Gemini is excluded because its revenue could not be reliably separated from Google's broader consumer revenue.

Charts are for informational purposes only. Past performance is not indicative of future results. None of the above should be taken as investment advice; see a16z.com/disclosures.

A16Z

Benchmarks: three ways to test a model



Standard tests

Fixed questions or code tests.

Score correct results.



Blind comparisons

Compare unnamed answers.

Ask people which works better.



Your business tasks

Use your own real examples.

Check task success and errors.

Same conditions. Repeat tests. Compare quality, time and cost.

Parameters: what does 7B actually mean?



7B

7 billion learned values.

Training adjusts these values.

More is not automatically better.

MoE: total and active counts differ.

Parameters are not your input tokens.

Use benchmarks and your own tests to judge capability.

TWO DIFFERENT IDEAS

Parameters are learned numbers. Tokens are text pieces.



Tokens

Words, word parts or punctuation.

The pieces sent in and generated.

Parameters

Learned values inside the model.

Used together to predict tokens.

7B parameters does not mean 7 billion words.

AI IQ Bell Curve / model estimates from benchmark data



Source: AI IQ / aiiq.org chart and methodology

AGI: what if everyone can access stronger models?



AGI: broad capability across many tasks.

One score does not prove AGI.

Access may broaden; limits differ.

Your edge: what you build with it.

If your competitors use the same model, why choose you?

CLASS DISCUSSION / THOUGHT EXPERIMENT

A smaller model: where would you draw the line?

1T > **100B** > **10B** > **1B**

100M > **10M** > **1M** > **100k**

10k > **1k** > **100** > **10**



Divide by 10 each step. Imagine progressively weaker language.

At which point, if any, would you call it conscious?

Hypothetical sizes and behaviour; this is not a consciousness test.

How does a model learn and reply?

Patterns in. Predictions out.

Training adjusts the model.

Your prompt supplies context.

The model generates a response.



Training is repeated prediction and correction.



01 EXAMPLE

Show a passage of text.

02 PREDICTION

Guess a hidden or next piece.

03 ERROR

Compare with the actual piece.

04 UPDATE

Adjust weights; repeat.

Training changes the weights. Inference uses them.

Google ML Crash Course, neural-network training and LLMs; Cursor Learn, How AI Models Work. Illustrative AI-generated photo.

It generates the reply one piece at a time.

PROMPT

Write a luxury line for a potato.



MODEL

Predicts the next token

then repeats with that token added.

The everyday, reimagined.

Tokens are text pieces; their boundaries depend on the tokenizer.

Google ML Crash Course, Introduction to LLMs; Cursor Learn, How AI Models Work. Token boundaries below are illustrative. AI-generated photo.

The model can take more than one plausible path.



A / DRAMATIC

The everyday, reimagined.

B / PLAYFUL

Everyday gold.

Which fits your audience? Neither is automatically the winner.

Cursor Learn, How AI Models Work. The two campaign lines are illustrative; results vary with model and generation settings. AI-generated photo.

A model call has two sides of the meter.

INPUT TOKENS

Your instruction

+ earlier conversation

+ files or tool results

OUTPUT TOKENS

The answer the model writes

+ any generated continuation

INPUT + OUTPUT = USAGE

Longer context and longer answers usually use more tokens.

Cursor Learn, Tokens & Pricing. Token counts and billing vary by model and modality; examples are conceptual.

AI is not one model.

TYPE = what it can do. MODEL = the version you choose.

Text & code	Writes language or code.	A reply or a small website.
Reasoning	Works through multi-step problems.	Compare a plan and its constraints.
Image / audio	Creates or understands media.	A product image or a transcript.
Multimodal	Works with more than one medium.	Read a photo and explain it in text.

Types overlap: one model can handle several of these tasks.

Compare models by quality, speed, cost and context limit.

CONTEXT: give the model the right background.

Too little: "Make a page."



Give it the context:

My business is a bike-repair shop.

Customers need a rough estimate.

Use only the price list I provide.

Label the result as an estimate.

Add a "Request a booking" button.

Ask me if a price is missing.

Right context = audience + goal + relevant facts + constraints.

Cloud can be rented by the task. Local can run on your hardware.

APP • READY TO USE

API • PAY PER CALL

LOCAL • RUN THE MODEL

Where should inference happen?



Conceptual comparison; products and model licenses differ.

Open models: choose the task, then the size



Start with the model card

Task, language, licence and format.

Check available weights

Can your chosen app run this version?

Benchmark your shortlist

Choose the smallest model that works.

Open weights, cloud and API are different choices



Weights / licence

Can you download and use it?

Hosting

Your machine or a cloud service?

Interface

An app or your software via API?

An open-weight model can also run in the cloud behind an API.

More parameters usually need more memory



Parameters	16-bit	4-bit
7B	14 GB	3.5 GB
13B	26 GB	6.5 GB
70B	140 GB	35 GB

Weights only: approximate decimal GB, before runtime overhead.

Leave memory for context, the app and the operating system.

Fit the model to the machine you already have

CanIRun.ai

1. Find your GPU, chip or phone.
2. Check your RAM / GPU memory.
3. Explore models that may fit.
4. Try a small one on your own task.



canirun.ai/devices

Estimates, not guarantees: model format and context also matter.

Three tools for running AI locally



LM Studio

A desktop interface for local models.

Ollama

Run models and connect local apps.

llama.cpp

A configurable inference engine.

Why run locally? Privacy, offline access and control



Privacy

Local inference can keep prompts on-device.

Offline use

Download first; work without a connection.

Control / "uncensored" models

Filtering depends on the model and app.

Run a small model on your phone



PocketPal AI

On-device models on iPhone and Android.

Start small

Use a supported small quantized model.

Prove it runs locally

Airplane mode, then send a new prompt.

Optional class demo: a customer reply with Wi-Fi switched off.

What is worth putting online?

A new idea

Test a service offer.

Show a small product demo.

Collect interest.

An extra for your business

Estimate a quote.

Request a booking.

Help a customer choose.

Who will open your link, and what should they be able to do?

A landing page should answer the buyer in order.

WHAT IS IT?	**A clear hero and promise.**
IS IT FOR ME?	**Audience, use case and benefit.**
CAN I TRUST IT?	**Real proof and honest answers.**
WHAT NEXT?	**How it works and one clear action.**
WHAT DOES IT COST?	**Price and terms when relevant.**



Landing Page Core Questions

by lander.studio

① Hero	What is this? Is it something I need?
② About	Who are you? Can I trust you?
③ Features	What can this do? How does it work?
④ Benefits	How will this help me? What's in it for me?
⑤ How it works	What happens after I sign up? Is it simple?
⑥ Services	What do you actually offer?
⑦ Use cases	Who is this made for? Can I use it the way I want?
⑧ Testimonials	Who has used this? What was their experience?
⑨ Case studies	Can I see a real example of success?
⑩ Comparison	How is this better than other options?
⑪ Pricing	How much does it cost? What do I get?
⑫ FAQ	I still have questions, can you answer them?
⑬ CTA	What's the next step I should take?
⑭ Contact	How do I get in touch with you?
⑮ Blog	Are you an expert in this space?
⑯ Changelog	Are you improving the product? What's new?
⑰ Integrations	Will this work with the tools I already use?
⑱ Security & Trust	Is this safe to use? Can I trust my data with you?
⑲ Newsletter signup	Can I stay updated? Will you send me helpful stuff?
⑳ Footer	Where can I find legal info, policies, or links?

SAVE IT FOR LATER

Start with the questions your own customer actually asks.

OPEN COURSE MEDIA >

User-supplied lander.studio reference sheet; the five questions here are an instructional summary.

Models can now use tools.

From writing an answer to doing a useful task.



Google Docs

Draft a proposal.

Create or update the document.



Google Calendar

Find an available time.

Create the approved event.

Model requests > connected software acts > result is checked.

Connect the tools and grant access first.

Jev: a model for structured decisions.



TypeSafe AI / decision model

Give it context and defined options.

Get a choice, score or probability.

Your software uses the result.

Example: route a customer message

Repair / Sales / Other

Structured does not mean always correct. Test the decision.

An agent connects the steps and checks the result.

Example: prepare a client meeting.

01

Understand

Read the brief.

Ask for missing details.

02

Use tools

Google Docs: agenda.

Google Calendar: free slot.

03

Check & finish

Confirm the details.

Approve, then create.

Inspect the result. Continue, finish or ask for help.

Keep a human in the loop.

Apocalyptic AI scenarios: who would stay in control?



Set limits

Give access only to what it needs.

Approve important actions

A person reviews before it acts.

Keep a way to stop it

Pause, inspect and recover.

AI can help with the work. People keep authority and responsibility.

From a website to a useful product



An offer page people can open.

A calculator they can use.

A request they can send.

Now choose the tools that make your idea work.

Match the tool to the task



Bubble
Custom workflows

Glide
Apps around structured data

Webflow
Websites and content

Airtable
Records and simple workflows

Choose only the tools your first useful workflow needs.

Pick one tool for each product job



Supabase / Airtable
Data and shared records

Clerk / Supabase Auth
User accounts

Vercel / Cloudflare
Hosting

Stripe / RevenueCat
Payments

Choose only the tools your first useful workflow needs.

Measure the action you care about



PostHog / Google Analytics

Product events and journeys

Clarity / Hotjar

Understand friction

Resend

Confirmation emails

Sentry / GitHub

Investigate and review changes

Choose only the tools your first useful workflow needs.

Free allowances can support an early test



Supabase
Database and user limits

PostHog
Analytics event allowance

Cloudflare Workers
Request and compute limits

Resend
Email volume and daily limits

Read the current full plan before choosing.

Build one booking request



One customer. One booking request.

A simple request form.

A saved request.

A confirmation and a next step.

Before you subscribe, could you build your own?



Build the feature you actually need.

A quote calculator.

A simple business tracker.

A landing page or small workflow.

AI can help you build your own version and avoid another subscription.

Your time, hosting and maintenance still count.

Choose tools you can explain



Explain each tool in one sentence

What job does it do?

Why did you choose it?

What would make you replace it?

NOW OPEN THE LAPTOP

From your computer to a working URL.

You will be able to explain every handoff.

What each tool does

Editor + agent	Create and improve the project files.
Git	Record meaningful versions of those files.
GitHub	Share the repository and review changes.
Vercel	Build and host a deployable version.
Browser	Open the page and test the user experience.

Make files > save a version > share > deploy > test

What deployment actually means

Make your project reachable

Put a version on hosting.

Give it a working URL.

Open it from another device.



Meet the project folder



Know the files you are changing

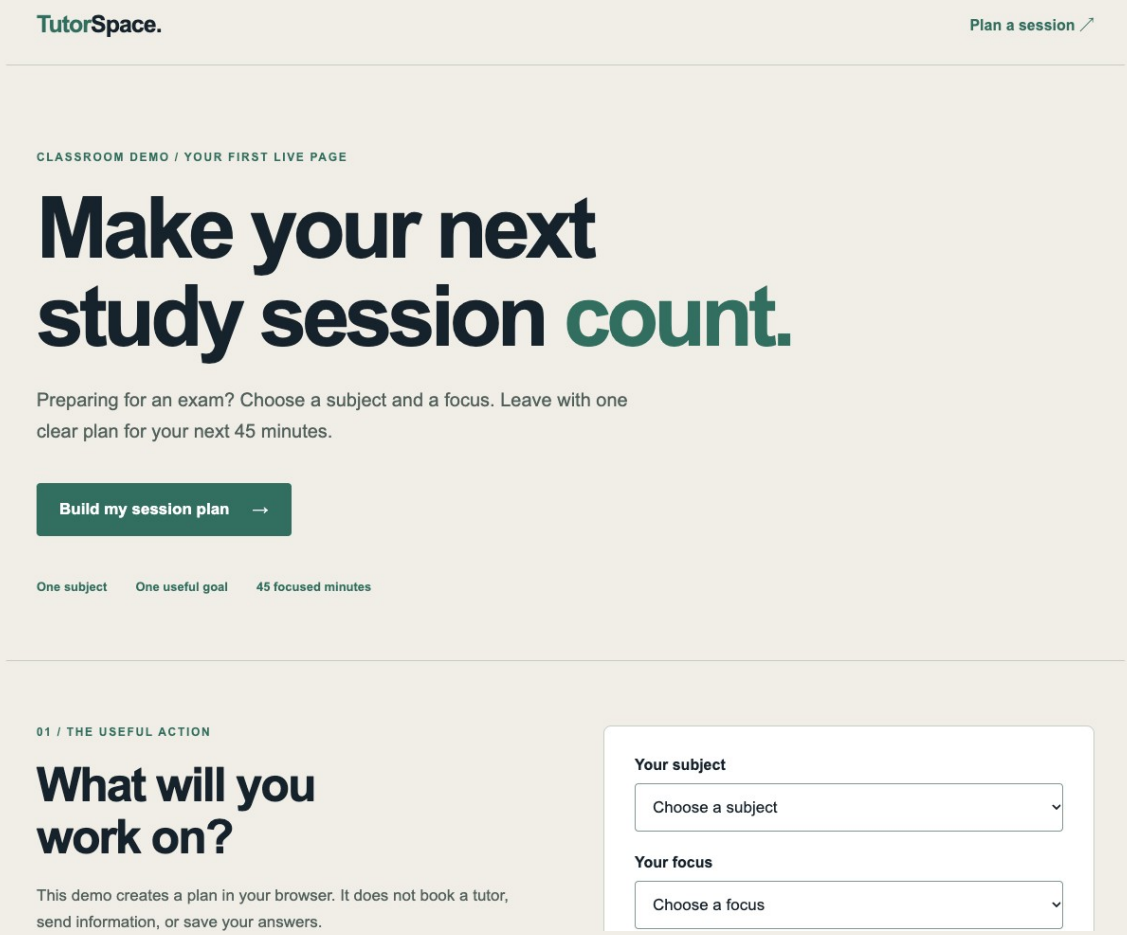
Find the page and its assets.

Read the project instructions.

Identify how it runs.

MEET THE WORKING EXAMPLE

This is the page you will deploy



One clear promise.

One useful interaction.

Three project files.

No API key or database.

Frontend, backend and database

Frontend	The interface and code running in the browser.
Backend	Server-side rules, private credentials and services.
Database	Shared information stored for later use.
Our starter	A page and a local session-plan preview.

A database becomes useful when the product needs shared, persistent data.

Local, preview and production

01

LOCAL

Work on your own computer

02

PREVIEW

Share a candidate version to test

03

PRODUCTION

Serve the approved version to users

Know which environment and commit you are looking at.

Three windows you will use



Three windows, three jobs

Editor: change the files.

Terminal: run commands.

Browser: inspect the result.

Git and GitHub are different

Git repository	Project files plus their recorded history.
Commit	A recorded change with a message and identifier.
GitHub remote	A shared repository hosted online.
Push / pull	Send your commits / bring remote changes in.

A branch gives a change its own path

01

Branch

Name the change before you edit

02

Commit

Record a small explainable step

03

Pull request

Discuss the diff and preview

04

Merge

Bring approved work into main

Small changes are easier to review, test and undo.

Read the diff before trusting the summary

Read what actually changed

Inspect the diff.

Question unrelated edits.

Test the behaviour yourself.





Record and share one small change

```
git switch -c improve-headline
git status
git diff
git add index.html
git commit -m "Clarify the offer"
git push -u origin improve-headline
```

Start in your cloned project.

Inspect first.

Stage the file.

Share the branch.

GIVE THE AGENT A USEFUL BRIEF

One task. Clear constraints. A visible check.

Read this project and explain how it runs.

Change the headline for students preparing for a maths exam.

Keep the existing style and session-plan interaction.

Do not add dependencies or a backend.

Show the diff. Test desktop, mobile and keyboard use.

Report what you tested and what still needs review.

Before you run the project



Understand the project first

Read its README.

Check scripts and dependencies.

Know the command you will run.

First deployment: import your repository

01

GitHub

Create your own repo

Upload the starter

02

Vercel

Add a new project

Import that repo

03

Configure

Check project root

and build settings

04

Deploy

Open the URL

Test the page

Use the project created for this class.

Four settings to recognise

Framework preset	How the platform understands the app.
Root directory	The folder containing this project.
Build command	The command that prepares deployable output.
Output directory	The folder served after the build.

Wrong folder + correct code can still produce a broken deployment.

Our starter needs no build command

Files	index.html, styles.css and app.js at the repo root.
Framework	Choose Other for this plain static site.
Build	Leave the build command empty.
Output	Serve the project root; do not invent a dist folder.

No database, API key or paid AI call is required by the starter.

Next deployment: preview a branch

01

Edit

Change one thing in your branch

02

Push

Share the commit with GitHub

03

Preview

Vercel builds the connected branch

04

Review

Open the new URL and try the action

Read the deployment status and commit before sharing.

Make sure your reviewer can open it

Test the link as a visitor

Open a private browser window.

Check access requirements.

Ask a peer to open the same URL.



Five checks before you call it finished



Finished means usable

Open it on mobile.

Try the main action and links.

Check content, errors and access.

Which layer failed?

Build log	The platform could not prepare the deployment.
Browser console	The page loaded but browser code failed.
Runtime log	A server function failed while handling a request.
Network panel	A requested file or service returned an error.

Common failures and the first useful check

404 / blank page	Is index.html in the configured output folder?
Missing styles	Does the file path match the deployed filename?
Build failed	Read the first error and check the build command.
Old page	Are you viewing the right deployment and branch?
Login wall	Does your reviewer have access to this preview?

Give the agent the error, the environment and the expected result.

Environment variables are deployment settings

Purpose	Provide configuration for the environment.
Example	A server reads an API credential by variable name.
Scope	Development, preview and production can differ.
After changes	Create a new deployment to use the new values.

Browser code is visible to the visitor



A visitor can inspect browser code

Keep secrets on the server.

Use public keys only as intended.

Check permissions as well as keys.

Publish the version you reviewed

01

Approve

Check the diff and preview

02

Merge

Into the configured production branch

03

Build

Wait for the new deployment status

04

Verify

Open production and repeat the task

Record the commit that production is serving.

A URL, a domain and hosting

Deployment URL	The platform’s address for a deployed version.
Custom domain	A name you own, connected to the project.
DNS	Records that direct the name to the right service.
Today	Use the provided Vercel URL for your first release.

Undo a bad release in two places

Restore service	Use a known-good deployment when available.
Fix the source	Revert or correct the offending Git change.
Deploy again	Make the repository and production agree.
Check	Repeat the task and record the recovered version.

Do not leave a broken commit waiting to return on the next deploy.

Know what can create a bill



Know what you are paying for

Check usage limits and billing.

Watch traffic, storage and model calls.

Set alerts or caps where available.

Start an evidence trail from the first session



Keep evidence as you work

Customer + problem.

Decision + experiment.

Result + reflection.

GUIDED LAB

Publish. Change. Review. Recover.

Leave with a live URL and the confidence to change it again.

Write the page before choosing the tool



Write the offer first

Who is it for?

What value does it promise?

What is the next action?

Checkpoint 1: get the starter online

01

Download

Unzip the starter

Read README

02

Open

Try it locally

Find the three files

03

Import

Put it on GitHub

Connect Vercel

04

Show

Open the first URL

with your partner

Expected time: 20-30 minutes.

Checkpoint 2: make a reviewed change

01	02	03	04
Branch	**Change**	**Inspect**	**Preview**
improve-headline	Audience, promise	Read the diff	Push the branch
One purpose	and next action	Test locally	Open the new URL

Expected time: 20-25 minutes.

Checkpoint 3: watch a peer use it

Observe. Don’t coach.

Give a peer the live link.

Watch them try the main task.

Record one change their test suggests.



Checkpoint 4: release and recover

Release	Merge the checked change into the production branch.
Verify	Test the production URL in a fresh session.
Rehearse	Revert a harmless text change in a practice branch.
Explain	Which version would you restore, and why?

Expected time: 15-20 minutes.

Diagnose the problem before asking for a fix

Bring evidence to the problem

What did you expect?

What happened instead?

Which log or screenshot shows it?



SHOW YOUR WORK

Leave with four links and one lesson

01	**Repository**	Where the source and change history live.
02	**Commit**	Which change you made and why.
03	**Preview**	The version you checked before release.
04	**Production**	The public page and commit behind it.

Next course: use an agent to build one useful feature on this foundation.

Can you explain the system without the agent?

1	What is the difference between saving and committing?
2	Which tool stores history, and which hosts the site?
3	Why review a preview before merging?
4	Where should a private API key live?
5	What does a deployment rollback leave unchanged?

Could you explain what you just shipped?



A final show of hands

Can you explain how it went live?

Can you review an AI-made change?

Can you recover a bad release?

NEXT STEPS

One idea can become a content system

Build a repeatable content workflow

Start from one useful idea.

Create for the right channels.

Review, publish and learn.



NEXT STEPS

Give the live page one useful capability.

A quote calculator	Help a customer estimate a cost.
A lead or booking request	Supabase: store the request.
A confirmation email	Resend: send a confirmation.
A payment step	Stripe: connect to checkout.

Choose one extra. Build it. Share the link. Watch someone use it.

Five steps for your own business case



Five questions for your venture

Strategy + value proposition.

Technology + organisation.

Business model: where is the value?

Today is one part of a broader minor

AI fits inside the broader minor

Marketing and customer growth.

Analytics and digital experiences.

Emerging technologies and experiments.



Four assessed parts of the module

15% Participation	Attend, contribute and share useful learning.
30% Digital presence	Website, analytics and active social accounts.
20% Advertising	Set up and optimise Facebook and Google ads.
35% Experimentation	Document the specified technology activities.

At least 5.5 in every assessed part and in the module overall.

Digital presence: build it and keep it active

Build a consistent digital presence

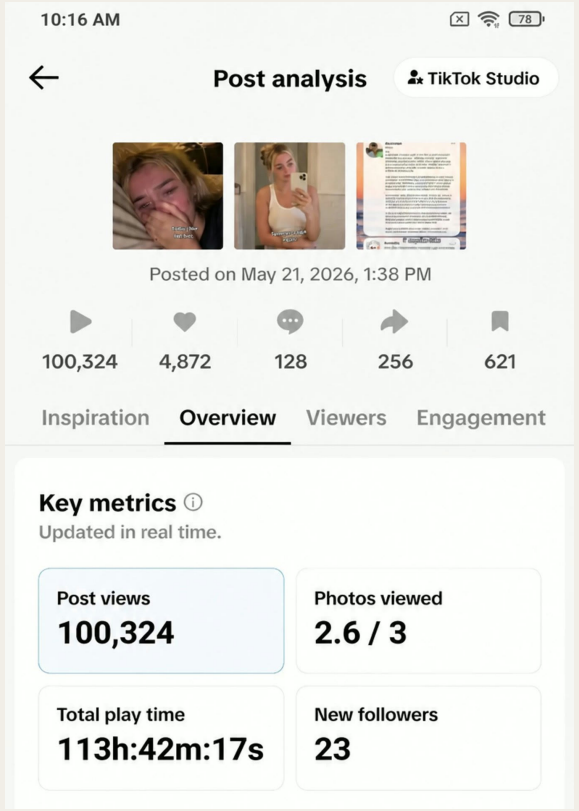
Functional website + Google Analytics.

3 properly set up social accounts.

1 month: at least 12 posts per account.



Advertising: plan, run, learn and improve



Launch. Measure. Improve.

One Facebook Ads campaign.

One Google Ads campaign.

Show setup, results and optimisation.

Experimentation: understand it and show evidence

Understand the action you took

Create and upload 1 NFT.

Trade once; buy and sell crypto.

Explain transactions with screenshots.



Take the five steps into the real world



Take the course outside

Observe a real customer experience.

Map the five experience steps.

Bring one supported insight back.

Plan for 196 hours across the full module

42 hours	Expert input and guided learning.
80 hours	Team work on the business case.
40 hours	Individual study and preparation.
34 hours	Examination work.

7 EC: the work continues between workshops.

SOTA spotlight: AI content creation

Show the range. Judge the result.

Images, video and content workflows.

Look for control and consistency.

Test usefulness for your customer.



READ THE EVIDENCE

A ranking answers only the question it measured.

01	**What is ranked?**	Observed monthly consumer spending in a U.S. data panel.
02	**What is missing?**	Business sales, other countries, and products outside the panel.
03	**What should we ask?**	Which customer problem makes someone pay repeatedly?

CLASS PROMPT

Pick one app. Name the job a customer hires it to do.

Source: a16z / YipitData, Top 100 Gen AI Consumer Apps, 7th edition

OPTIONAL EXTENSION

AGENTIC COMMERCE

When software can spend.

An agent can have payment access, a budget and a job to finish.

HTTP 402: the web asks to be paid.

402

PAYMENT REQUIRED

1997 • HTTP/1.1

Reserved; no payment flow.

x402 proposes one now.

01 Agent requests data

Within a budget you approved

02 Service responds 402

Price + payment instructions

03 Agent pays; work continues

Payment proof unlocks the result

FOUR PROTOCOLS • FOUR DIFFERENT JOBS

x402	paid API access
AP2	payment authorization
ACP	merchant checkout
UCP	discovery to checkout

Sources: RFC 2068 (January 1997) §10.4.3; RFC 9110 §15.5.3; x402; Google AP2 and UCP; Stripe/OpenAI ACP.

Give the agent a wallet with a limit.

EUR 50

AVAILABLE TO THE TASK

The wallet pays for approved services.

Nothing beyond the rules.

MAX PER CALL

No surprise EUR 30 API request.

SESSION BUDGET

Stop when the cap is reached.

APPROVAL GATE

Ask before a product order.

TRANSACTION LOG

Show what was bought and why.

The person sets the authority. The agent acts inside it.

Coinbase Agentic Wallet docs: per-call and per-session limits; Google AP2: verifiable authorization. Example amounts are illustrative.

Agents can buy services from other agents.

YOUR AGENT

Needs a market brief.

SPECIALIST SERVICE

Returns the data.

REQUEST — 402 + PRICE — PAYMENT

EUR 0.20 research **+ EUR 0.40 compute** **= a finished brief**

A paid API can be the seller. A wallet and protocol make the handoff possible.

x402 protocol and Coinbase Agentic Wallet paid-service documentation. Values are illustrative, not service prices.

YOUR BUSINESS, ONE CONCRETE JOB

The agent can research, pay and bring back options.

OWNER'S BRIEF

Find three packaging suppliers.

Max EUR 3 for data. Ask before sample orders.

---

01 **Research supplier options**

02 **Pay for a useful dataset**

03 **Compare price, fit and evidence**

04 **Request approval for a sample order**

The agent brings back a decision. The owner approves the consequential purchase.

Illustrative supplier-research workflow. x402: paid API call; AP2: payment authorization; human approval for any order.



Keep these references close

Git	git-scm.com/book
GitHub	docs.github.com
Vercel	vercel.com/docs
Coding agents	openai.com/codex • code.claude.com/docs
Creative tools	higgsfield.ai/apps

Choose the tool by the task. Keep the result inspectable.