DeltaForceOS

CODEX ROUTE / BEGINNER MODULE 01

DeltaForceOS Curriculum Map: Codex Route

BUILD OUTCOME

Know exactly what to learn first, what to ignore, and how every PDF connects into one operator curriculum.

BEST FOR

Use this PDF when you want exact prompts and clicks for this route instead of general theory.

MODEL TIER

Light for summaries, mid for build planning, heavy only for architecture reviews.

FIRST INSTRUCTION

```
Inspect this repo for the DeltaForceOS Curriculum Map build.
Explain the files a beginner needs: README.md, package.json, src, public, scripts, .env.local, and any
Supabase files.
Do not edit yet. Give me the smallest safe build plan.
```

Before using the agent, make the workspace boring and clear.

01 Open the project folder. If you do not have one yet, create a new folder on Desktop and name it after the lesson.

02 Install Node.js if npm is not available. Beginners can check by opening Terminal and typing npm -v.

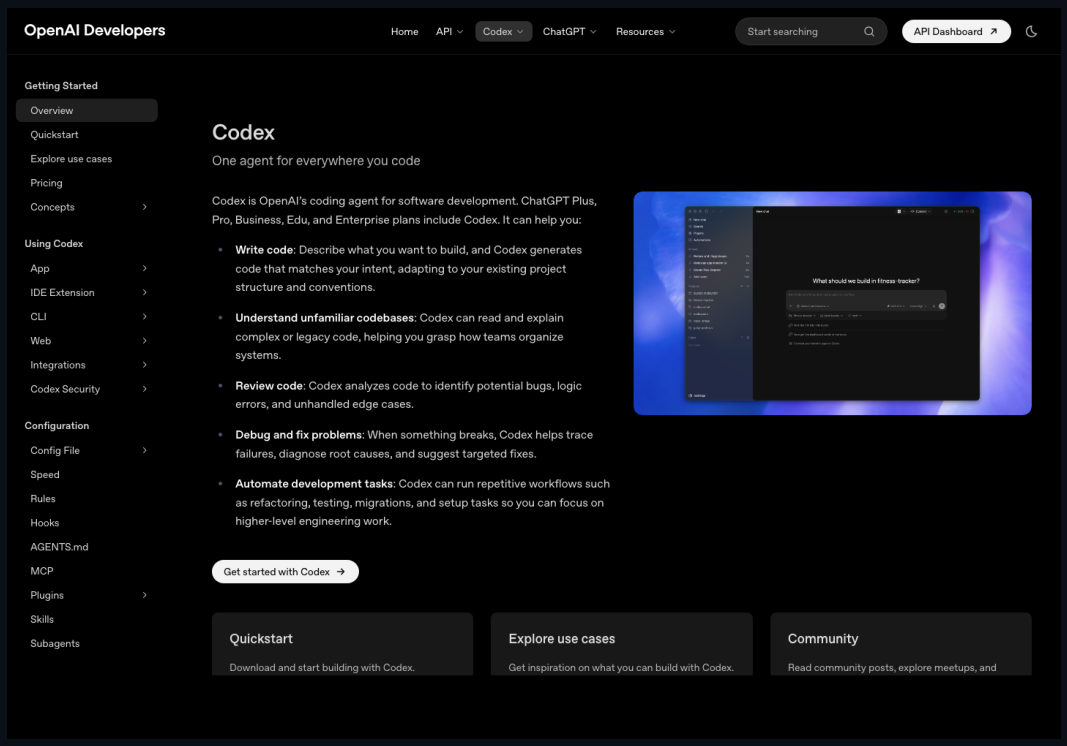
03 Create .env.local in the project root and add only the keys this lesson needs.

04 Confirm .env.local is listed in .gitignore so secrets do not go to GitHub.

05 Open the official dashboard for the first two tools in this lesson and create one tiny test project.

06 Decide where proof will live: Supabase row, screenshot, live URL, or Skool post.

Codex: click path for this route.



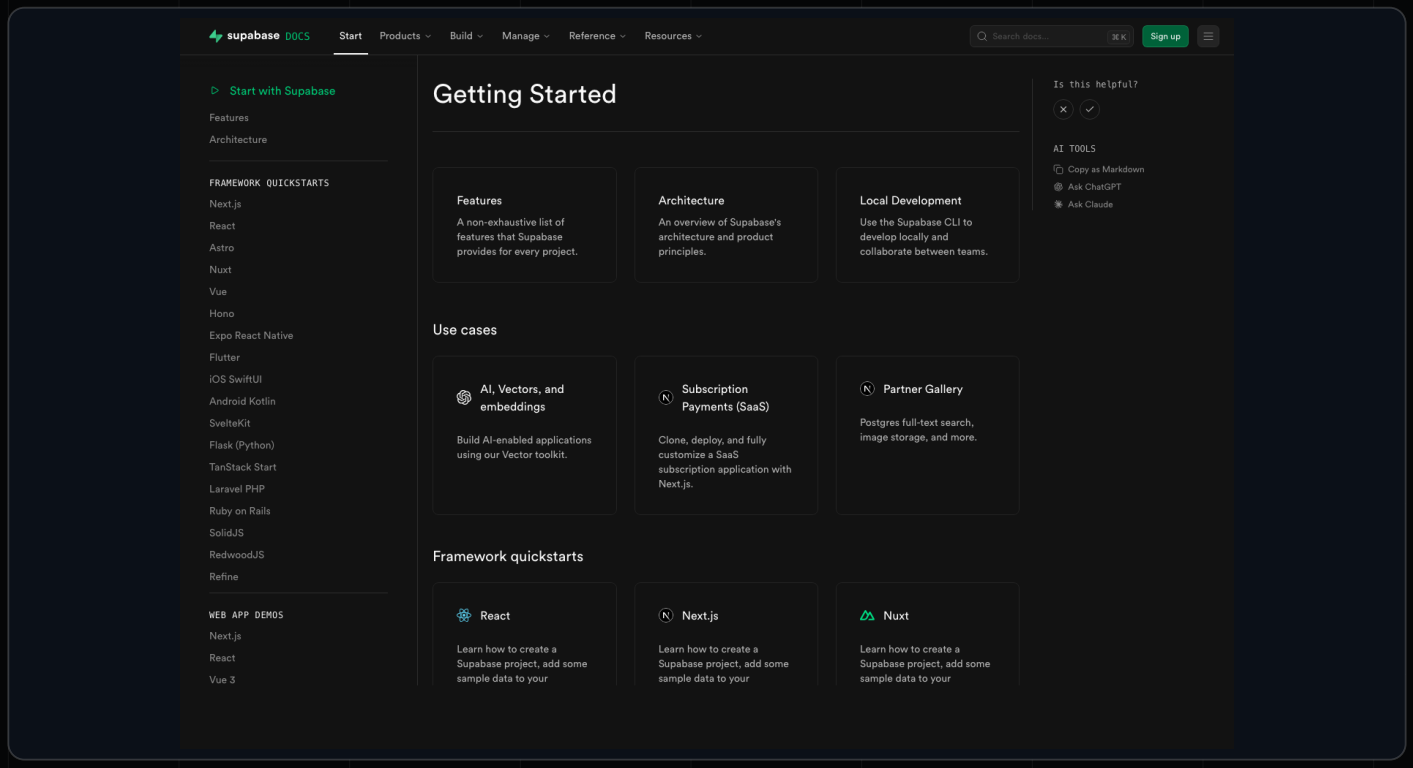
WHERE TO CLICK

Open the project in Codex, let it inspect the repo, ask for a small implementation, then review diffs and tests.

ENV NAMES

```
Use project .env.local files for app secrets; do not paste secrets into prompts.
```

Supabase: click path for this route.



WHERE TO CLICK

Open app.supabase.com, click New project, choose org, name it, save the project URL and anon/publishable key.

ENV NAMES

`NEXT_PUBLIC_SUPABASE_URL, NEXT_PUBLIC_SUPABASE_ANON_KEY, SUPABASE_SERVICE_ROLE_KEY`

Copy this box exactly, then let the agent ask clarifying questions only if needed.

INSPECT PROMPT

```
Inspect this repo for the DeltaForceOS Curriculum Map build.
Explain the files a beginner needs: README.md, package.json, src, public, scripts, .env.local, and any
Supabase files.
Do not edit yet. Give me the smallest safe build plan.
```

BEGINNER CHECK

If the agent proposes a huge rebuild, ask it to shrink the scope to the smallest artifact that proves this lesson.

DO NOT PASTE

Do not paste service-role keys, payment secrets, signing keys, or private client data into the chat.

Copy this box exactly, then let the agent ask clarifying questions only if needed.

BUILD PROMPT

```
Implement the smallest useful version of DeltaForceOS Curriculum Map.
Outcome: Know exactly what to learn first, what to ignore, and how every PDF connects into one operator curriculum.
Tools allowed: Skool, Codex, Next.js, Supabase, Vercel, OpenAI, Claude, Stripe
Keep secrets server-side. Add code comments only where a beginner would get lost.
```

BEGINNER CHECK

If the agent proposes a huge rebuild, ask it to shrink the scope to the smallest artifact that proves this lesson.

DO NOT PASTE

Do not paste service-role keys, payment secrets, signing keys, or private client data into the chat.

TYPE THIS INTO FILES

```
# README.md lesson block
Lesson: DeltaForceOS Curriculum Map
Outcome: Know exactly what to learn first, what to ignore, and how every PDF connects into one operator
curriculum.
Tools: Skool, Codex, Next.js, Supabase, Vercel, OpenAI, Claude, Stripe

# .env.local placeholders
NEXT_PUBLIC_SUPABASE_URL=your_value_here
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_value_here
OPENAI_API_KEY=server_only
ANTHROPIC_API_KEY=server_only

# Verify
npm run build
```

WHERE IT GOES

README.md gets instructions. .env.local gets real local values. Vercel gets production values. Source files get implementation.

Basic English. Click by click. Do not skip ahead.

01 Open Chrome. Go to skool.com/delta-force-network and sign in with your email.

02 Click the Classroom tab in the top nav. Find the lesson card titled Beginner 01: Curriculum Map and open it.

03 Open a new browser tab. Go to github.com. Sign in or click Sign up to create a free account.

04 On your GitHub home page, click the green New button. Name the repository operator-thesis. Click Create repository.

05 On your Mac, press Cmd+Space, type Terminal, and press Enter. A black window opens.

06 Type cd ~/Desktop and press Enter. This puts you on your Desktop folder.

Finish the build. The last step always posts proof in Skool.

01 Copy the clone command from the GitHub page. It looks like git clone https://github.com/yourname/operator-thesis.git. Paste into Terminal and press Enter.

02 Type cd operator-thesis and press Enter. You are now inside the project folder.

03 Type open -e README.md and press Enter. TextEdit opens. Type one sentence: I will use agents to save X hours per week by automating Y workflow for Z audience. Replace X, Y, Z with your real plan.

04 Press Cmd+S to save. Close TextEdit.

05 Back in Terminal, run these three commands one at a time: git add . then git commit -m 'first thesis' then git push.

06 Open Skool again, find the Beginner 01 thread, and post a screenshot of your repo with your thesis sentence visible.

The build is not finished until verification is boring.

01 Can you explain the stack in two minutes without buzzwords?

02 Can a new member find the next lesson and PDF without asking?

03 Can your first agent run twice and produce the same type of output?

04 Run npm run build or the focused test command.

05 Download or open the produced artifact and inspect it visually.

06 Write the failure and fix in README.md or the Skool proof post.

01 Commit safe files to GitHub after reviewing the diff.

02 Open Vercel and deploy a preview first.

03 Add missing env vars under Project Settings -> Environment Variables if the build fails.

04 Run the workflow with fake or test data in preview.

05 Promote or deploy production only after the smoke test passes.

06 Save the live URL and proof artifact in Skool.

AGENT DEPLOY PROMPT

```
Verify the preview build, list any missing env vars, and give me the exact smoke test before production.
```

If something goes wrong, ask for a small repair instead of a total rewrite.

01 Trying to learn 30 tools before shipping one workflow.

02 Keeping agent memory in chat history instead of a real database.

03 Publishing without a rollback path or environment separation.

04 Ask the agent to explain the failing layer: local app, env var, database, external API, prompt, deploy, or auth.

05 Ask for one fix, one test, and one rollback step.

06 Do not let the agent change unrelated files to hide the failure.

SKOOL PROOF

Write your one-sentence operator thesis: 'I will use agents to save X hours per week by automating Y workflow for Z audience.'

WHAT TO POST

Artifact, screenshot or URL, what worked, what broke, exact prompt used, and the next target.

SOURCES

Codex: https://developers.openai.com/codex

Skool: https://help.skool.com/

Next.js: https://nextjs.org/docs

Supabase: https://supabase.com/docs/guides/getting-started

Vercel: https://vercel.com/docs/environment-variables