

Guidelines for the Feynman App Store

How to build and ship an application so the store can clone it, start it, and serve it at <https://www.feynmanlabs.com/apps/{slug}/>.

Upload this file (or paste it) as the first message in a coding chat. Then describe the product in one short paragraph.

You are building a new application that will be pushed to GitLab and deployed on the **Feynman Labs App Store** (<https://www.feynmanlabs.com/apps/{slug}/>).

The store clones the repo and starts a long-running process. It does not rewrite the app to make it boot. If the contract below is wrong, deploy fails.

Use only stacks listed here. If the product could use several, pick the **Recommended** row. Do not invent another framework.

1. Pick a framework (Feynman-supported only)

Choose **one**. Prefer the first row that fits.

Use this	When	Do not use if
FastAPI (Python) - Recommended default	Almost every new app: dashboards, APIs, lab tools, forms, research UIs	You already have a large React/Next codebase you must keep
Static index.html	A single page, brochure, or no backend	You need login, a database, or an API
Go (go.mod)	A small HTTP API with no frontend, or a backend that must be a single binary	You need a rich browser UI in the same repo
Vite + Nitro / Nuxt / TanStack Start	You truly need a modern JS SPA or SSR	You can do the UI in FastAPI + HTML/JS instead
Next.js (standalone only)	The product is already Next, or you must use the Next App Router	This is a greenfield app. FastAPI is safer on this store
Flask (app.py)	You already know Flask and the app is tiny	Greenfield. Use FastAPI
Django (manage.py)	You already have a Django project	Greenfield. Use FastAPI
Docker (Dockerfile)	Java, CLI science tools, or a binary you cannot run as Node/Python/Go	You can run FastAPI or Node natively. Docker is last resort

Default if unsure: FastAPI with [app/main.py](#), [requirements.txt](#), and [feynmanlabs.json](#).

Do not use (the store cannot run these cleanly)

Vercel-only / serverless functions, Remix, SvelteKit, Astro SSR, Create React App with no server, Cloudflare Workers as the only target, vinext unless the user already has a vinext repo, Rust, Ruby, PHP, .NET, monorepos that hide the web app in `apps/` without a root start command, or any host that binds only to `localhost` / a hardcoded port.

2. Non-negotiable contract

- Long-running HTTP server in the foreground. Not a one-shot script. Not a Vercel function.
 - Listen on 0.0.0.0 and honor PORT / \$PORT. Never hardcode 3000, 8010, or 8740.
 - Allow Host 127.0.0.1 and localhost. The store proxy uses those. Read X-Forwarded-Host and X-Forwarded-Proto for the public URL.
 - The public URL is `/apps/{slug}/`. Docs, manuals, and downloads must be `/apps/{slug}/docs/...` (for example `/apps/axiom-vera/docs/DEPLOYMENT_AND_ACTIVATION.md`), never `/docs/...` at the site root. Use `APP_BASE_PATH` or relative links. The proxy already strips the prefix for the process. Do not rewrite `event.path`.
 - Serve a real page at `/`. Serve JSON `{"ok": true}` at `/healthz`. Also serve `/health` if easy.
 - Repo root is the app root. Put `feynmanlabs.json` there. Do not nest the runnable app in `web/` unless the root start command installs and builds that folder first.
 - Commit the lockfile (`requirements.txt` or `package-lock.json` from Node 22).
 - No secrets in git. List required env names in README. Runtime keys go on the App Store card under Edit & env after connect.
 - Config must not crash at import time when `DATABASE_URL` or API keys are missing. `prisma generate` / `next build` / `pip install` must work without a live database.
 - Auth/OAuth (including Better Auth) must use an absolute `https://` origin env (`BETTER_AUTH_URL`). Never pass Vite `BASE_URL` or `/apps/{slug}/` as the auth base URL.
-

3. Recipes (use only the stack you picked)

A. FastAPI (default)

Layout:

```
app/main.py
app/__init__.py
requirements.txt
feynmanlabs.json
README.md
```

`app/main.py` must create `app = FastAPI()`, mount `/`, `/healthz`, and `/health`, and bind only via `uvicorn`:

```
uvicorn app.main:app --host 0.0.0.0 --port $PORT
```

`requirements.txt` must include `fastapi` and `uvicorn`.

`feynmanlabs.json`:

```
{
  "name": "APP_NAME",
  "description": "One sentence.",
  "start_command": "python3 -m pip install -r requirements.txt && python3 -m uvicorn app.mai
n:app --host 0.0.0.0 --port $PORT",
  "health_url": "http://127.0.0.1:$PORT/healthz"
}
```

B. Static page

Repo root has `index.html` (and optional CSS/JS next to it). No build step. The store will serve the folder on `$PORT`.

C. Go

`go.mod` at root. `main` reads `PORT` and listens on `0.0.0.0`. `feynmanlabs.json` `start_command`:

```
PORT=$PORT go run .
```

D. Vite + Nitro / Nuxt / TanStack Start

- Nitro preset: `process.env.NITRO_PRESET || (process.env.VERCEL ? "vercel" : "node-server")`. Never hardcode vercel.
- Vite: base: `process.env.VITE_BASE || "/"`.
- Production start: `node .output/server/index.mjs` on `HOST=0.0.0.0` and `PORT=$PORT`.
- You may add `feynman.deploy.json` only for this stack (schema `feynman-labs-deploy/v1`).
- Never add `feynman.deploy.json` to a Next.js repo.

E. Next.js (only if required)

- output: "standalone" in `next.config`. Do not add Next 16-invalid keys such as `eslint` in `next.config`.
- Do not create `feynman.deploy.json`. That file is a Vite/Nitro contract and will break Next deploys.
- Start must produce and run `.next/standalone/server.js` with `HOSTNAME=0.0.0.0` and `PORT=$PORT`.
- If Prisma exists: `prisma generate` must work without a real `DATABASE_URL`. Do not throw in `prisma.config.ts` when the URL is unset. Put the real URL on the App Store card later.

`feynmanlabs.json` shape:

```
{
  "name": "APP_NAME",
  "start_command": "npx next build && node .next/standalone/server.js",
  "health_url": "http://127.0.0.1:$PORT/healthz"
}
```

F. Flask

`app.py` at root, reads `PORT`, binds `0.0.0.0`. `requirements.txt` includes `flask`.

G. Django

`manage.py` at root. Honor `$PORT`. Prefer this only for an existing Django repo.

H. Docker (last resort)

`Dockerfile` `EXPOSE` and `CMD` must use `PORT` (same numeric port the process listens on). `CMD` stays in the foreground. Do not hardcode a different listen port than `EXPOSE`.

4. GPU / cluster (only if the product needs it)

The store runs on a 4-node DGX Spark cluster. Only add this when the user asked for multi-GPU or distributed training:

```
{
  "gpu": "cluster",
  "nccl": true,
  "start_command": "torchrun --nnodes=$FL_CLUSTER_SIZE --nproc_per_node=1 --rdzv_backend=c10
d --rdzv_endpoint=$MASTER_ADDR:$MASTER_PORT train.py"
}
```

Ordinary web apps do not need this.

5. What to output

- A complete repo that runs locally with the start command in feynmanlabs.json.
 - README.md with: stack chosen and why, start command, required env names, and "meant for the Feynman Labs App Store."
 - No .env with secrets. A .env.example with empty values is fine.
 - After the repo is ready, tell the human: - push to GitLab (main, unless they named another branch) - in the App Store assistant, say the org branch: Watstock, Prepaire, GenetiQ, FAST, or CleverLab - paste the GitLab URL - add any secrets under Edit & env, then deploy - do not ask Reconstruct to turn a Next app into Vite
-

6. Product to build

(If this section is empty, wait for the next user message. If they already described the app above or below this file, start building.)

App name:

Org branch (if known): Watstock | Prepaire | GenetiQ | FAST | CleverLab

What it does:

Who uses it:

Must-have screens / endpoints:

Needs a database? no (default) / yes (name the tables; still do not require DATABASE_URL at import)

Needs GPU / NCCL? no (default) / yes