Byggplan för ZapLearn

# 0) Förutsättningar

• Node 20+, npm 10+  
• Git + GitHub/GitLab  
• Docker Desktop (för senare steg)

# 1) Initiera projekt

1. npm create vite@latest flashcards -- --template react-ts  
2. cd flashcards && git init && git add -A && git commit -m "chore: init"  
3. Lägg till grundläggande konfig (.editorconfig, .nvmrc, README.md)  
 - I tsconfig.json: aktivera "strict": true och sätt paths-alias @/\*

# 2) Installera beroenden

Kärna:  
npm i zustand localforage zod framer-motion react-hook-form @hookform/resolvers react-router-dom  
  
UI & styling:  
npm i -D tailwindcss postcss autoprefixer  
  
PWA, kvalitet & test:  
npm i -D vite-plugin-pwa eslint @typescript-eslint/eslint-plugin @typescript-eslint/parser eslint-config-prettier vitest jsdom @testing-library/react @testing-library/user-event @types/node  
  
Initiera Tailwind: npx tailwindcss init -p (koppla in index.css).

# 3) Lägg in shadcn/ui

1. npx shadcn-ui@latest init  
2. Lägg till komponenter du behöver direkt:  
 npx shadcn-ui@latest add button card input textarea dialog sheet select table progress badge toast  
3. Uppdatera tailwind.config.ts → content inkluderar ./src och ./components.  
4. Lägg till dark mode-toggle (finns exempel i shadcn/ui docs).

# 4) Mappstruktur

src/  
 app/ # layout, router, providers  
 components/ # UI-komponenter (shadcn) + egna  
 features/  
 decks/ # deckRepo (IndexedDB), deckService, deckStore (Zustand)  
 train/ # repetitionlogik, vyer  
 lib/ # helpers (uuid/hash, sanitize, hotkeys)  
 pages/ # Home, Train, Edit, Manage  
 types/ # Deck/Card typer + Zod schema  
 styles/ # tailwind.css etc  
 main.tsx, App.tsx  
public/  
 runtime/ # config.json (skapad av docker-entrypoint)

# 5) Typer & schema (Zod-validering)

• Skapa types/deck.ts med TS-typer och Zod-schema (DeckSchema) för uppladdad JSON.  
• Visa tydliga fel i UI vid ogiltiga filer (rad/kort om möjligt).

# 6) Persistens (IndexedDB)

• features/decks/deckRepo.ts med localforage: saveDeck, listDecks, deleteDeck, getDeck.  
• Separera stores: decks, progress, settings. Lägg schemaVersion i metadata för framtida migreringar.

# 7) Laddning av data

• deckService.ts:  
 - loadSeedDeckFromRuntime(): läser /runtime/config.json om det finns.  
 - importDeckFromFile(file): parse + validera + spara.  
 - importDeckFromUrl(url): (senare) med ETag/If-None-Match.  
• UI: Uploader (drag & drop + knapp) och DeckManager (lista, byt, export).

# 8) State (Zustand)

• deckStore.ts: activeDeckId, setActiveDeck, cards, filters, shuffleSeed.  
• progressStore.ts: per cardId: bucket (0/1/2), ease, interval, dueAt. Actions: markCorrect/markWrong/resetProgress.

# 9) Routing (React Router)

• Installera: react-router-dom (ingår ovan).  
• I main.tsx: wrappa appen i <BrowserRouter>.  
• Rekommenderad struktur:  
 / = Home (importera/hantera lekar)  
 /train/:deckId = Övningsläge (flip + rätt/fel, SR)  
 /edit/:deckId = Editor (redigera fråga/svar, taggar, svårighet)  
 /manage = Lista alla lekar, exportera, nollställ progress  
• Lägg till shadcn Navbar/Sidebar för navigation.

# 10) Kort & flip-UI

• Bygg Flashcard-komponent med shadcn Card + Framer Motion (flip 0°/180°).  
• Tangentbordsstöd: Space/Enter = vänd, ←/→ = föregående/nästa, 1/2 = fel/rätt.  
• Tillgänglighet: role="button", tabIndex, aria-attributes, respektera prefers-reduced-motion.

# 11) Övningsläge (MVP → enkel SR)

• 3 buckets: nytt → lärs → behärskat.  
• Rätt: öka intervallet (t.ex. +1h → +1d → +3d). Fel: flytta till lägre bucket.  
• Senare: SM-2 (ease, repetitions, interval) för mer exakt schemaläggning.

# 12) Editor

• EditorPanel (Dialog/Sheet) med Input/Textarea (shadcn) och RHF+Zod.  
• Lägg till/ta bort/ändra kort, autospara (debounce 300–500 ms).  
• Forka seed-lek till lokal kopia innan redigering (read-only seed).

# 13) Filter, sök & vyer

• Filter: kategori, taggar, svårighet (Select/Checkbox/Badge från shadcn).  
• Sökfält: Input + debounce.  
• Vyer: Bläddra, Öva, Quiz (fri text/multival senare).

# 14) Import/Export

• Exportera deck (+ valfritt progress) som JSON.  
• Importera: ersätt allt eller behåll progress om card.id matchar (fuzzy-match vid behov).  
• Sanera ev. markdown/HTML (dompurify).

# 15) PWA (offline)

• vite-plugin-pwa: cachea statiska assets och ge installbar app.  
• Hantera /runtime/config.json med stale-while-revalidate (inte hårdcacha seed-konfig).

# 16) Testning

• Enhet: Zod-validering, repetitionlogik, repositories (mocka localforage).  
• UI: React Testing Library – flip, tangentbord, importfel, editor autosave.

# 17) Kvalitet

• ESLint + Prettier, CI för lint/typecheck/test.  
• A11y-pass: fokus, kontrast, aria.  
• Säkerhet: CSP (minst default-src 'self'), X-Content-Type-Options: nosniff.

# 18) Bygg för produktion

npm run build (Vite bygger till dist/)  
npm run preview

# 19) Docker (seed + statisk server)

Dockerfile (2-steg: build → Nginx). docker-entrypoint.sh genererar runtime/config.json med DECK\_URL.  
Nginx SPA-routing: try\_files $uri /index.html;  
Körningsexempel:  
docker build -t viktor/flipdeck:0.1.0 .  
docker run --rm -p 8080:80 -e DECK\_URL="/data/projektledning.json" -v $(pwd)/decks:/usr/share/nginx/html/data viktor/flipdeck:0.1.0

# 20) CI/CD (rekommenderat)

• GitHub Actions: install → lint → typecheck → test → build.  
• Release: bygg & pusha Docker-image (tags: latest + git-tag).

# 21) Definition of Done (MVP)

[ ] UI-uppladdning med Zod-validering  
[ ] Visa/flip kort (tangentbord + touch)  
[ ] IndexedDB-lagring (lekar + progress)  
[ ] Enkel 3-bucket repetition  
[ ] Editor (Dialog/Sheet) + autosave  
[ ] React Router (Home/Train/Edit/Manage)  
[ ] Export/import  
[ ] Docker-image med seed via DECK\_URL

# 22) Nästa steg (post-MVP)

• Multival & fritext-quiz (auto-rättning)  
• Statistik (per kategori/tagg)  
• URL-import + "Ladda om från källa" (ETag)  
• Anki/CSV-import  
• AI-assist: generera kort från text/URL