

learntoaction – Module-wise Development Prompts & Local Setup Guide

Audience: Solo founder developing locally (no Docker), vibe coding, architecture locked

Purpose: This document tells you *exactly what to build, in what order*, and gives **copy-paste AI prompts** you can use module by module without breaking the system.

0. Local Development Setup (One-Time)

0.1 System Requirements

- Node.js **20 LTS**
 - PostgreSQL **15+** (local)
 - Git
 - Any code editor (VS Code recommended)
-

0.2 Project Structure (Keep This Simple)

```
learntoaction/  
├─ api/           # NestJS backend  
├─ web/           # React frontend (student first)  
├─ prisma/        # Prisma schema & migrations  
└─ docs/
```

0.3 Backend Setup (NestJS)

```
npm i -g @nestjs/cli  
nest new api  
cd api  
npm install @prisma/client prisma zod jsonwebtoken
```

Create `.env.local`:

```
DATABASE_URL=postgresql://user:password@localhost:5432/learntoaction
JWT_SECRET=local_dev_secret
NODE_ENV=local
```

Initialize Prisma:

```
npx prisma init
```

0.4 Frontend Setup (React)

```
npm create vite@latest web -- --template react-ts
cd web
npm install
npm install axios zustand react-hook-form
```

1. Module 1 — Core Data Model (FOUNDATION)

Goal

Define the **unbreakable backbone** of the system.

Modules Covered

- Workspace
- Worksheet
- Field
- Student
- Response

AI Prompt

```
You are a senior backend engineer.
Create a Prisma schema for a multi-tenant SaaS called learntoaction.
```

```
Requirements:
```

- workspace_id present in all tables
- Worksheet stored as JSON schema
- Field keys are immutable and unique per workspace
- Responses support autosave and resume
- Soft delete via deleted_at

- Timestamps everywhere

Output only Prisma schema.

Deliverables

- `schema.prisma`
- First migration

DO NOT proceed until migrations run cleanly.

2. Module 2 — Worksheet JSON Schema (CONTRACT)

Goal

Define the **canonical worksheet structure** used by editor, runtime, analytics.

Must Support

- Blocks
- Layout (columns)
- Field references
- Progress tracking

AI Prompt

Design a JSON schema for an interactive worksheet system.

Constraints:

- Worksheet is one page
- Blocks are ordered
- Layout supports columns (no nested rows)
- Each input block references a `field_key`
- Schema must be forward compatible

Return example JSON + explanation.

Deliverables

- `worksheet.schema.json`
- One real example worksheet JSON

This schema is LOCKED after this step.

3. Module 3 — Student Runtime (THE PRODUCT)

Goal

A student can complete, leave, resume, and finish a worksheet.

Backend Responsibilities

- Create anonymous student
- Persist responses
- Resume by student_id + worksheet_id

AI Prompt (Backend)

Create REST endpoints for a worksheet runtime.

Endpoints:

- GET /ws/:slug
- POST /responses/autosave
- GET /responses/resume
- POST /responses/complete

Assume Prisma + NestJS.

Include DTOs and service logic.

AI Prompt (Frontend)

Create a React worksheet renderer.

Requirements:

- Render blocks from JSON
- Bind inputs to field_key
- Autosave on blur
- Persist student_id in localStorage
- Show progress percentage

Use functional components.

Deliverables

- Student can resume reliably
- Autosave works

If this fails, STOP and fix.

4. Module 4 — Field System (IDENTITY)

Goal

Fields are globally reusable and bind data across worksheets.

Rules

- field_key auto-generated
- Not editable
- Unique per workspace

AI Prompt

Design a field generation system.

Input:

- Label text

Output:

- Field name
- Field key in dot notation

Ensure uniqueness per workspace.

Include collision handling.

Deliverables

- Field table
 - Field creation logic
-

5. Module 5 — Teacher CRUD (MINIMAL)

Goal

Teacher can manage worksheets (no fancy UI yet).

Features

- Create worksheet
- Edit JSON
- Draft / Publish

AI Prompt

Create a minimal teacher dashboard for managing worksheets.

Requirements:

- List worksheets
- Draft vs Published
- Edit worksheet JSON

No permissions, no folders.

6. Module 6 — Workbook (AFTER VALIDATION)

Goal

Multi-page learning flow.

Rules

- Workbook = ordered worksheet snapshots
- Import = copy, not sync

AI Prompt

Design a workbook system that contains ordered worksheet pages.

Constraints:

- Pages reference worksheet snapshots
- Reordering allowed
- No live sync with source worksheet

Provide DB model + API.

7. Module 7 — Analytics v1 (EVENTS ONLY)

Goal

Measure learning, not vanity metrics.

Events

- worksheet_viewed

- autosave
- completed

AI Prompt

Design an event-based analytics system for worksheets.

Requirements:

- Append-only events table
- Async aggregation
- Per worksheet metrics

Provide schema + example queries.

8. Development Rules (READ THIS)

Allowed

- Hardcoded styles
- Minimal UI
- Skipped edge cases

Not Allowed

- Changing field keys
- Mutating responses
- Tight coupling UI ↔ DB

9. Progress Checklist

- ☐ Student completes worksheet
- ☐ Resume works after refresh
- ☐ Field data persists
- ☐ Completion locks worksheet

If all checked → product is real.

10. Final Principle

Momentum without regret.

Build only what proves learning → action → persistence. Everything else is secondary.