From Vibe Code to Production-Ready

Spec-Driven Development for AI-Assisted Builders

by DoctorEw | SLIM VERSION (36 slides)

Press Space to begin your journey →

From PRD to Production

Spec-Driven Development Workshop

Transform product requirements into production-grade code
using AI-assisted frameworks that actually work

DoctorEW's Guide to Intentional Building

Today's Journey

The Problem

- ❌ Vibe coding gives you databaseless apps
- ❌ Feature drift and hallucinations
- ❌ Context loss mid-project
- ❌ Unmaintainable AI-generated code

The Solution

- ✅ **Spec-Driven Development** with frameworks
- ✅ **Atomic breakdown:** Organisms → Molecules → Atoms
- ✅ **8 Prompt Frameworks** for different task types
- ✅ **PRD → SPEC → Guardrails → AC** pipeline

What You'll Walk Away With

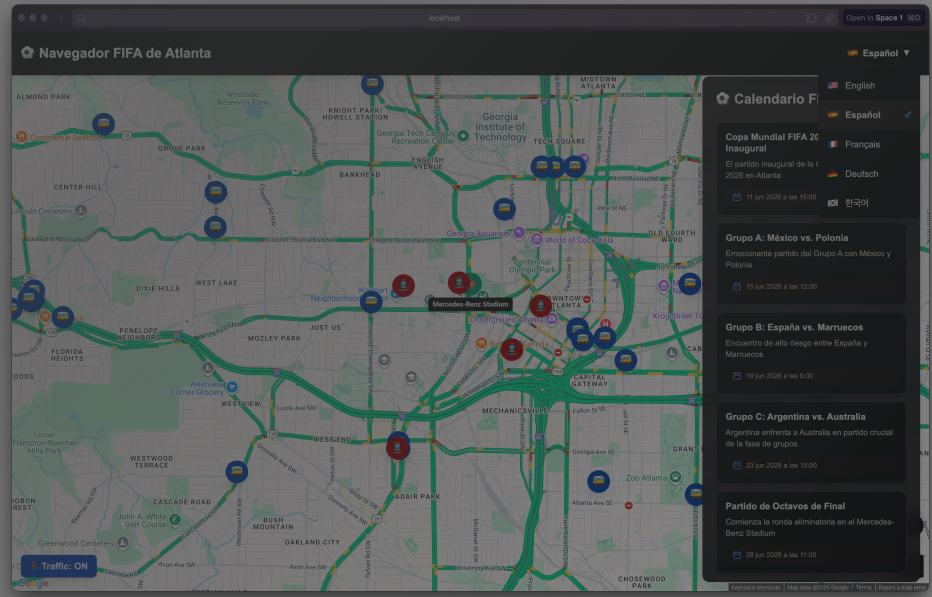
Working FIFA Navigator app + the methodology to build anything with AI

What You'll Build Today

FIFA Atlanta Navigator

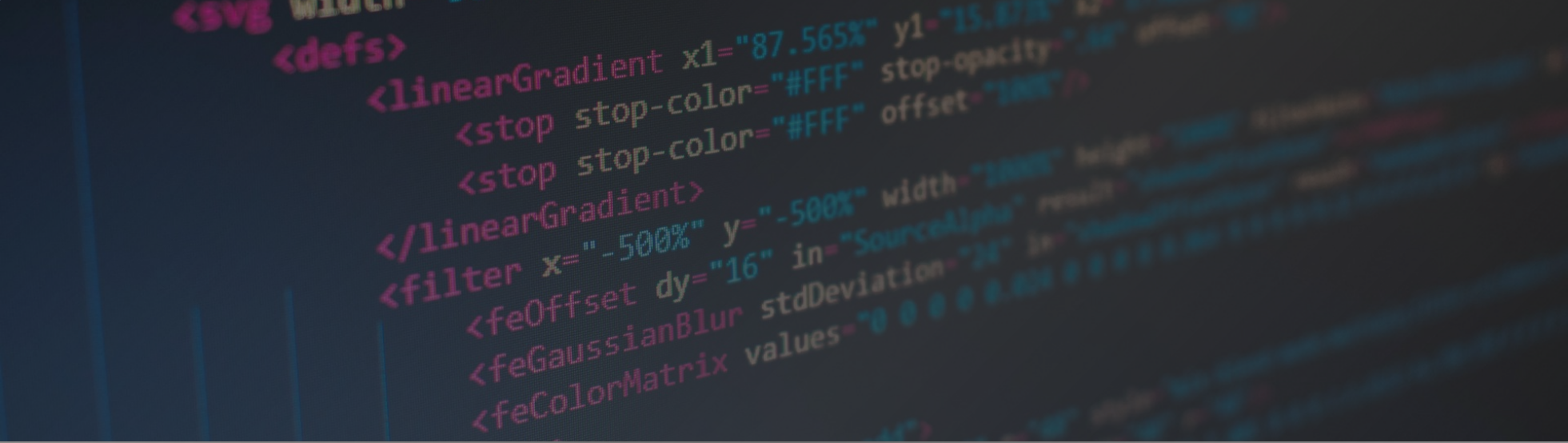
A production-ready Next.js app with:

- 🗺️ **Google Maps** with live traffic
- 🚌 **MARTA Transit** (Bus & Train real-time)
- ⚽ **FIFA Schedule** with event details
- 🌍 **Multi-language** (EN, ES, DE, KO, FR)
- 📱 **Mobile-responsive** design



What is Vibe Coding?

Building by feel, not by spec


<defs>
<linearGradient x1="87.565%" y1="15.87%"
<stop stop-color="#FFF" stop-opacity="
<stop stop-color="#FFF" offset="100%"/>
</linearGradient>
<filter x="-500%" y="-500%" width="1000%"
<feOffset dy="16" in="SourceAlpha"
<feGaussianBlur stdDeviation="24"
<feColorMatrix values="0 0 0 0 0.454 0 0 0 0 0

When Vibe Coding Actually Works

✅ Good For

- Quick prototypes
- Throwaway demos
- Learning/exploration
- Solo hackathons
- Proof of concepts

"Move fast, break things"
when breaking things is fine

❌ Bad For

- Production applications
- Team projects
- Maintainable code
- Scalable systems
- Client deliverables

"Move fast, break things"
when things can't break

🧙‍♂️ *"Know your intent, choose your method"*

The AI Trust Problem

> "Never trust anything that can think for itself if you can’t see where it keeps its brain."
>
> — Arthur Weasley

Vibe Coding’s Hidden Danger

When you ask AI to "just make it work" without a spec, you’re trusting a black box you don’t understand.

What is Spec-Driven Development?

Intentional architecture + AI assistance

Vibe vs Spec: The Contrast

Aspect	🎨 Vibe Coding	📐 Spec-Driven Development
Start Point	"Make me an app"	Product Requirements Document (PRD)
Process	Conversational iteration	Structured frameworks (8 types)
Architecture	Emergent (maybe)	Intentional from day 1
AI Role	Creative partner	Precision tool
Output	Prototype	Production-ready
Maintainability	Low (context lost)	High (documented decisions)
Scalability	Hope-based	Plan-based
Debuggability	"It broke somewhere?"	Clear component boundaries

Why Spec-Driven Matters

The Problems It Solves

🎯 Feature Drift

AI hallucinates features you never asked for

🧠 Context Loss

After 20 prompts, AI forgets the original goal

🔧 Unmaintainable Code

"It works but I don’t know why"

🚫 No Database Syndrome

Missing fundamental architecture

The Benefits You Get

✅ Clear Boundaries

Every component has a defined purpose

✅ Reusable Patterns

8 frameworks = 8 reusable approaches

✅ Team Alignment

Specs are documentation

✅ Production-Ready

Not "maybe works" but "definitely works"

The PRD → PROD Pipeline

Four phases, one goal: Production-ready code

Phase 2: SPEC (Technical Specification)

What Is It?

The **technical blueprint** generated from the PRD using the **C.R.A.F.T** framework

Contains

- Architecture decisions
- Tech stack choices
- Component structure
- API integrations
- Data models

Generated By

AI using the PRD as input

Example: FIFA Navigator SPEC

```
## Architecture
Next.js 15 App Router + TypeScript

## Components
- MapView (Google Maps React)
- TransitLayer (MARTA API integration)
- EventList (FIFA schedule display)
- LanguageSwitcher (i18n)

## APIs
- Google Maps JavaScript API
- MARTA Bus API (direct)
- MARTA Train API (Codespaces proxy)

## Database
Prisma + SQLite (dev) / PostgreSQL (prod)

## Deployment
Vercel with environment detection
```

Phase 3: Guardrails (Do’s & Don’ts)

What Are They?

Constraints and best practices generated using the **D.E.C.I.D.E** framework

Purpose

- Prevent common mistakes
- Enforce architecture decisions
- Guide AI code generation
- Maintain consistency

Generated By

AI using SPEC + PRD as input

Example: FIFA Navigator Guardrails

✅ MUST DO

- Use `NEXT_PUBLIC_*` for client-side env vars
- Detect Codespaces environment for Train API
- Use pnpm (not npm/yarn)
- Follow Next.js App Router patterns
- All strings use i18n (no hardcoding)

❌ MUST NOT DO

- Modify .devcontainer files
- Use Pages Router patterns

Phase 4: Acceptance Criteria (Success Metrics)

What Is It?

Measurable verification steps generated using the **M.A.P** framework

Purpose

- Define "done"
- Enable testing
- Validate features
- Document success

Generated By

AI using SPEC + Guardrails as input

Example: FIFA Navigator AC

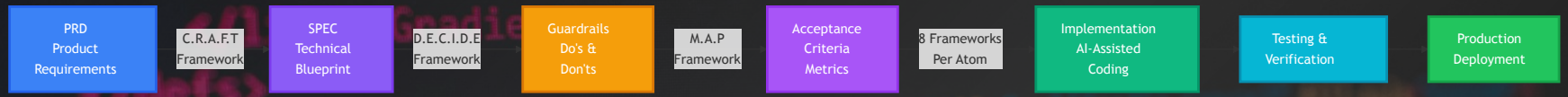
Map Component

- [ ] Map renders centered on Mercedes-Benz Stadium
- [ ] Traffic layer displays current conditions
- [ ] Map is interactive (pan, zoom work)
- [ ] Mobile: Map takes full viewport

Transit Data

- [ ] Blue markers show bus locations
- [ ] Red markers show train locations

The Complete Pipeline



🧙‍♂️ *"Structure before code. Intent before action."*

Atomic Design for Milestones

Organisms → Molecules → Atoms

Or as developers know it: Epics → Stories → Tickets

Atomic Design Mapping

Atomic Level	Dev Term	Notation	Example	Duration
Organism	Epic / Milestone	MS-01, MS-02	"Core Navigation Features"	3-6 hours
Molecule	Story / Feature	MS-01.01, MS-01.02	"Google Maps Integration"	45-90 min
Atom	Ticket / Task	MS-01.02-A, MS-01.02-B	"Install Maps Library"	10-30 min

Organism

Grouping mechanism

Molecule

Feature scope

Atom

Implementation unit

Example: FIFA Navigator Breakdown

```
🧬 MS-01: Core Navigation Features (Organism)
|
├── 🧪 MS-01.01: Next.js Project Setup (Molecule)
|   ├── ⚛️  MS-01.01-A: Initialize Next.js with TypeScript
|   ├── ⚛️  MS-01.01-B: Configure Tailwind CSS
|   └── ⚛️  MS-01.01-C: Set up project structure
|
├── 🧪 MS-01.02: Google Maps Integration (Molecule)
|   ├── ⚛️  MS-01.02-A: Install @vis.gl/react-google-maps
|   ├── ⚛️  MS-01.02-B: Create MapView component
|   ├── ⚛️  MS-01.02-C: Add traffic layer
|   └── ⚛️  MS-01.02-D: Center on Mercedes-Benz Stadium
|
├── 🧪 MS-01.03: MARTA API Integration (Molecule)
|   ├── ⚛️  MS-01.03-A: MARTA Bus API (direct)
|   ├── ⚛️  MS-01.03-B: MARTA Train API (with Codespaces proxy)
|   └── ⚛️  MS-01.03-C: Animated transit markers
|
└── 🧪 MS-01.04: FIFA Schedule Display (Molecule)
    ├── ⚛️  MS-01.04-A: Create mock schedule data
    ├── ⚛️  MS-01.04-B: EventList component
    └── ⚛️  MS-01.04-C: EventCard component with click handler
```

Setting Up Your Environment

Fork, launch, build

Step 2: Choose Your AI Assistant

Claude vs Gemini: The Tradeoff

Feature	Claude Code	Gemini Code Assist
Context Window	200K tokens	**2M tokens (10x larger!)**
Code Quality	**Excellent**	Very Good
Cost	$$	**Free tier available**
Speed	Fast	Fast
Best For	Production code	Exploration, large codebases

Step 3: Configure API Keys

Environment Variables Setup

```
# In your Codespace terminal
cd fifa-traffic-demo
cp .env.example .env
```

Then edit `.env` with your keys:

```
# Google Maps (required)
NEXT_PUBLIC_GOOGLE_MAPS_API_KEY=your_key_here

# MARTA APIs (required)
MARTA_API_KEY=your_key_here
MARTA_TRAIN_API_KEY=your_key_here

# Stadium location (already configured)
NEXT_PUBLIC_STADIUM_NAME="Mercedes-Benz Stadium"
NEXT_PUBLIC_STADIUM_LAT="33.754542"
NEXT_PUBLIC_STADIUM_LNG="-84.402492"
```

The 8 Prompt Frameworks

Different tasks need different structures

Step 2: Load Meta-Framework Guide

The Framework Selector

```
# After loading Master Guide
Read docs/PromptTemplates/00-META-FRAMEWORK-GUIDE.md
```

What This Does

The Meta-Framework Guide provides:

- ✅ Decision logic for selecting the right framework
- ✅ Templates for all 8 frameworks
- ✅ Examples of when to use each one
- ✅ Quality checklist for generated prompts

The 8 Frameworks

S.P.A.R.K

Educational Setup

- Situation, Purpose, Audience, Roadmap, Knowledge
- Use for: Project initialization, learning-focused tasks

F.O.C.U.S

Constraint Solving

- Frame, Objective, Constraints, Users, Steps
- Use for: Environment-specific problems (Codespaces proxy!)

C.R.A.F.T

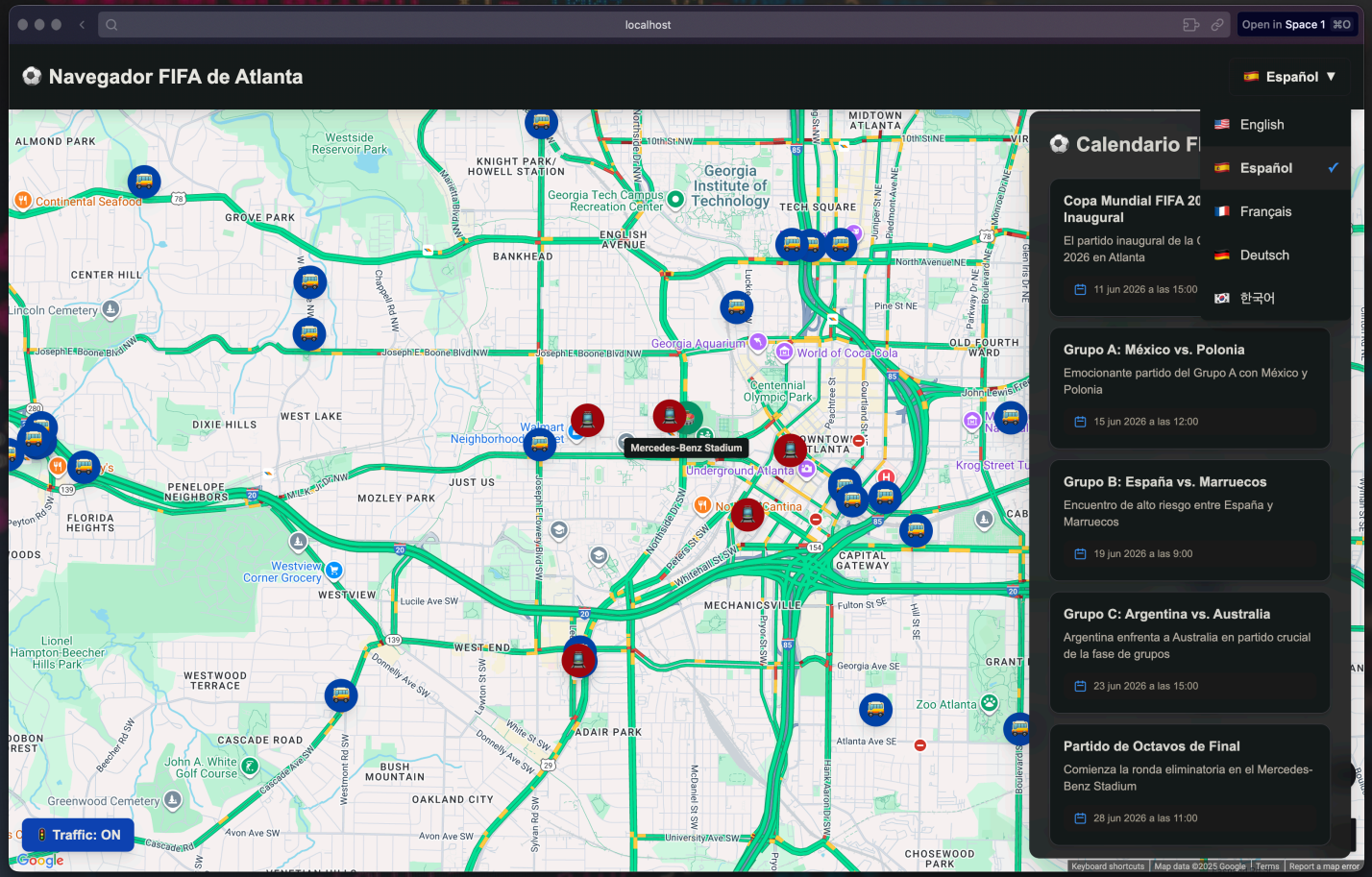
Technical Integration

- Context, Role, Action, Format, Tone

I.D.E.A

Creative/UX Design

- Issue, Details, Example, Action


localhost
Open in Space 1 ⌘O
Navegador FIFA de Atlanta
Español
English
Español
Français
Deutsch
한국어
Calendario F
Copa Mundial FIFA 20
Inaugural
El partido inaugural de la C
2026 en Atlanta
11 jun 2026 a las 15:00
Grupo A: México vs. Polonia
Emocionante partido del Grupo A con México y Polonia
15 jun 2026 a las 12:00
Grupo B: España vs. Marruecos
Encuentro de alto riesgo entre España y Marruecos
19 jun 2026 a las 9:00
Grupo C: Argentina vs. Australia
Argentina enfrenta a Australia en partido crucial de la fase de grupos
23 jun 2026 a las 15:00
Partido de Octavos de Final
Comienza la ronda eliminatoria en el Mercedes-Benz Stadium
28 jun 2026 a las 11:00
Mercedes-Benz Stadium
Traffic: ON
Georgia Institute of Technology
Georgia Aquarium
World of Coca-Cola
Centennial Olympic Park
Underground Atlanta
Zoo Atlanta
Georgia Tech Campus Recreation Center
Westside Reservoir Park
Lincoln Cemetery
Greenwood Cemetery
John A. White Golf Course
Lionel Hampton-Beecher Hills Park
Westview Corner Grocery
Continental Seafood
ALMOND PARK
GROVE PARK
CENTER HILL
KNIGHT PARK/ HOWELL STATION
BANKHEAD
ENGLISH AVENUE
TECH SQUARE
MIDTOWN ATLANTA
OLD FOURTH WARD
DIXIE HILLS
WEST LAKE
PENELOPE NEIGHBORS
FLORIDA HEIGHTS
MOZLEY PARK
JUST US
WESTVIEW
WEST END
WESTWOOD TERRACE
CASCADE ROAD
BUSH MOUNTAIN
OAKLAND CITY
ADAIR PARK
MECHANICSVILLE
CAPITAL GATEWAY
CHOSEWOOD PARK
Keyboard shortcuts
Map data ©2025 Google
Terms
Report a map error

Guardrails & Acceptance Criteria

Boundaries and proof of success

Guardrails Deep Dive

What Are Guardrails?

Explicit **constraints** that prevent AI from generating problematic code

Purpose

- 🚫 Prevent anti-patterns
- ✅ Enforce architecture decisions
- 📐 Maintain consistency
- 🛡️ Protect critical infrastructure

✅ MUST DO (Positive Guardrails)

```
- Use `NEXT_PUBLIC_*` for client env vars
- Detect Codespaces for Train API proxy
- Use pnpm (never npm/yarn)
```

❌ MUST NOT DO (Negative Guardrails)

```
- Never modify .devcontainer files
- Never use Pages Router
- Never hardcode coordinates
```

Acceptance Criteria Deep Dive

What Are Acceptance Criteria?

Measurable, testable checkboxes that define when a feature is complete

Purpose

- ✅ Define "done" unambiguously
- 🧪 Enable testing
- 📊 Measure progress
- 📝 Document expected behavior

Characteristics

- **Specific:** "Map centers on 33.754542, -84.402492"
- **Measurable:** "Loads in < 3 seconds"
- **Testable:** Can be verified true/false
- **Observable:** Can see/measure the result

Acceptance Criteria Examples

FIFA Navigator AC Checklist

🗺️ Map Component

- [ ] Map renders centered on Mercedes-Benz Stadium (33.754542, -84.402492)
- [ ] Traffic layer displays current road conditions
- [ ] Map is interactive (pan, zoom, rotate work)
- [ ] Mobile: Map takes full viewport minus header
- [ ] Desktop: Map takes 60% width, schedule takes 40%
- [ ] Loads in < 3 seconds on 3G connection

🚌 Transit Data

- [ ] Blue markers show current bus locations
- [ ] Red markers show current train locations
- [ ] Markers update automatically every 30 seconds
- [ ] Codespaces environment: Console logs "Using proxy URL"
- [ ] Production environment: Console logs "Direct connection"

Milestone Review & Retrospective

Closing the loop

Milestones ↔ Atomic Design

How They Correspond

Organism = Milestone

MS-01: Core Features

Represents a major deliverable or release phase

Tracks:

- Epic-level progress
- Team capacity
- Release readiness

Molecule = Sub-Milestone

MS-01.02: Google Maps

Represents a user-facing feature or technical capability

Tracks:

- Feature completion
- Integration status

Atom = Task Checkpoint

MS-01.02-A: Install library

Represents an atomic unit of work

Tracks:

- Individual commits
- AC verification
- Technical debt

Key Takeaways

What You Learned

🎨 Vibe vs Spec

- Vibe coding = prototyping
- Spec-Driven = production
- Know when to use each

📐 The Pipeline

- PRD → SPEC → Guardrails → AC
- Each phase feeds the next
- Structure prevents drift

⚛️ Atomic Design

- Organisms → Molecules → Atoms
- Epics → Stories → Tickets
- Hierarchy maintains clarity

What You Built

🗺️ FIFA Navigator App

- Next.js + TypeScript
- Google Maps with traffic
- Real-time MARTA transit
- Multi-language support
- Production-ready architecture

🛠️ Reusable Skills

- 8 prompt frameworks
- Framework selection logic
- Environment-aware development
- AI-assisted structured building

Your Starter Kit

The Repository You Built With

🔗 https://github.com/doctor-ew/prd-to-prod
Branch: demo

What's Inside

📚 Documentation

- `docs/PromptTemplates/`
  - 00-MASTER-GUIDE.md
  - 00-META-FRAMEWORK-GUIDE.md
  - 01-QUICK-REFERENCE-CARD.md
  - 02-CLASS-SESSION-WORKFLOW.md
  - 03-PHASE-3-Milestone-Prompts.md
  - 04-INSTRUCTOR-QUICK-START.md

🎨 Visual Assets

- `prompt_framework.gif`
- Atomic design diagrams
- Example milestones
- Pre-built templates

Next Steps

Where to Go From Here

🏃 Immediate Actions (Today)

1. **Deploy Your App**

```
cd fifa-traffic-demo
vercel --prod
```

2. **Document Your Work**
   - Create `docs/Milestones/MS-XX-Retrospective.md`
   - Answer the reflection questions
   - Screenshot your working app

3. **Share Your Success**
   - Tweet your deployed app

"Vibes build prototypes.
Specs build products."

"Structure enables speed.
Constraints enable creativity."

"AI is not a replacement for thinking.
It's an amplifier for intention."

"Think in atoms, build in molecules,
ship in organisms."