Paper Peloton: Development Blueprint

1. Technical Vision & Architecture

Paper Peloton is a stylized, browser-based cycling simulator. It combines minimalist vector art with high-fidelity performance data from indoor smart trainers. A TDD approach allows for hardware simulation and robust testing. The connection logic should be inspired by the open-source Auuki project's implementation of Web Bluetooth for fitness machines.

2. Tech Stack & Target Hardware

- Language: TypeScript
- Engine: Phaser 3
- Build Tool: Vite
- Testing Framework: Vitest
- Hardware Protocol: Bluetooth FTMS (0x1826)
- Primary Testing Device: Saris H3 Smart Trainer

3. SDLC Phases

Phase 1: Connectivity & Data Validation
- Goal: Dashboard displaying real or mocked wattage.
- TDD: Feed binary buffers to the parser and assert correct Wattage/Cadence.
- Hardware: Validate specifically against Saris H3 FTMS data structures.
- UI: Minimalist HUD with 'Connect' and 'Mock' buttons.

Phase 2: Movement & The Vector World
- Goal: Parallax scrolling linked to physical/mocked power.
- Physics: Calculate velocity using power-to-weight algorithms.
- Visuals: Implement the paper-style 3-layer parallax backgrounds.

Phase 3: The Macro HUD & Elevation
- Goal: Dynamic resistance and course profiling.
- Feature: Real-time scrolling elevation graph.
- Logic: Bi-directional Bluetooth communication for resistance control (0x2AD9).

4. The Master Prompt for Claude

Paper Peloton: Development Blueprint

Copy the following into Claude to begin Phase 1:

'I am building Paper Peloton, a 2D cycling simulator. We are starting Phase 1 using Phaser 3, TypeScript, and Vitest. Reference the Auuki GitHub project for how to handle Web Bluetooth FTMS connections. The primary testing hardware is a Saris H3 trainer.

Tasks:
1. Create a TrainerService.ts that parses FTMS (0x2AD2) bytes.
2. Write a Vitest test file that mocks a 10-byte buffer (8th/9th bytes = 250W).
3. Build a Phaser scene with a digital power display and a button to toggle a Mock Mode for testing. Ensure the Bluetooth logic is modular and easy to swap between mock and real data.'