Let’s break down exactly **how to proceed efficiently** 👇

**🚀 Phase 1: Backend + Core Cognitive Logic (Start here)**

**1️⃣ Set up project base**

* **Tech Stack:** Node.js + Express + MongoDB
* **Structure:**
* /cogniflow-backend
* ├── models/
* │ ├── User.js
* │ ├── Session.js
* │ └── CognitiveData.js
* ├── routes/
* │ ├── sessionRoutes.js
* │ ├── cognitiveRoutes.js
* │ └── gptRoutes.js
* ├── utils/
* │ ├── cognitiveScore.js
* │ └── fatigueEstimator.js
* ├── server.js
* └── config/
* └── db.js

**2️⃣ Implement core data models**

**Session model:**

{

userId: ObjectId,

startTime: Date,

endTime: Date,

tabsUsed: [String],

idleTime: Number,

tabSwitches: Number,

typingSpeed: Number,

cognitiveScore: Number,

fatigueLevel: Number,

momentumScore: Number

}

**3️⃣ Write algorithms for the 3 main features**

**🧩 Cognitive Fatigue Estimation**

File: fatigueEstimator.js  
Logic:

* Track idleTime %, tabSwitchRate, typingSpeed drop.
* If fatigueLevel > threshold → suggest a 5-min break or breathing animation.

Simple example:

export function estimateFatigue(idleTime, tabSwitches, typingSpeed) {

const fatigueScore = (idleTime \* 0.5) + (tabSwitches \* 0.3) - (typingSpeed \* 0.2);

return Math.min(Math.max(fatigueScore, 0), 100); // 0–100 scale

}

**💭 Cognitive Momentum Score**

Track uninterrupted “deep work” blocks (e.g., 25 mins focused = +1 streak).

export function calculateMomentum(focusIntervals) {

const deepWorkStreaks = focusIntervals.filter(interval => interval >= 25 \* 60 \* 1000);

return deepWorkStreaks.length;

}

**😌 Emotion-Aware Feedback**

Later, integrate **face-api.js** on the frontend to detect emotion → send to backend.  
Backend calls GPT with emotion context:

POST /gpt/feedback

{

"emotion": "stressed",

"fatigue": 72,

"momentum": 3

}

GPT prompt example:

“User seems stressed with moderate fatigue and 3 deep-work streaks. Generate a short 2-line motivational message in a calming tone.”

**🧱 Phase 2: Minimal Frontend (after core backend works)**

Build a **simple dashboard** (React + Tailwind) just to:

* Show fatigue meter (progress bar)
* Show momentum streak count
* Show GPT feedback message

Once this pipeline works end-to-end (data tracking → logic → feedback → display), you can later beautify or add GSAP animations.

**🗓 Suggested Plan (Solo)**

| **Week** | **Focus** |
| --- | --- |
| 1 | Setup backend + DB + models |
| 2 | Implement fatigue + momentum logic + APIs |
| 3 | Integrate GPT + test responses |
| 4 | Build minimal UI (React) and connect APIs |
| 5 | Integrate face-api.js + emotion detection |
| 6 | Polish UI, add visual fatigue meter and streak graphs |