**Step 1: Frontend Core (Phase 1)**

**Goal: Track user activity, idle, thinking/distraction, and deep thinking timer.**

**Flow:**

**User Activity (mousemove, keydown, scroll, click)**

**|**

**v**

**Update lastActive**

**|**

**v**

**Check idle every few sec**

**|**

**Idle > threshold? ----> Show Thinking/Distraction Prompt**

**| |**

**| User clicks:**

**| Thinking -> Start Thinking Timer**

**| Distraction -> Log Distraction**

**| No response -> Auto Mark Distraction**

**v**

**Update UI (Floating Timer)**

**Key Components to Build:**

1. **FocusTracker → activity + idle detection**
2. **ThinkingToast → prompt after idle**
3. **TimerDisplay → floating timer for deep thinking**
4. **Dashboard Placeholder → display metrics (later)**

**Frontend Only:**

* **Everything works in local state**
* **Log events in console first → later send to backend**

**Step 2: Backend Integration (Phase 2)**

**Goal: Persist events in database, allow analytics and dashboard.**

**Flow:**

**Frontend Event (Thinking, Distraction, Active)**

**|**

**v**

**Send to Backend API**

**|**

**v**

**Store in MongoDB**

**|**

**v**

**Retrieve for Dashboard**

**Backend Tasks:**

1. **Node.js + Express API endpoints:**
   * **POST /focus-events → save events**
   * **GET /focus-events → fetch events for dashboard**
2. **MongoDB Collection focus-events:**
3. **userId, eventType, duration, timestamp, autoDetected**
4. **Connect frontend → send real-time logs to backend**

**Step 3: Cognitive Metrics & Dashboard (Phase 3)**

**Goal: Give user insights & visual feedback.**

**Metrics:**

* **Deep work streaks (20–25 min uninterrupted)**
* **Focus score calculation:**
* **focusScore = 100 - (distractionTime\*factor) + (thinkingTime\*factor)**
* **Timeline visualization:**
  + **Green = thinking**
  + **Blue = active**
  + **Red = distraction**

**Dashboard Features:**

* **Show daily/weekly trends**
* **Show deep work streaks**
* **Optional: mini charts for quick glance**

**Step 4: Advanced Features (Phase 4)**

**Goal: Make Cogniflow smarter and more interactive.**

1. **Multi-Tab Sync**
   * **BroadcastChannel API → only track one active tab**
2. **Emotion-Aware Feedback**
   * **face-api.js + webcam → detect stress/frustration**
   * **GPT → modify tone: encouraging / strict**
3. **Predictive Suggestions**
   * **Weekly summary: “You spent 40% of time in deep thinking. Try to reduce distractions by 10% next week.”**
4. **Chrome Extension**
   * **Track external distractions → integrate into focus score**

**Step 5: Testing & Iteration**

1. **Test frontend standalone → idle detection, toast, timer**
2. **Integrate backend → test logging and retrieval**
3. **Test edge cases:**
   * **Multiple tabs**
   * **No user response**
   * **Stopping thinking timer mid-session**
4. **Refine UI → animations, Tailwind styles, usability**

**Key Mindset**

* **Frontend first → understand every event, timer, prompt, and logic**
* **Backend later → support persistence & analytics**
* **Advanced features last → multi-tab, emotion detection, GPT  
    
    
    
    
    
    
    
  Frontend Phase 1: Core FocusTracker**
* **We want to track:**
* **User activity (keyboard, mouse, scroll)**
* **Idle time (when user stops interacting)**
* **Thinking/Distraction prompt after idle**
* **Thinking timer to measure deep study time**
* **Step 1: Activity Tracking**
* **Goal: Know if user is active or idle.**
* **Conceptual Steps:**
* **Store a timestamp whenever the user interacts:**
* **e.g., lastActive = Date.now()**
* **Listen to browser events:**
* **mousemove, keydown, scroll, click**
* **Keep checking currentTime - lastActive at intervals:**
* **If difference > idleThreshold (e.g., 5 minutes) → user is idle**
* **How to think about it:**
* **Imagine a variable lastActive. Every time user moves mouse or types → update it.**
* **A separate “watcher” checks every few seconds → if too long → trigger prompt.**
* **Your task: Write a function that updates lastActive on events and logs in console whether user is active or idle.**
* **Step 2: Thinking / Distraction Prompt**
* **Goal: Ask user if they are thinking deeply or distracted after idle.**
* **Conceptual Steps:**
* **After idleThreshold, show a prompt (toast, modal, div) with two options:**
* **Thinking → start thinking timer**
* **Distracted → log distraction event**
* **If user doesn’t respond in 30 seconds → automatically log as distracted**
* **How to think about it:**
* **You are reacting to a condition (idle > threshold).**
* **Create a component or popup that appears conditionally.**
* **Buttons trigger different actions, timeout triggers auto-action.**
* **Your task: Build a simple div or modal that shows after idle and logs user choice in console.**
* **Step 3: Thinking Timer**
* **Goal: Measure how long the user is in deep thinking mode.**
* **Conceptual Steps:**
* **When user clicks Thinking:**
* **Store thinkingStart = Date.now()**
* **Show floating timer on screen**
* **When user clicks Stop Thinking:**
* **Calculate duration = Date.now() - thinkingStart**
* **Log it → later use for metrics**
* **How to think about it:**
* **Timer = difference between current time and start time**
* **Display timer by updating state or variable every second**
* **Your task: Implement a floating div with timer updating every second, stops on button click.**
* **Step 4: Combine Logic in a Flow**
* **User Flow Concept:**
* **User interacts → update lastActive**
* **|**
* **v**
* **Check idle every few sec**
* **|**
* **Idle > threshold? ----> Show Thinking/Distraction Prompt**
* **| |**
* **| User clicks:**
* **| Thinking -> Start Thinking Timer**
* **| Distraction -> Log Distraction**
* **| No response -> Auto mark distraction**
* **v**
* **Floating Timer shows while thinking**
* **Your task: Draw this flow on paper and map what state/variable is needed for each step.**
* **Step 5: Experiment and Learn**
* **Try doing one step at a time:**
* **Step 1: Track activity → console log idle vs active**
* **Step 2: Show prompt after idle → log choice**
* **Step 3: Implement timer → log deep thinking duration**
* **Important: Don’t move to Step 2 until Step 1 works fully — build incrementally.**