Guidelight MVP Sprint Plan

A detailed playbook for building the first working version of Guidelight in one focused day — or across several smaller focused sessions

> **Purpose of this document**
> This is a long-form, detailed plan for how you (Justin) can spend a focused day building the **Guidelight** MVP using **Cursor**, **GPT-5.1**, **Claude Sonnet 4.5**, and **Codex IDE**.
> It's meant to be something you can **read tonight** to get mentally ready, and **glance at tomorrow** as a guide while you work.

How to Use This Plan

This playbook was originally written as a one-day sprint, but you can also use it as a sequence of small work sessions.

- When it says "morning", "today", or "end of day", read that as "the next time I sit down to work on Guidelight" or "the end of this work session".
- It's okay to pause between phases and pick up on another day; you still follow the same overall order.
- The self-care notes (water, food, breaks, wrap-up) are meant for any session, not just one big marathon.

Use whatever chunks of it fit the time and energy you have in real life.

---

0. Big Picture

0.1 What "MVP" means for Guidelight

By the end of the sprint, the **Guidelight MVP** should:

- **Boot and load in a browser** on your dev machine (Vite + React + TS).
- Connect to a **dev Supabase/Postgres** instance (read and write).
- Have a functional **Staff View** where a budtender (or you) can:
  - See the list of budtender picks.
  - Create a new pick.
  - Edit an existing pick.
  - Delete a pick or archive it (even if this is basic / rough).
- Have a simple, clean **Customer View** where:
  - Picks are displayed grouped or tagged by budtender.
  - Info is clear and legible, with only the most important details (name, strain/product, 1–2 key attributes).
- Have **docs that match reality** at a high level:
  - `README.md` roughly describes what's actually there.
  - `GUIDELIGHT_SPEC.md` is mostly correct (minor future work is ok).
  - `ARCHITECTURE_OVERVIEW.md` reflects the structure you actually implemented.

Not perfect. Not pretty. But honest, working, and understandable.

---

1. Tooling & Model Strategy (Reference)

This is the **fixed strategy** we are committing to for this sprint.

1.1 Environment & tools

- **IDE:** Cursor on Windows (your main home).
- **Backend:** Supabase/Postgres (dev project).
- **Frontend stack:** Vite + React + TypeScript.
- **Data fetching:** Initially, simple Supabase client functions and/or a thin API layer in `src/lib/api`. React Query can be introduced later if needed, but it's not required for day one.

1.2 Models and their roles

You will **manually choose the model in Cursor**. The agent cannot do that itself.

- **Primary model – GPT-5.1 Thinking (in Cursor)**
  - Use for:
    - Most feature implementation.
    - Data layer wiring.
    - UI components & layout.
    - Small and medium refactors.
  - Think of it as your **default co-dev**.

- **Specialist model – Claude Sonnet 4.5 (in Cursor)**
  - Use **only when you choose to switch the model** for:
    - Large, cross-cutting refactors (e.g. cleaning up multiple views + data layer).
    - Deep, multi-file debugging when things feel tangled.
    - Architecture review / "make this section clearer and simpler" tasks.
  - Think of it as your **surgical refactor/debug expert**.

- **Composer model (Cursor's own)**
  - Use only when you explicitly want **fast scaffolding or experiments**:
    - Rough component layouts you intend to clean up.
    - Quick "what if" variants.
  - Not for final code that you intend to keep unchanged.

- **Cursor Auto mode**
  - **Do not** use Auto for this repo during the sprint.
  - Always explicitly set a model for Guidelight work.

- **Codex IDE**
  - Use near the **end of the day** as a **reviewer**, not as your main implementer.
  - It should:

  - Review diffs / PR.
  - Point out type issues, runtime risks, and complexity.
  - Suggest specific fixes that you then implement with GPT-5.1 in Cursor.

1.3 The dev agent file

- `GUIDELIGHT_DEV_AGENT.md` is your **AI configuration / system prompt** for Guidelight inside Cursor.
- You will:
  - Keep it in the repo, and
  - Paste its contents into a **custom Cursor agent** for this project.
- The file now instructs the model to:
  - Respect project docs.
  - Use MCP tools for schema and API, not guess.
  - Suggest which model (GPT-5.1 vs Claude) would be best for a task, while still answering as the currently selected model.

You can think of `GUIDELIGHT_DEV_AGENT.md` as the **job description and rulebook** for any AI that works on this repo.

---

2. Overall Day Structure

Here's the **high-level flow** of the day:

1. **Morning Setup & Intent** – Get machine/tools ready & set mindset.
2. **AI Reads Docs & Proposes a Plan** – GPT-5.1 internalizes the repo and outlines steps.
3. **App Shell & Routing** – Basic Vite app + Customer/Staff routes.
4. **Data Layer & Supabase Wiring** – Typed API functions, real data.
5. **Staff View MVP** – CRUD or near-CRUD for picks.
6. **Customer View MVP** – Display picks with clean layout.
7. **Refactor Pass with Claude** – One controlled refactor sweep.
8. **Codex Review & Docs Sync** – External review, fix issues, update docs.
9. **Wrap-Up & Notes** – Capture what's done and what's next.

We'll expand each phase below with **recommended prompts, mental notes, and breaks**.

---

3. Phase-by-Phase Plan

Phase 0 – Morning Reset & Priming (15–30 minutes)

Goal: Start clear and grounded, not frantic.

Steps:

1. **Hydrate & food**

- Drink water. Have coffee/tea if you'd like.
- Eat *something* (even if small) to avoid brain fog mid-morning.

2. **Light review away from the computer (optional but helpful)**
   - On your phone or tablet, skim:
     - `GUIDELIGHT_SPEC.md`
     - `ARCHITECTURE_OVERVIEW.md`
   - Just to recall:
     - Who uses Guidelight.
     - What the two main views are supposed to do.
     - The data model at a high level.

3. **Set a simple intention**
   - A sentence in your head, on a sticky note, or in a small text file:
     - "Today I'm building a functional, honest MVP that runs; I'm not chasing perfection."

This frames the day as **completion, not perfectionism**.

Phase 1 – Workspace & Cursor Setup (20–30 minutes)

Goal: Have everything ready in Cursor so you're not configuring tools mid-sprint.

Steps:

1. **Open laptop, start Cursor, open the Guidelight repo**.

2. **Verify documentation is present:**
   - `README.md`
   - `GUIDELIGHT_SPEC.md`
   - `ARCHITECTURE_OVERVIEW.md`
   - `CONTRIBUTING.md`
   - `GUIDELIGHT_DEV_AGENT.md`

3. **Create / configure the Guidelight Dev Agent in Cursor:**
   - In Cursor's agent/settings UI:
     - Add a new custom agent named something like **"Guidelight Dev Agent"**.
     - Copy/paste the contents of `GUIDELIGHT_DEV_AGENT.md` into the agent's system prompt.
     - Set **model** = `gpt-5.1` (or "GPT-5.1 Thinking").

4. **Create a feature branch for the sprint:**
   ```bash
   git checkout -b feature/guidelight-mvp
   ```

5. **Supabase environment sanity check:**
   - Ensure you have at least:
     - `VITE_SUPABASE_URL`
     - `VITE_SUPABASE_ANON_KEY`

- in your `.env.local` or appropriate config file.
- If anything is missing, note a small TODO like:
  - “TODO: fill Supabase env variables before data layer wiring.”

At this point, **you’re on a dedicated branch** and **your AI agent is configured** with the correct system prompt and model.

Phase 2 – Let GPT-5.1 Read & Plan (30–40 minutes)

Goal: Have the AI generate a concrete MVP plan rooted in your actual docs.

In Cursor (with Guidelight Dev Agent + GPT-5.1):

Use Composer/Agent and paste:

```text
You are the Guidelight Dev Agent.

Task:
1. Read and summarize briefly:
   - README.md
   - GUIDELIGHT_SPEC.md
   - ARCHITECTURE_OVERVIEW.md
2. Based on those documents, propose a concrete MVP plan that I can complete today:
   - 5–8 steps total.
   - Each step should be small enough to implement in a single Composer run.
   - For each step, list:
     - Goal.
     - Files/folders you expect to touch.
     - A suggested model (GPT-5.1 vs Claude Sonnet 4.5) using the “Model suggestion” line.
3. Do NOT modify any files yet. This is planning only.
```

What to do with the response:

- Read the summary to confirm the AI understood the project.
- Read the steps and make sure they roughly match what you expect:
  - Shell & routing,
  - Data layer,
  - Staff View,
  - Customer View,
  - Refactor / cleanup,
  - Review / docs.

If it looks mostly right, reply with:

```text
This plan looks good. Please restate the steps in a short, numbered list that I can reference
```

during the day, then stop. After that I will call you step by step.
```

Now you have a **roadmap created by the AI from your real docs**.

If needed, copy the numbered list into a small file like `MVP_PLAN_TODAY.md` for quick reference.

---

### Phase 3 – App Shell & Routing (60–90 minutes)

**Goal:** Get the basic app running with routes and placeholder views.

**Model:** GPT-5.1 (default).

**Prompt example:**

```text
Implement Step 1 of the MVP plan: app shell and routing.

Requirements:
- Vite + React + TypeScript setup (if not already done).
- A main layout and routing structure, with at least:
  - Customer View route.
  - Staff View route.
- Placeholder components for both views (they can show simple text for now).
- Follow the folder and file structure described in ARCHITECTURE_OVERVIEW.md.

Constraints:
- Do not add complex logic yet.
- Keep changes minimal and focused on getting the app to boot and route correctly.

At the end:
- List the files you created or modified.
- Confirm the npm scripts I should run to start the dev server.
```

Then in terminal:

```bash
npm install      # if not already done
npm run dev
```

Open the app in the browser and confirm:

- It loads without errors.
- You can click or route to “Customer” and “Staff” views (even if they’re just placeholders).
```

Commit:

```bash
git add .
git commit -m "feat: scaffold Guidelight app shell and routing"
```

Mini break (5–10 minutes):
Stand up, stretch, maybe grab water. No doomscrolling.

Phase 4 – Data Layer & Supabase Wiring (60–90 minutes)

Goal: Create a small, typed data layer that talks to Supabase and matches the schema.

Model: GPT-5.1.

Prompt example:

```text
Implement Step 2 of the MVP plan: data layer and Supabase wiring.

Goals:
- Use the Postgres MCP (if available) to inspect the Supabase dev schema.
- Based on GUIDELIGHT_SPEC.md and the actual schema, define TypeScript types for:
  - Budtenders.
  - Picks.
  - Any other absolutely essential entities.
- Implement a small API layer in src/lib/api (or as defined in ARCHITECTURE_OVERVIEW.md) with
functions like:
  - listBudtenders()
  - listPicksForBudtender(budtenderId)
  - createPick(...)
  - updatePick(...)
  - deletePick(...)

Constraints:
- Do not put Supabase client calls directly in React components.
- Keep types aligned with the actual schema. If MCP is unavailable, describe assumptions
clearly.

At the end:
- Summarize the schema you used.
- List the functions you created and their signatures.
- Note any assumptions or TODOs regarding the schema.
```

Then:

- Run `npm run build` or `npm run lint` to catch TypeScript issues.
- Fix any obvious errors with small follow-up prompts like:
  - “Fix the TS errors related to the new data layer in the simplest way that preserves type safety.”

Commit:

```bash
git add .
git commit -m "feat: add typed Supabase data layer for budtenders and picks"
```

Lunch break (30–45 minutes):
Eat something real. Walk a bit if you can. Let your head detach from the code for a bit.

Phase 5 – Staff View MVP (90–120 minutes)

Goal: Staff can view and manage picks through the UI.

Model: GPT-5.1.

Prompt example:

```text
Implement Step 3 of the MVP plan: Staff View MVP.

Requirements from GUIDELIGHT_SPEC.md:
- The Staff View should allow a staff member to:
  - See the list of budtender picks.
  - Create a new pick.
  - Edit an existing pick.
  - Delete or archive a pick.
- Use the data layer functions from src/lib/api; do not call Supabase directly from components.
- Handle loading, error, and empty states in a simple, honest way.
- Keep the layout clean and readable for backroom use; no need for fancy styling yet.

At the end:
- Describe the flow: what the staff member sees and how they interact.
- List which files were changed or created.
- Include a model suggestion line as configured in GUIDELIGHT_DEV_AGENT.md.
```

Then:

- Run the app and manually test:
  - Can you see picks?
  - Can you create/edit/delete a pick?
  - Does the UI show something sensible when there are no picks?

  - Do errors show up in a non-confusing way?

Fix bugs as you find them—this may take a couple of small, targeted prompts.

Commit:

```bash
git add .
git commit -m "feat: implement Staff View MVP for managing picks"
```

Short break (10–15 minutes):
Stretch, maybe step away from the screen. Check in with how you're feeling. You're past the halfway point once Staff View works.

Phase 6 – Customer View MVP (60–90 minutes)

Goal: Customer View displays picks in a clean, legible way.

Model: GPT-5.1.

Prompt example:

```text
Implement Step 4 of the MVP plan: Customer View MVP.

Requirements:
- Display budtender picks in a way that is useful at the POS.
- For each pick, show:
  - Budtender identity (name or label).
  - Product/strain name.
  - 1–2 high-signal attributes (category, simple effect tags, or similar) as described in GUIDELIGHT_SPEC.md.
- Use the existing data layer; do not introduce new direct Supabase calls.
- Layout should be:
  - Readable from a short distance.
  - Not overloaded with text.
  - Grouped or organized in a way that makes sense for a budtender talking to a customer.

At the end:
- Describe how a budtender would actually use this view with a customer.
- Note any UX issues or future improvements you see but did not implement.
```

Then:

- Run the app and walk through it like:
  - A budtender at the counter, quickly referencing picks.

- A customer glancing at the screen.

Ask: “Is this **actually** helping the conversation?”
If something is obviously missing or confusing, fix the smallest possible thing that makes it better.

Commit:

```bash
git add .
git commit -m "feat: implement Customer View MVP for displaying budtender picks"
```

Phase 7 – Refactor & Cleanup with Claude (60–90 minutes)

Goal: Use Claude’s strengths once, in a controlled way, to improve clarity without chaos.

Model: Switch to **Claude Sonnet 4.5** in Cursor for this phase only.

Prompt example:

```text
You are Claude Sonnet 4.5 acting as a refactor specialist for the Guidelight app.

Context:
- Guidelight is a Vite + React + TypeScript app using Supabase for data.
- We now have working Staff and Customer views wired through src/lib/api.
- Please read:
  - GUIDELIGHT_SPEC.md
  - ARCHITECTURE_OVERVIEW.md
  - GUIDELIGHT_DEV_AGENT.md

Task:
1. Inspect:
   - src/views/StaffView*.tsx
   - src/views/CustomerView*.tsx
   - src/lib/api/*
2. Propose a SMALL set of improvements that will:
   - Simplify data and prop flow.
   - Improve type safety.
   - Remove obvious duplication.
3. Apply these improvements in a scoped way:
   - Do NOT rewrite the entire app or change the high-level structure.
   - Limit changes to these files and any minimal helper modules they depend on.
4. Summarize:
   - What you changed.
   - Why the code is now clearer/safer.
```

Keep the refactor surgical, not sweeping.
```

**Your job here:**

- Review diffs carefully. If Claude tries to change too much, tell it explicitly:
  - “That’s too large a refactor. Revert X and keep Y, focusing only on simplifying data flow in StaffView for now.”
- Re-run the app and basic checks:
  - `npm run dev`
  - `npm run build` or `npm run lint`

**Commit:**

```bash
git add .
git commit -m "refactor: clarify views and data layer for Guidelight MVP"
```

Phase 8 – Codex Review & Docs Sync (60 minutes)

Goal: Let an external brain (Codex) review your work, then align docs with reality.

8.1 Push branch & open PR

```bash
git push -u origin feature/guidelight-mvp
```

Open a PR on GitHub from `feature/guidelight-mvp` → `main`.

8.2 Codex IDE review

In Codex IDE, give it the PR or diff and say something like:

```text
This is the Guidelight MVP pull request.

Please review the diff with focus on:
- TypeScript correctness and type safety.
- Potential runtime errors (null/undefined, missing fields, unhandled promises).
- Data-fetching edge cases (loading, error, empty states).
- Any obviously over-complicated logic that could be simplified.

Output:
- A numbered list of concrete issues with file + line hints.
- Plain-language suggestions for fixes (no full rewrites).
- Any high-level design concerns that are worth noting but not required to block this MVP.
```
```

```

**8.3 Apply high-value fixes with GPT-5.1 in Cursor**

Back in Cursor (switch model back to GPT-5.1), ask it to apply Codex's best suggestions, one cluster at a time. For example:

```text
Here is Codex's review of the Guidelight MVP PR (paste excerpt).

Please:
1. Fix the concrete issues they've listed related to null/undefined handling in StaffView and CustomerView.
2. Keep changes minimal and focused.
3. Summarize what you changed.

Do not restructure anything beyond what is required for these fixes.
```

Re-run the app and basic checks after changes.

Commit:

```bash
git add .
git commit -m "chore: address Codex review feedback for Guidelight MVP"
```

8.4 Update docs with GPT-5.1

Ask GPT-5.1 in Cursor to bring docs up to date:

```text
Update the documentation to reflect the current Guidelight MVP:

1. Update README.md to:
   - Briefly describe the current MVP features (Staff View and Customer View).
   - Mention that this is an internal tool for State of Mind.
2. Review GUIDELIGHT_SPEC.md and adjust any details that are clearly outdated compared to the implementation.
3. If the architecture differs meaningfully from ARCHITECTURE_OVERVIEW.md, suggest small edits to align it (or add a short "Current Status" note).

Make the smallest, clearest edits that make the docs honest about the MVP.
```

Review the changes, tweak anything that feels off, and commit:

```bash
git add .
```
```

git commit -m "docs: sync Guidelight docs with MVP implementation"
```

---

### Phase 9 – Wrap-Up & Decompression (15–30 minutes)

**Goal:** End the day with clarity instead of mental static.

1. **Create or update `NEXT_STEPS.md` (or similar)**
   - Bullet out:
     - Small feature gaps (e.g., better filters, sorting, metadata).
     - Known tech debt.
     - Any future ideas you don't want to lose.

2. **Short personal note (optional but helpful)**
   - One paragraph somewhere (Obsidian, Notion, a text file):
     - What you actually got done.
     - What you're proud of.
     - What felt hard or confusing.

3. **Step away**
   - Close Cursor.
   - Do something off-screen for a while (music, show, walk, etc.).
   - Let tomorrow's work wait.

---

## 4. Quick-Reference Prompts

You can copy these snippets into a small cheat sheet file if you want.

### 4.1 Plan the day (GPT-5.1)

```text
Read README.md, GUIDELIGHT_SPEC.md, and ARCHITECTURE_OVERVIEW.md.

Then:
- Summarize the project in a few sentences.
- Propose a 5–8 step MVP plan for today with goals and files per step.
- Add a "Model suggestion" line for each step based on our rules.

Do not edit files yet.
```

4.2 Implement a step (GPT-5.1)

```text
Implement Step X of today's MVP plan.
```
```

- Focus only on this step.
- Minimize unrelated changes.
- Use existing patterns and the data layer.
- At the end, list files changed and summarize behavior.

Model suggestion: include it, but proceed with GPT-5.1.
```

### 4.3 Claude refactor

```text
You are Claude Sonnet 4.5 acting as a refactor specialist.

Inspect:
- src/views/StaffView*.tsx
- src/views/CustomerView*.tsx
- src/lib/api/*

Propose and apply a small set of improvements to simplify data/prop flow and improve type safety without changing overall behavior or structure. Keep changes surgical.
```

4.4 Codex review

```text
Review this Guidelight MVP pull request.

Focus on:
- TS correctness and type safety.
- Null/undefined and runtime risks.
- Data-fetch edge cases.
- Overly complex logic.

Return a numbered list of issues with file/line hints and plain-language fix suggestions.
```

---

5. Closing Note

This plan is not a rigid contract; it’s a **scaffold**. If tomorrow goes differently (longer data-layer work, faster UI work, etc.), that’s fine. The important part is:

- You start with a clear **stack and model strategy**.
- You let the AI do the heavy lifting where it’s strongest.
- You stay in control of architecture, UX, and quality.
- You end the day with a **real, running Guidelight MVP** and a sense of where to go next.
```

Use this document however feels natural: read it through once tonight, then tomorrow just dip into the relevant sections as you move through the day.