SecondBrain – v0.1 MVP Spec (4-week Test Sprint)

Version: Draft v0.1
Founders:

- Nour – product/clinical
- Saahil – tech/implementation

Timeframe: 4 weeks (part-time for both)

---

1. Goal of v0.1

Build a **very small, usable first version** that:

- A resident / busy physician can actually use during real shifts
- Helps them **organize patients and tasks into a single mental queue**
- Gives us **real usage data and feedback** from at least 3–5 doctors

If, at the end of 4 weeks, 3–5 doctors say:

“This is actually useful, I’d like to keep using it if you improve it,”
then v0.1 is a success.

---

2. Constraints (Reality Check)

- Both of us are **part-time** for the next 1–2 months (IELTS, current project, etc.)
- No hospital integrations in v0.1
- No heavy AI/LLM features yet
- Must be **simple web app**, mobile-friendly (most doctors will use it on phone)
- Focus on **one core workflow**, not many features

---

3. Target User Persona for v0.1

“Internal medicine resident in a busy hospital”

- 10–30 inpatients under their care
- Constantly adding mental tasks:
  - “Check K+ in 4 hours”
  - “Follow up CT report”
  - “Call cardiology back”
  - “Review antibiotics in the evening”
- Juggles:
  - Rounds

    - New admissions
    - Calls / messages
    - Documentation
- Biggest feeling:

  > "I'm forgetting something important…"

We design v0.1 for this person, not for everyone.

---

4. Core Use Case for v0.1

Scenario:

- Resident starts / is in the middle of their shift
- Wants to see, in one place:
    - All their patients
    - All open tasks
    - What they should focus on in the next 20–60 minutes

v0.1 must answer one question well:

"Given my patients and tasks, what should I work on now?"

No AI magic is required yet – smart sorting + a clean UI already beats their current chaos.

---

5. v0.1 Scope – Features

5.1. Entities

We keep the data model very simple:

- **User**
    - id
    - name
    - email (optional in v0.1)
- **Patient**
    - id
    - user_id (owner doctor)
    - name / nickname
    - bed / room (optional free text)
    - main problem / diagnosis (free text)
    - priority flag: `normal` / `important` / `critical`
    - status: `active` / `discharged`
- **Task**
    - id
    - patient_id

    - title (short text: “Recheck K+”, “Call cardio”, …)
    - notes (optional longer text)
    - due time (datetime or simple “today / tomorrow”)
    - priority: `low` / `medium` / `high`
    - status: `open` / `done`
    - created_at, completed_at

No more than this for v0.1.

5.2. Screens / Flows

Screen 1 – Today Queue (Home)

This is the **main screen**.

- Shows **list of open tasks across all active patients**
- Each row:
    - Patient name (with small badge if `critical`)
    - Task title
    - Due time (or “Overdue / Today / Tomorrow”)
    - Priority indicator
    - Checkbox / button to mark as “Done”

Sorting logic v0.1:

1. Overdue tasks first
2. Then by due time (earlier first)
3. Then by priority (high > medium > low)

Basic filters (optional but nice if easy):

- Filter by patient
- Filter by priority
- Toggle: “Only critical patients”

From this screen user can:

- Tap task → go to patient detail
- Mark task as done
- Add new task (with patient selection)

Screen 2 – Patients List

Shows all **active patients** of that user.

- For each patient:

    - Name
    - Bed / room
    - Main problem
    - Priority badge (normal / important / critical)
    - Count of open tasks (e.g. “3 open tasks”)

User can:

- Add new patient
- Tap patient → go to Patient Detail

Screen 3 – Patient Detail

For each patient:

- Header:
    - Name
    - Bed / room
    - Main problem
    - Priority flag (user can change)
- Section: **Open Tasks**
    - List of open tasks for this patient (title, due time, priority, mark done)
- Section: **Completed Tasks (collapsed by default)**
- Section: **“What changed” (manual v0.1)**
    - For now, just:
        - A free text area: “Notes since last review”
        - Or a simple list of recent edits (v0.1 can be very basic)

User can:

- Add new task for this patient
- Edit patient info
- Change patient priority
- Mark patient as “discharged” (moves out of active list, but not deleted)

Screen 4 – (Very) Simple Onboarding / Access

v0.1 must be **easy to give to test doctors**, so:

Option A (simplest for test):

- No full auth yet
- One shared “test link” per doctor (with a unique URL token)
- Each token = one “User”

Option B (if not hard):

- Simple email + password registration
- Or magic-link login

We can decide based on your tech preference, but key is:

- **Speed to onboard test users**
- Not security perfection in v0.1

5.3. Basic Analytics / Tracking (for us only)

We need minimal tracking to learn:

- How many times a user logged in
- How many patients created
- How many tasks created / completed
- When they last used it

So v0.1 should at least store:

- last_login_at for each user
- created_at / completed_at for tasks

If easy, we can later show simple stats in an internal admin view or just query the DB.

---

6. Non-goals (Out of Scope for v0.1)

To keep focus, we explicitly **exclude** for now:

- EMR/HIS integrations
- Automatic data ingestion (labs, vitals, notes)
- AI/LLM-based "what changed?"
- State-aware modes (pre-round / bedside / handoff)
- Notifications / reminders (push / SMS / email) – unless you think a very simple version is trivial
- Team collaboration features between multiple doctors

v0.1 = **single doctor, single device, manual data entry**, but with a **much smarter queue** than their current mental one.

---

7. Tech & Implementation – High-level (up to you)

This part is mostly your call as tech lead, but my preferences:

- Web app, mobile-first design

- Any modern stack you’re comfortable moving fast with (React / Next.js / etc. for frontend, Node / Python / etc. for backend)
- Simple deployment (e.g. Vercel / Render / Railway)
- Shared task board in Notion / Trello for sprint tracking

I don’t care about stack purity in v0.1 – I care about **speed, stability, and ability to iterate.**

---

8. 4-week Sprint Breakdown (Proposal)

You can adjust based on your sense of effort, but here’s a first pass:

Week 1 – Skeleton & Data Model

- Set up repo + basic project structure
- Implement:
  - Data model for User / Patient / Task
  - Very simple storage (DB or even hosted DB like Supabase)
- Rough UI for:
  - Patients list (no fancy design)
  - Patient detail
- No full auth yet or very basic auth
- Internal-only version for us to click through

Output:
Clickable prototype with static-ish data so we can validate flow.

Week 2 – Today Queue + CRUD

- Implement **Today Queue** screen with sorting logic
- Full CRUD:
  - Create/Edit/Delete patient
  - Create/Edit/Complete task
- Connect queue ↔ patient detail
- Make UI at least **clean and usable on mobile** (no need for perfect design)

Output:
First end-to-end version where a test doctor could:

- Add patients
- Add tasks
- See them in the queue
- Use it for a few minutes continuously

Week 3 – Polishing & Test Users

- Basic polishing:
    - Better layout on mobile
    - Small improvements in UX (shortcuts, default values)
- Add manual “What changed” notes on patient detail
- Add minimal tracking (last_login, tasks created/completed)

On my side (Nour):

- Bring in **2–3 real doctors** to test the app in real life
- Collect structured feedback (what’s confusing, what’s missing, what’s useful)

Week 4 – Iterate on Real Feedback

Based on feedback from first users:

- Fix obvious bugs / friction points
- Add **one or two** high-impact improvements, for example:
    - Quick “add task” shortcut from queue
    - Simple filter (e.g. “show only critical patients”)
    - Better default due times (“today evening”, “tomorrow morning”)

By end of week 4:

- We should have:
    - At least 3–5 doctors who have used it multiple times
    - Concrete quotes + metrics
    - A clear sense if this is worth going deeper

---

9. Success Criteria for the 4-week Test

We consider the 4-week sprint successful if:

- Product:
    - v0.1 is live, stable enough to use in real shifts
- Users:
    - 3–5 doctors have used it for at least **3 days** each
    - At least 2 of them say they’d be unhappy if we took it away
- Team:
    - We both feel that:
        - We can work well together
        - We move at a reasonable speed given our constraints
        - It’s worth discussing deeper commitment / equity / next steps

If not, we still win because:

- We’ll have tested the idea in reality

- We’ll have code, structure, and learnings for the next iteration
- We’ll know if we should pivot, change team, or change approach

---