

Vita AI - Smart Task Manager Feature V1

▼ Task Instructions - Candidate Brief

This task is a **fictional feature** created only for assessment. It does **not** reflect what you will be working on in your role. The purpose is to test the skills and problem-solving you'll need day-to-day, in a format that mirrors our real feature specifications. Hence the big document as a representative example.

Scope

For this task, focus **only** on:

- The **prioritisation algorithm** (scoring engine).
- **Substitution logic** (how tasks change after repeated dismissals).
- **Backend API endpoints** to expose the results.

All other sections are included for context only. Anything shown with strikethrough is explicitly **out of scope**. we've done our best to strike through anything that is out of scope but if there's anything that doesn't line up EG any parts related to syncing with external hardware like Apple watches then that is out of scope (just saying in case there is a tiny bit that we haven't strike through)

Imperfections in the Spec

This document is **not perfect**. It may contain gaps, errors, or unrealistic details (e.g. in data fields or the ERD).

- You are free to **adapt, rename, or extend fields** if it helps make the system coherent.
 - You may add your own assumptions, metrics, or sample data as needed.
 - In the *real role* you would never do this silently — you'd raise the issue, and we'd resolve it together. For this test, we want to see how you handle it independently.
-

Expectations

- **Timebox**: 4–6 hours total.
 - **Language/stack**: JavaScript/TypeScript with Node.js + Express (backend) and React (frontend).
 - **Deliverable**: Working code demonstrating the engine logic and endpoints. Frontend polish is optional unless applying for frontend/full-stack roles.
 - **Creativity**: Encouraged for this task. Treat it as a sandbox to show your skills, not a production build.
-

Role-specific notes

- **Backend candidates**: Build the backend for the Smart Task Manager only.
- **Frontend candidates**: Build the dashboard UI (Figma link provided), focusing on task cards and polish/animations.
- **Full-stack candidates**: Do both, but keep frontend lightweight except for task cards.

Here is the figma, you will need to make a copy of this on your own account so that you can actually see it all and use the components or do whatever you need to do:

<https://www.figma.com/design/eU6DNC9pEys6v2gR90ZScI/Vita-AI-assistant?m=auto&t=gB9iWEBjmO2OHWuG-1>

Final Notes

- Treat this as a **one-day test project**, not a production build.
- You are free to use the resulting work in your portfolio.

- Focus on exams first — only do this once they're finished!

▼ Scope

This take-home is **backend/full only**. Build a small Node.js/Express (TypeScript preferred) service that implements a **deterministic prioritisation engine** for wellness tasks and exposes a minimal HTTP API. No UI, no cloud, no external integrations.

In scope

- Deterministic scoring engine that returns **exactly 4** recommended tasks at any time.
- Anti-nag behaviour (dismiss → micro-substitution after N ignores).
- Time-of-day gating (soft).
- Daily reset of volatile state (`ignores` , `completedToday`).

Out of scope

- Auth, accounts, wearables/APIs, schedulers/cron, analytics, GDPR/WCAG, localisation, microservices.

▼ Objective

Given **today's user metrics**, compute and return the **top 4** tasks from a predefined catalog. The engine must:

1. score tasks using urgency/impact/effort/time-of-day,
2. avoid immediate repeats after dismiss,
3. substitute micro-tasks after repeated ignores,
4. hide completed tasks for the day.

▼ Acceptance Criteria (Hard Pass/Fail)

1. `/recommendations` returns **exactly 4 unique** tasks in deterministic order for a given state.
2. Scores for **Scenario A** match the provided reference values (≥ 4 dp).
3. **Substitution**: after 3 dismissals of a task with `micro_alt` , the micro task appears in subsequent recommendations.
4. **Completion** hides the task for the rest of the day.
5. **No immediate repeat** in the same response after dismiss.
6. **Time gating** affects scores (Scenario D proves the delta between day vs evening).
7. Rationale strings include **actual metric values** (not placeholders).

▼ Submission

- Git repo with `server` folder.
- Scripts: `dev` , `test` .
- README: setup, design notes, any **extra categories** you added and their urgency functions, and how you enforce daily reset.

▼ Tests (minimum)

- Unit tests for scoring:
 - Hydration/steps urgency ramps behave as specified (boundary tests at goal).
 - `inverseEffort` yields expected values for 3, 5, 10, 15 minutes.
 - Time factor 1 vs 0.2 changes final score accordingly.
- Behavioural tests (can be request-level):
 - Scenario A output order & scores match reference.

- Substitution triggers after 3 dismissals.
- Completion hides task until date change.

▼ Evaluation Rubric (Backend-only, 100 pts)

- **Correctness (40):** Meets Acceptance Criteria; deterministic outputs.
- **Code Quality (25):** Readable, modular, pure functions for scoring; no hidden state.
- **Tests (20):** Covers scoring and behaviours; clear fixtures.
- **Operational Simplicity (10):** Clean scripts, no over-engineering.
- **Documentation (5):** README and inline comments explain key trade-offs.

▼ Task Context

for context let's say that this application is something that ties into a user's a daily hardware such as an Apple Watch and phone device and laptops So the application is installed across everything and that is how it is able to monitor things such as sleep steps screen time etc

the objective of the application as a whole is to be able to gather all the user's data so they're not switching between different health apps and we are able to essentially help the user to make smarter decisions in their life and be a happier healthier person

▼ Feature Document

▼ Feature Description

this feature is built to be able to help a user in their daily life and take away the constant thinking process of what do I do to keep myself healthy Essentially giving them 4 small little cards which they can look at and determine what tasks that they would like to do in that moment in order to improve their daily Wellness. The tasks should be created on the basis of metrics that the user being monitored for and on other metrics such as user drinking water which would require user input. the key in this smart Task Manager is it being smart. the idea being that it is able to track things like happiness against how much the user is drinking water and other factors and being able to actually correlate that to create a healthier happier user and is prioritising tasks based on the users happiness but also on the level of effort or procrastination that the user may have in doing said task or even the amount of time that they take to do said task. an example is telling a user to lower screen time when let's say they are a computer developer as their job role That doesn't necessarily work however perhaps suggesting an activity such as going and taking their dog for a walk in the evening rather than watching TV would help with lowering screen time so it's not just directly stating lower screen time as a task but instead being smart and offering smart suggestions.

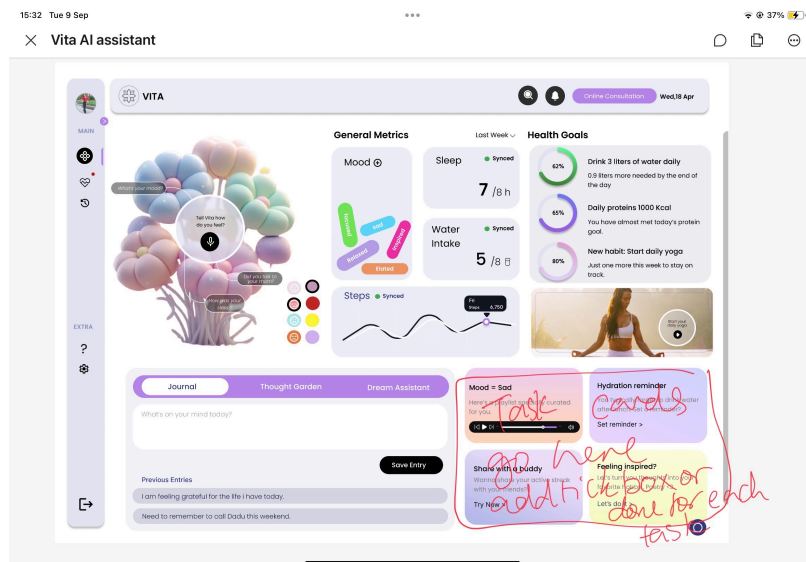
the task would require you to also have space to input things like drinking water however it is also possible depending on how you approach the task to instead have that as something that is preset but is part of a different feature where a different feature may have that in the settings so it is assumed that that is where you would be pulling the data from however you would need to state that assumption if that is what you choose to do. you can also choose to instead do it in a way The data is asked for as a simple input at the beginning. the idea behind the algorithm is that it gets smarter on the basis of the user's daily inputs and monitoring through external hardware.

Goals	Tasks
Drink XL water daily	Task: "Complete today's water intake, your YL of intake remaining"
Sleep Xh daily	Task: "Get Yh of sleep"
Walk X steps daily	Task: "Walk Y steps to reach today's step count goal"
Mood check-ins	Task: "Log your mood today"
Monitor the Screen Time	Task: "Your screen time is Y hrs till now"

▼ User story

- As a user, when I set my sleep goal (X hours), I want the system to track and remind me so I can maintain a healthy rest.
- As a user, when I set my daily water intake goal (X litres), I want the system to monitor and prompt me so I stay hydrated.
- As a user, when I set my step goal (X steps), I want the system to track my activity so I can stay active and healthy.
- As a user, when I want to improve my mood, I want the system to suggest and track positive actions so I can feel happier.
- As a user, when I set a screen time limit, I want the system to monitor and notify me so I can reduce digital fatigue.
- When a new task is added I want my tasks to be reprioritised
- As a user I want only my top 4 tasks to show on the task cards
- As a user I want to be able to complete task cards (this can be via animation or a simple checkbox)

▼ UX



▼ Example Flow

- 1. Morning:** Cards show:
 - "Drink 0.5L of water before noon."
 - "Walk 1,000 steps before lunch."
 - "Log your morning mood."
 - "Set bedtime target tonight."
- 2. Afternoon (user low mood, low water):**
 - "Take a 10-min walk outdoors."
 - "Drink 0.5L water now."
 - "Log your mood check-in."
 - "Stretch away from your desk for 5 mins."
- 3. Evening (user hits steps but high screen time):**
 - "Take your dog for a 20-min walk before TV."
 - "Drink 0.5L water — 1L remaining."

- "Wind down early: aim for 7h sleep."
- "Write a short reflection in your journal."

⚠ **Critical pushback:**

This only works if you're ruthless about **avoiding "nagware."** If users feel pestered by repetitive reminders ("drink water, drink water"), they'll abandon the feature. The intelligence has to feel **personalised, adaptive, and relevant**. Otherwise, you've just rebuilt a habit tracker with prettier cards.

UI/UX Requirements

Task Card

- Max 4 visible at once.
- Contains:
 - Title (e.g., "Drink 0.5L water now").
 - Progress indicator (e.g., "1.5L left today").
 - Estimated time/effort (icon or text).
 - Completion control (checkbox / swipe / animation).

Behaviours

- On completion:
 - Smooth animation (card flips/vanishes).
 - Confetti or progress bar update.
- On dismissal:
 - Task marked "ignored," replacement pulled in.
- On empty state (all done):
 - Show motivational card ("You crushed today's wellness goals!").

▼ Detailed Behaviours

▼ 1. Task Card Generation

- System generates up to **4 task cards** at a time.
- Cards are **context-aware**: they're not generic reminders (e.g., "lower screen time") but actionable alternatives (e.g., "take your dog for a 15-min walk").
- Cards refresh dynamically when:
 - User completes a task.
 - A new priority emerges (e.g., hydration is low, or mood dips).
 - A scheduled check-in occurs (e.g., every 2–3 hours).

Behaviour: When a user drinks less than 30% of their water goal by 4pm, the system **automatically surfaces a hydration card** instead of, say, a step card.

▼ 2. Task Prioritisation Engine

Tasks are ordered by **smart weighting**, combining:

- **Urgency** (how close the user is to missing a daily target).
- **Impact** (historical data showing which tasks improve the user's happiness/energy).
- **Effort/procrastination score** (quick wins vs. big commitments).

- **Contextual fit** (avoiding nonsense tasks, e.g., "reduce screen time" when user's job is computer-based).

Behaviour: If a user logs low mood and has spare time in the evening, the system surfaces an **outdoor activity task** instead of a hydration reminder.

▼ 3. Task Completion

- Each card has a **completion action**: checkbox, swipe away, or satisfying animation (confetti, progress bar filling).
- Completing a task updates **progress metrics** (water %, steps, mood logged).
- Completing a task can **trigger reprioritisation** of remaining cards.

Behaviour: After ticking off "Drink 0.5L water," the hydration card updates to:

"Only 1.5L remaining today — you're 50% done."

▼ 4. Data Inputs

Two approaches (you'll need to choose):

1. **Settings-driven input:** user sets daily targets (water, steps, sleep, screen time) in app settings. ~~The system pulls data automatically from wearables/APIs (Fitbit, iOS Health, Google Fit).~~
2. **User-input driven:** app prompts for manual entries (e.g., "How many glasses of water have you had so far today?").

Assumption to state clearly: If external hardware is connected, water/sleep/steps can sync automatically. Otherwise, fallback = manual inputs.

▼ 5. Learning/Adaptation

- System **correlates tasks completed vs. mood check-ins** over time.
- Learns which activities boost mood or reduce fatigue (e.g., walks improve mood more than stretching).
- Uses that learning to **reprioritise future tasks**.

Behaviour: After 2 weeks of data, system notices that user's mood improves most on days they walk ≥5,000 steps. In the afternoon slump, it prioritises a **"Take a 10-min walk" card** over "Drink water."

▼ 6. Context-Aware Adjustments

- **Time-aware:** won't suggest "get 7h sleep" at 3pm, but may suggest "start winding down early tonight."
- **Job-aware:** won't push "reduce screen time" to a developer at 2pm, but might suggest "step away for 10 mins of stretching."
- **Environment-aware** (optional, if data available): e.g., if weather API says it's raining, instead of "walk outside," suggest "do 10 mins of indoor movement."

▼ 7. Task Refresh & Rotation

- New tasks are pulled in when:
 - A goal is almost complete.
 - A task becomes irrelevant (e.g., sleep card disappears after waking).
 - User explicitly dismisses a card.
- Maximum of 4 cards displayed at once.

Behaviour: If a user finishes steps goal at 8pm, the "walk" card disappears and may be replaced with "journal your mood before bed."

▼ 8. Feedback Loop

- Completed tasks feed back into **progress visualisation** (hydration %, steps %, screen hours vs. limit).
 - Missed tasks are tracked to adjust future reminders (if user consistently ignores hydration reminders, tone/format changes).
 - User can give quick **feedback** ("helpful" / "not useful"), teaching the algorithm what resonates.
-

▼ Adaptive behaviour for incomplete tasks

1. Silent Tracking Instead of Asking

- Don't pop up a "Was this helpful?" nag.
- Instead, monitor **which cards the user consistently ignores** or dismisses.

Behaviour: If the user ignores "Journal your mood" three days in a row, the system automatically suppresses that card.

2. Goal Reframing

If a task isn't being done, reframe the same outcome into a **different, less resistant form**.

- **Ignored:** "Journal your mood for 5 minutes."
- **Alternative:** "Rate your mood 1–5 in one tap."
- **Alternative:** "Pick an emoji that matches your mood right now."

→ Goal (mood tracking) is still met, but friction is reduced.

3. Substitute Behaviours

If the system sees that a category of task is consistently skipped, it offers a **different activity with similar wellness benefits**.

- **Skipped:** "Take a 20-min walk."
- **Alternative:** "Do a 3-min stretch at your desk."
- **Alternative:** "Walk around the block once."

→ System still hits movement target, but via smaller/lower-barrier steps.

4. Contextual Substitution

Instead of repeating the same ignored task, system looks at **user context** and offers a *different* way to hit the same metric.

- **Skipped screen time warning:** Instead of repeating "*lower screen time*" → suggest "*call a friend while walking your dog*" (achieves both social connection and movement, indirectly lowering passive screen use).
-

5. Dynamic Rotation

- If a user skips a task **3 times in a row**, it gets pushed down in priority and replaced with a fresh alternative.
 - System doesn't delete the original goal — it just **tries a different angle**.
-

6. Micro-Success Strategy

If user resists larger tasks, system offers **smaller wins** to build momentum.

- **Skipped water intake 2 days:** Instead of “Drink 1.5L today,” system surfaces:
 - “Take 3 sips of water now.”
 - “Refill your water bottle — don’t worry about finishing it.”

Small completions give the **dopamine hit** that drives compliance.

7. Progressive Intelligence

- Over time, the system learns:
 - “This user never journals but responds well to emoji mood check-ins.”
 - “This user never does 20-min workouts but often accepts 5-min ones.”
 - It adapts baseline tasks to what’s **realistic and fulfilling for this individual**.
-

Example Scenario

User repeatedly ignores hydration tasks.

1. **Day 1:** “Drink 0.5L water now.” → Ignored.
2. **Day 2:** “You have 1.5L left to drink.” → Ignored.
3. **Day 3:** Instead of repeating, the card changes to:
 - “Eat a water-rich food (fruit/veg).”
 - “Carry a bottle to your desk now.”

System still promotes hydration without the same direct demand.

⚠ Critical check for you:

This is a **much harder product to build** than a simple tracker, because it requires a **library of alternative behaviours per goal** and a prioritisation engine that understands *why* the user isn’t engaging (effort, time, or interest). Without that, it risks collapsing back into “just another tracker.”

▼ Functional Requirements

- ✓ System displays max 4 tasks at once.
- ✓ Tasks adapt dynamically to user progress.
- ✓ Ignored tasks rotate into micro/alternative options.
- ✓ Completed tasks update metrics and refresh task list.
- ✓ System pulls from APIs if connected, manual input otherwise.
- ✓ Algorithm reprioritises tasks based on urgency, impact, effort, and context.
- ✓ User never sees “nagging repeats” more than 3 times in a row.

▼ Non-Functional Requirements

1. Performance

- Task cards must load **within 1 second** after app open.
 - Re-prioritisation (when a task is completed/dismissed) must resolve and refresh visible cards **in under 500 ms**.
 - ~~Background sync with APIs (Google Fit, Apple Health, etc.) should complete in <5 seconds on normal Wi-Fi/4G.~~
-

~~2. Scalability~~

- System must support **10,000 concurrent users** in MVP without degradation.
 - Architecture must allow scaling to **100,000+ users** by horizontal scaling (stateless services, task-scoring engine can run as microservice).
-

3. Reliability & Availability

- Daily task data must persist with **99.9% reliability**.
 - If API sync fails, system must fall back to cached/manual inputs without crashing.
 - End-of-day reset runs reliably, even if offline — syncs next time connection is available.
-

4. Security

- All health/goal data encrypted **in transit (TLS 1.2+)** and **at rest (AES-256)**.
 - No raw health API tokens stored — use OAuth with refresh tokens where possible.
 - User mood and wellness data must be treated as **sensitive PII** → comply with GDPR.
-

5. Privacy

- Explicit user consent before connecting wearables/APIs.
 - Users can request data export/deletion in compliance with **GDPR/CCPA**.
 - Data anonymisation required for any analytics or model training.
-

6. Usability

- Max 4 task cards visible at once, always.
 - User should be able to **complete or dismiss a task in ≤2 taps/click/swipes**.
 - Progress indicators must be **clear at a glance** (hydration %, steps left, hours sleep remaining).
-

7. Maintainability

- Task scoring/substitution logic must be configurable
 - Code must be **modular**:
 - **task-service** (CRUD tasks, track history)
 - **scoring-service** (calculate priority, substitutions)
 - **sync-service** (integrations with wearables/APIs)
 - Unit test coverage for critical logic (task scoring, substitutions) **≥80%**.
-

8. Extensibility

- New wellness goals (e.g., nutrition, meditation) can be addable without rewriting core engine.
 - Substitution framework must support **multiple alternatives per goal**.
 - New data sources (e.g., smart bottle, sleep tracker) can be integrated via **adapter pattern**.
-

9. Offline Mode

- App must function in offline mode with cached tasks for the day.
- Syncs automatically when back online.


- No data loss allowed when switching offline → online.

10. Localization & Accessibility

- Support for **multiple languages** (text must be externalised).
- All interactions (checkboxes, swipes) must be **screen-reader accessible**.
- Minimum **WCAG 2.1 AA** compliance.

11. Analytics & Logging

- System must log:
 - Task completion rate
 - Task ignore/dismiss rate
 - Correlation between tasks & mood check-ins
- Logs anonymised by default; identifiable only with explicit opt-in.

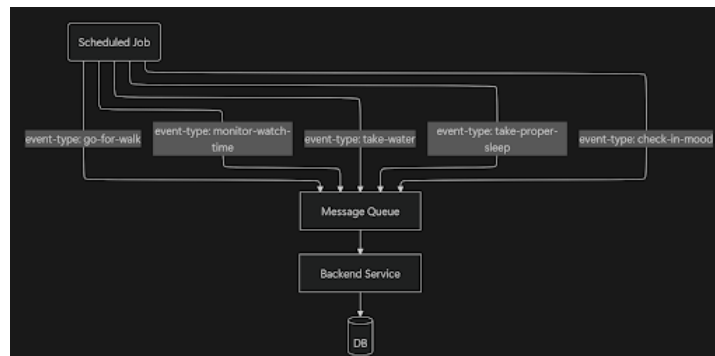
 **Acceptance Criterion for NFRs:** System must meet all above in staging load tests, code reviews, and security scans before production release.

▼ Data Requirements

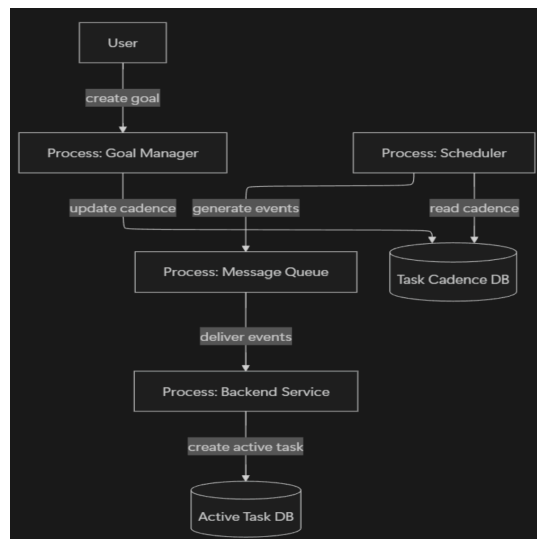
- **User**
 - `id`
 - `profile` (job role, preferences, connected devices)
 - `goals` (hydration L/day, sleep hrs/day, steps/day, screen time limit)
- **Metrics (daily logs)**
 - `date`
 - `water_intake (L)`
 - `steps (count)`
 - `sleep (hours)`
 - `screen_time (hours)`
 - `mood (score 1-5 / emoji)`
- **Task**
 - `id`
 - `goal_type` (hydration, sleep, steps, mood, screen)
 - `title` (user-facing text)
 - `effort_level (low, medium, high)`
 - `impact_weight` (numeric, effect on wellbeing)
 - `duration_estimate` (minutes)
 - `status (pending, completed, dismissed, ignored)`
- **Task History**
 - `user_id`
 - `task_id`
 - `date`
 - `completion_status`

▼ Architecture Diagram Chosen Solution

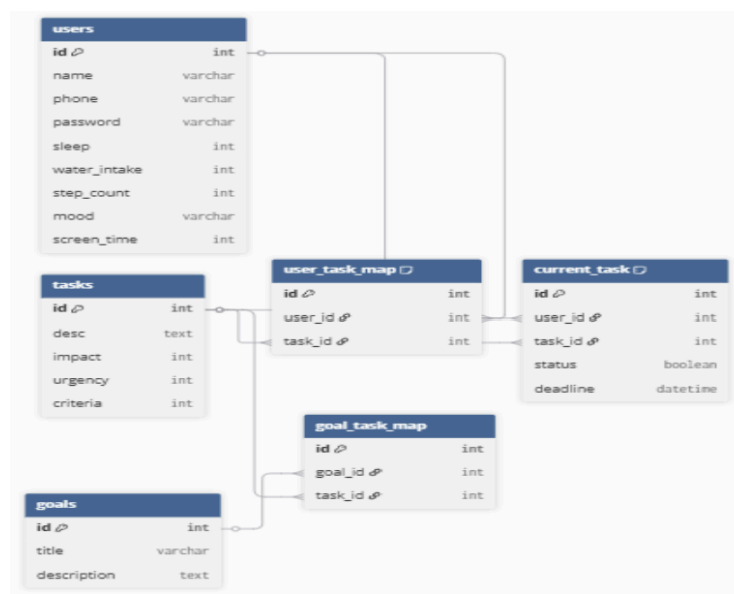
this is just what's provided you can modify as needed



▼ Data Flow Diagram



▼ Entity Relationship Diagram (ERD)



▼ APIs

- **Step count/sleep:** Google Fit, Apple Health, Fitbit APIs. would apply but for this task : manual entry for all metrics as if no devices connected.
- **Water intake:** manual input (later optional smart bottle integration).
- **Screen time:** iOS Screen Time API, Digital Wellbeing (Android).
- **Mood:** manual check-in UI.

▼ Implementation notes

Time-of-Day Windows

- `morning` = 05:00–11:59
- `day` = 12:00–17:59
- `evening` = 18:00–23:59

Tasks without a `time_gate` are always eligible.

Algorithms & Logic

A. Task Selection (max 4 active cards)

1. Collect all tasks linked to goals.
 2. Score tasks based on:
 - **Urgency** (how far from daily goal).
 - **Impact** (historical effect on mood/wellbeing).
 - **Effort** (quick win vs. big ask).
 - **Context fit** (time, job role, device activity).
 3. Select top 4 tasks with **highest weighted scores**.
-

B. Task Substitution (if ignored/dismissed)

- If a task is **ignored 3x in a row**, demote it.
- Replace with a **micro or alternative task** that hits the same underlying goal.

Example substitution rules:

- Hydration:
 - Primary = "Drink 0.5L water."
 - Micro = "Take 3 sips now."
 - Alternative = "Eat a water-rich fruit."
- Movement:
 - Primary = "Walk 2,000 steps."
 - Micro = "Stand and stretch for 2 mins."
 - Alternative = "Take stairs once today."
- Mood:
 - Primary = "Journal 5 minutes."
 - Micro = "Pick an emoji for your mood."
 - Alternative = "Say one thing you're grateful for."

C. Task Completion Flow

1. User taps checkbox/swipes card → mark **completed**.
 2. Update metrics (hydration %, steps %, etc.).
 3. Trigger re-scoring of task list → refresh top 4 cards.
-

D. Learning

- Track **completion vs. mood correlation**.
- If certain tasks improve mood scores, raise their **impact weight**.
- If tasks consistently ignored, lower their priority and show alternatives.

Goal → Task Assignment

- When a **user registers for a goal**, all the tasks linked to that goal (via `goal_task_map`) are automatically added to `user_task_map`.
- This indicates that the user has subscribed to these tasks in a **cadence-based manner**.

Task Scheduling

- A **scheduler job** runs at regular intervals to check for tasks that are available for each user.
- When conditions are met, the scheduler triggers the **task creation workflow**, which inserts the corresponding task(s) into `current_task` with deadlines and initial status.

Task Execution & Display

- The **current active task(s)** for each user are fetched from `current_task` and displayed in the application for the user to work on.
- Tasks are ordered and shown based on **priority scoring (algorithm-driven)** that considers impact, urgency, and criteria.

Task Progress Tracking

- As the **user updates progress metrics**, the system evaluates them against the defined criteria of the task.
- If the criteria are fulfilled, the corresponding task is marked as **completed** (status = true).
- Once **all required metrics** for a task are satisfied, the task is officially **closed**.

Task Prioritization

- The system applies a **prioritization algorithm** (e.g., weighted score using impact, urgency, and criteria) to determine the next best task.
- The prioritized list is then surfaced to the user to ensure focus on the **most important and time-sensitive work**.

▼ Sample prioritisation algo:

Scoring (Deterministic)

Compute each **eligible** task's score using the formula below, then return the **top 4** (see §7 for algorithm & tie-breakers).

```
score(task) =  
W_urgency * urgencyContribution(task, metrics)  
+ W_impact * task.impact_weight
```

```
+ W_effort * inverseEffort(task.effort_min)
+ W_tod * timeOfDayFactor(task.time_gate)
• W_penalty * task.ignores
```

Reference Weights (must be defaults):

```
W_urgency=0.5, W_impact=0.3, W_effort=0.15, W_tod=0.15, W_penalty=0.2
```

Components

- `urgencyContribution(task, metrics)` (clamp 0..1):
 - Hydration: if `water_ml < 2000`, `(2000 - water_ml)/2000` else `0`.
 - Steps: if `steps < 8000`, `(8000 - steps)/8000` else `0`.
 - Sleep: if `sleep_hours < 7` → `1`, else `0`.
 - Screen: if `screen_time_min > 120` → `1`, else `0`.
 - Mood: if `mood_1to5 <= 2` → `1`, else `0.3`.
- `inverseEffort(mins) = 1 / log2(mins + 2)`
(smaller tasks get a slight boost; ensure `mins >= 1`).
- `timeOfDayFactor(gate)` → if within window return `1`, else `0.2`; if no gate, `1`.

You may extend categories, or can change the defaults above. If you add categories or change anything document it, document the urgency function.

Selection Algorithm & Tie-Breakers

- Build **candidate set** = tasks where `completedToday === false`.
- Apply **substitution rule**: if task has `micro_alt` and `ignores >= 3`, use the micro task in place of the parent for this cycle.
- Compute scores (4dp).
- No immediate repeat**: if a task was dismissed in the same request cycle, don't re-insert it into the current response; pick the next best.
- Sort** by: score **desc**, then `impact_weight` **desc**, then `effort_min` **asc**, then `id` **asc**.
- Return the first **4 unique** tasks. If <4 eligible, relax only by removing time gates (treat `timeOfDayFactor=1`) but **never** duplicate IDs.

Seed Catalog & Reference Scenarios

Use the following **seed tasks** (you can load them via `/admin/seed`).

```
[
  { "id": "water-500", "title": "Drink 500 ml water", "category": "hydration", "impact_weight": 4, "effort_min": 5, "micro_alt": "water-250", "ignores": 0, "completedToday": false },
  { "id": "water-250", "title": "Drink 250 ml water", "category": "hydration", "impact_weight": 3, "effort_min": 3, "ignores": 0, "completedToday": false },
  { "id": "steps-1k", "title": "Walk 1,000 steps", "category": "movement", "impact_weight": 4, "effort_min": 10, "micro_alt": "steps-300", "ignores": 0, "completedToday": false },
  { "id": "steps-300", "title": "Walk 300 steps (indoors ok)", "category": "movement", "impact_weight": 3, "effort_min": 5, "ignores": 0, "completedToday": false },
  { "id": "screen-break-10", "title": "Take a 10-min screen break", "category": "screen", "impact_weight": 4, "effort_min": 10, "ignores": 0, "completedToday": false },
  { "id": "sleep-winddown-15", "title": "15-min wind-down routine", "category": "sleep", "impact_weight": 5, "effort_min": 15, "time_gate": "evening", "ignores": 0, "completedToday": false },
  { "id": "mood-check-quick", "title": "Quick mood check-in", "category": "mood", "impact_weight": 2, "effort_min": 3, "ignores": 0, "completedToday": false }
]
```

Scenario A (reference numbers in §3)

Metrics (today): `{ water_ml: 900, steps: 4000, sleep_hours: 6, screen_time_min: 150, mood_1to5: 2 }`

Local time: 15:00 ("day" window).

Expected `/recommendations` (top 4):

1. `screen-break-10` **score: 1.8918**
2. `water-500` **score: 1.6784**
3. `steps-1k` **score: 1.6418**
4. `mood-check-quick` **score: 1.3146**

Scores must match ≥ 4 dp. `sleep-winddown-15` may rank high but is outside its time gate (soft factor 0.2) and is not guaranteed to be in the top 4.

Scenario B (substitution after ignores)

Start from Scenario A. Call `POST /actions/dismiss` for `water-500` **three times** (same day).

Next `/recommendations`: `water-250` **must** appear instead of `water-500`.

Reference score (for the same metrics): `water-250` **1.3896**.

Scenario C (completion)

From Scenario A, `POST /actions/complete` on `screen-break-10`.

Next `/recommendations` must exclude it for the rest of the day and pull the next best task.

Scenario D (time gating softens, not zeros)

Set local time to **15:00** and metrics so sleep is urgent (e.g., `sleep_hours: 5`).

`sleep-winddown-15` is allowed in the list but receives the **0.2 time factor** in scoring. Confirm its score differs when you move time to **20:00**.

Scenario E (no duplicates & relaxation)

If fewer than 4 eligible tasks remain, you may relax time gates (factor=1) to return 4 unique tasks. Never duplicate IDs.

9) Daily Reset Behaviour

At **local midnight**:

- Reset `ignores` to `0` and `completedToday` to `false` for all tasks.
- Keep historical actions in `TaskHistory` (optional).

Implementation may be a simple check on first request after date change; no scheduler required.

Edge Cases

- **No data:** fall back to generic tasks (e.g., "Log your mood today").
- **All tasks ignored:** surface micro-wins (lowest friction actions).
- **Offline:** cache tasks locally, sync on reconnect.
- **End of day:** reset progress, generate fresh tasks for tomorrow.