# **Product Requirements Document (PRD)**

## **1) Product Overview**

Build a lightweight, cross‑platform web app inspired by **TeuxDeux’s horizontal day columns** with an **expandable Notes area per day** and **Franklin‑Covey–style task carry‑forward**. Users can create tasks with due dates and tags, see them in daily columns, and any **incomplete tasks automatically roll to the next day at local midnight**. A global search returns **the specific day(s)** where a task or note appears.

**Working name:** "FlowApp Daily Planner"

**Primary value:** zero‑friction daily planning + automatic continuity for unfinished items.

## **2) Goals & Non‑Goals**

**Goals**

* Make daily planning fast and visual (TeuxDeux‑like horizontal days).
* Preserve momentum by auto‑moving incomplete tasks to the next day at midnight (Franklin‑Covey behavior).
* Support basic organization with **due date**, **tags**, and **search** across tasks and notes.
* Provide simple task filtering with tags
* Provide a clean, responsive UI that works on desktop and mobile.
* Integrate with Google Calendar

**Non‑Goals (v1)**

* Collaborators/multi‑user sharing.
* Complex recurrence rules (beyond daily rollover).
* Notifications/push reminders.

## **3) Personas**

* **Solo Planner (Primary):** Individual tracking personal & work tasks daily.
* **Task Tidy:** Wants lightweight structure (tags, due dates) without project management overhead.

## **4) Core User Stories & Acceptance Criteria**

1. **Create task in a day column with a weekly view on desktop and daily view on mobile**
   * *As a user*, I can add a task under a specific day (Today by default) with optional description, **priority (A/B/C)**, due date, and tags.
   * **AC:** Typing in the quick‑add row and pressing Enter creates the task at the end of that day’s list. Keyboard: Enter add, Shift+Enter newline in description modal.
2. **Mark task complete**
   * *As a user*, I can mark a task complete via a checkbox. Completed tasks remain in the day as crossed‑out (config in settings) or hidden.
   * **AC:** Checkbox toggles status = done; UI strikes through; filter can hide/show completed.
3. **Auto‑rollover at local midnight**
   * *As a user*, any **incomplete** tasks from *today* move to **tomorrow** automatically at local midnight, preserving metadata.
   * **AC:** At 00:00:05 local time, a scheduled job moves status = open tasks whose scheduled\_for = yesterday into scheduled\_for = today and increments their rollover\_count.
   * **AC:** Rollover respects user timezone stored in profile; backfills if app was offline (next launch computes missed midnights).
4. **Due date distinct from scheduled day**
   * *As a user*, I can set a **due date** independent of the day column a task is currently in (e.g., working ahead). Overdue tasks show the task in red color.
   * **AC:** Overdue = today > due\_date AND status != done shows in red color.
5. **Tags & filter**
   * *As a user*, I can tag tasks (#work, #home) and filter the week by tag. When a tag is selected, the weekly view shows only the tasks with those tags.
   * **AC:** Adding # in the input autocompletes tags; clicking a tag chips filters tasks.
6. **Search tasks & notes**
   * *As a user*, I can search text across **tasks and daily notes**; results show **specific day or days** and highlight matches.
   * **AC:** / focuses global search; search results list YYYY‑MM‑DD: snippet; clicking navigates to that day and scrolls to match.
7. **Daily Notes**
   * *As a user*, I can expand a **Notes** panel under each day to jot free‑form notes.
   * **AC:** Notes are rich‑text light (bold/italic/links) or plaintext in v1; auto‑save on blur.
8. **Drag & drop reorder / move between days**
   * *As a user*, I can reorder tasks within a day and drag across day columns.
   * **AC:** Persist position\_index within each day.
9. **Mobile responsive**
   * *As a user*, I can horizontally swipe between days; the Notes panel collapses into an icon.
   * **AC:** Today remains centered on load; 7‑day viewport scrolls.
10. Default view
    * By default, show the current weekly view with weekends on the right shown in a light grey color.
    * The daily view for tasks and notes show only columns with line like a lined notebook.
    * The tasks are shown with a checkbox and small tiny named tags on the bottom.
    * To add a task, the user simply clicks on the next empty line. Enter completes the add and shows the due date and tags icon for the user to add IF they want.

## **5) UX Spec (TeuxDeux‑style)**

* **Layout:** Horizontal scrollable week strip of day columns (Mon…Sun). Today is centered and visually accented.
* **Day Column:**
  + Header: DAY, Mon 12 + mini calendar picker.
  + Task list: minimalist rows, checkbox left, title, small due‑date chip, tags as chips below the task text and very small, priority badge (A/B/C).
  + **Quick Add** row at bottom.
  + **Notes panel** collapsible area labeled "Notes" with caret; expands inline below tasks. Notes panel is twice the size of the task column.
* **Top Bar:** App name, global search (/), filter by tag, “show completed” toggle, user menu.
* **Interactions:**
  + n toggles notes panel for the current day, c toggles show completed, / search.
  + Drag & drop between columns; keyboard move with [/] for previous/next day.

**Empty States**

* Today has a friendly prompt: “Add task. Press Enter to add.”

**Visual Tone**

* Clean, mono/duo‑tone, lots of whitespace; subtle color for overdue/priority.

## **6) Franklin‑Covey Behavior**

* **Priorities:** Optional A, B, C priority with implicit order A1, A2… B1….
* **Daily Planning:** Tasks are parked in a specific day column.
* **Rollover:** At local midnight, any status=open from the previous day are **migrated** to the next day, preserving priority and incrementing rollover\_count. Provide a subtle "Rolled over (x)" indicator.
* **Audit:** Keep a TaskHistory record for rollovers with timestamp.

## **7) Data Model (v1)**

User(id, email, timezone)

Day(id, user\_id, date)

Task(id, user\_id, title, description, status[open|done], priority[A|B|C|null],

scheduled\_for: date, due\_date: date|null, rollover\_count:int, position\_index:int,

created\_at, updated\_at)

Tag(id, user\_id, name)

TaskTag(task\_id, tag\_id)

DayNote(id, user\_id, date, content\_text, created\_at, updated\_at)

TaskHistory(id, task\_id, type[create|complete|reopen|rollover|edit], at: datetime, meta: json)

## **8) API Endpoints**

**Auth** (simple email‑magic link or mock in dev).

* GET /api/days?start=YYYY-MM-DD&end=YYYY-MM-DD → days with tasks & notes
* POST /api/tasks {title, scheduled\_for, due\_date?, priority?, tags?}
* PATCH /api/tasks/:id {title?, description?, status?, scheduled\_for?, due\_date?, priority?, position\_index?}
* POST /api/tasks/:id/tags {tag\_names: string[]}
* DELETE /api/tasks/:id soft delete in v1 optional
* PUT /api/day-notes/:date {content\_text}
* GET /api/search?q=... → [{type: 'task'|'note', date, id, snippet}]

**Rollover job**

* POST /internal/rollover (cron‑invoked): Moves incomplete tasks from D-1 to D for each user timezone bucket.

## **9) Rollover Logic & Timezone Handling**

* Run a cron at five minutes past each hour that processes users whose **local time** just crossed midnight (group by timezone).
* For each user/timezone:
  + Determine yesterday = local\_date - 1 and today = local\_date.
  + Select tasks where status=open AND scheduled\_for=yesterday.
  + Update scheduled\_for=today, rollover\_count++, append TaskHistory(rollover).
* **Backfill:** On login/app open, compute the number of midnights since last seen and apply rollovers for each missed midnight sequentially.
* Guard against duplicate rollovers by idempotency key (task\_id, target\_date) in TaskHistory meta.

## **10) Search**

* Simple ILIKE search on Task.title, Task.description, and DayNote.content\_text with ranking by recency + exact match.
* Return {date, snippet, anchor\_id} to jump to a day and scroll to result.

## **11) Non‑Functional Requirements**

* **Performance:** Load 14 days around today in <300ms server time; infinite scroll for older/newer days.
* **Reliability:** Rollover job is idempotent and retry‑safe.
* **Accessibility:** ARIA roles for lists, checkboxes; keyboard nav across days.
* **Security:** Row‑level security by user\_id.
* **Privacy:** No 3rd‑party trackers.

## **12) Analytics (optional v1)**

* Daily Active Users, tasks created/completed, rollover counts, search success rate.

## **13) Risks & Mitigations**

* **Timezone drift / DST:** Use luxon/Temporal for precise local dates.
* **Missed jobs:** Backfill logic on app open.
* **Over‑aggressive rollover:** Provide per‑user toggle to disable auto‑move (future setting).

## **14) Milestones**

* **M1 – Skeleton UI (1 week):** Horizontal day columns, add/complete tasks, notes panel (local mock).
* **M2 – Data & Search (1 week):** SQLite/Prisma, tags, due dates, search API.
* **M3 – Rollover (0.5 week):** Cron + idempotent server job + backfill.
* **M4 – Polish (0.5 week):** Drag/drop, keyboard shortcuts, mobile polish, perf.

## **15) Out of Scope / Future**

* Recurring tasks; multi‑user sharing; reminders; rich text notes; Kanban.

# **Technical Blueprint (for Cursor/Codegen)**

## **Stack**

* **Frontend:** Next.js (App Router) + React + Tailwind + shadcn/ui; dnd-kit for drag‑drop; react‑virtual for lists; luxon for time.
* **Backend:** Next.js API routes; **SQLite via Prisma** (swap to Postgres later); Vercel Cron for rollover or a Node worker (BullMQ) if self‑hosted.
* **Auth:** NextAuth email provider (dev: mock).

## **Suggested File Structure**

app/

layout.tsx

page.tsx (Today centered view)

day/[date]/page.tsx

api/

days/route.ts

tasks/route.ts

tasks/[id]/route.ts

day-notes/[date]/route.ts

search/route.ts

internal/rollover/route.ts

components/

DayColumn.tsx

TaskItem.tsx

NotesPanel.tsx

QuickAdd.tsx

Search.tsx

TagChips.tsx

lib/

db.ts (Prisma client)

dates.ts (local date helpers)

rollover.ts (idempotent logic)

prisma/

schema.prisma

## **Prisma Schema (excerpt)**

model User { id String @id @default(cuid()); email String @unique; timezone String }

model Day { id String @id @default(cuid()); userId String; date DateTime; @@unique([userId, date]) }

model Task {

id String @id @default(cuid())

userId String

title String

description String?

status String @default("open")

priority String?

scheduledFor DateTime

dueDate DateTime?

rolloverCount Int @default(0)

positionIndex Int @default(0)

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

tags TaskTag[]

}

model Tag { id String @id @default(cuid()); userId String; name String; @@unique([userId, name]) }

model TaskTag { taskId String; tagId String; @@id([taskId, tagId]) }

model DayNote {

id String @id @default(cuid())

userId String

date DateTime

contentText String

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

@@unique([userId, date])

}

model TaskHistory {

id String @id @default(cuid())

taskId String

type String

at DateTime @default(now())

meta Json

}

## **Rollover Pseudocode**

for (const tz of distinctUserTimezones()) {

const localNow = now().setZone(tz)

if (localNow.hour === 0 && localNow.minute < 10) {

const yesterday = localNow.minus({ days: 1 }).startOf('day')

const today = localNow.startOf('day')

const tasks = db.task.findMany({ where: { status: 'open', scheduledFor: yesterday } })

for (const t of tasks) {

if (await alreadyRolled(t.id, today)) continue

await db.task.update({ where: { id: t.id }, data: { scheduledFor: today, rolloverCount: { increment: 1 } } })

await db.taskHistory.create({ data: { taskId: t.id, type: 'rollover', meta: { to: today } }})

}

}

}

## **Search Endpoint (SQL idea)**

SELECT 'task' AS type, scheduled\_for AS date, id, snippet

FROM task

WHERE (title LIKE :q OR description LIKE :q)

UNION ALL

SELECT 'note' AS type, date, id, snippet

FROM day\_note

WHERE content\_text LIKE :q

ORDER BY date DESC

LIMIT 50;

# **Test Plan (Happy Path + Edge Cases)**

**Unit**

* Rollover moves only status=open tasks; preserves tags/priority; increments rollover\_count.
* Idempotency: repeated rollover on same (task, date) has no effect.
* Backfill applies multiple missed midnights.

**Integration**

* Creating tasks with due dates displays overdue badge when today > due\_date.
* Drag‑drop updates position\_index and scheduled\_for when moved between days.
* Search returns correct day; clicking navigates and highlights match.

**Edge Cases**

* DST transitions (23/25‑hour days).
* User changes timezone; next rollover uses new timezone.
* Offline usage: first launch after days away performs batch backfill.