assignment

Owner	ⓢ suyog trivedi
Tags	

📘 4-Hour Intern Assignment – Simple Tasks App

Goal: Build a very simple Tasks app with login, tasks list, and a tiny bit of backend logic.

You can use tools like Cursor, Lovable, Vercel, Supabase, Railway, Neon, etc. as you like.

We care about clear thinking, basic coding ability, and understanding of what you built.

🕒 Timebox

- Please spend maximum 4 hours.
- It's okay if you don't finish everything – just mention what's missing.
- Don't over-engineer; keep it small and clean.

🎯 Features (Keep It Simple)

Implement the features in this order. Stop wherever your 4 hours end.

1. Authentication (Basic)

- Simple signup + login (email + password is enough).
- Only logged-in users should see their tasks.
- You may use:
    - Your own auth, or
    - A hosted provider (e.g., Supabase Auth, NextAuth, Clerk, etc.)

-

2. Tasks CRUD

Each task should have:

- title (string, required)
- status (TODO / IN_PROGRESS / DONE)

Logged-in user can:

- Create a task.
- See a list of their own tasks.
- Update the status of a task.
- (Optional if time) Edit title or delete a task.

Requirement:

Each user should only see their own tasks, not tasks of other users.

3. Small Summary

On the main page, show a tiny summary for the logged-in user, like:

- TODO: X
- IN_PROGRESS: Y
- DONE: Z

Text counters are enough; charts are not required.

🏗️ Technical Requirements

Keep it minimal, but please follow these:

Backend

- Expose APIs (REST is fine) for:
    - Login/signup (or use provider SDK)
    - Create task
    - List tasks

    - Update task status
-
- Use any SQL database (e.g., Postgres, MySQL, Supabase, Neon, SQLite).
- Implement basic logging on the server:
    - For each task-related API, log at least:
        - HTTP method
        - URL/path
        - Timestamp
    -
    - Logging can go to:
        - Console, or
        - A DB table (whichever is easier)
    -
-

🔍 Deliberately open-ended:

You decide what extra fields (if any) are useful to log.

Frontend

- Simple UI (React / Next.js / any SPA is fine) that lets the user:
    - Sign up / log in.
    - See their tasks list.
    - Create a new task.
    - Change status of a task.
    - See the summary counts.
-

UI does not need to be beautiful – basic and usable is okay.

(Small) Caching Requirement

Implement one very small caching behavior:

- When the user loads the tasks list, cache the result in memory on the server (or equivalent) for 30 seconds per user.
- Requests within 30 seconds should reuse the cached result instead of hitting the database again.

You can implement this in any simple way (e.g., a JS Map / in-memory dict).

If you run out of time, describe in the README how you planned to implement it.

📦 What You Must Submit

1) GitHub Repo

- Put your code in a public GitHub repo.
- Single project or /backend + /frontend – your choice.

2) README.md (Important)

Please include a README.md with the following sections:

A. How to Run

- Steps to run the app locally (backend + frontend).
- Mention any services used (e.g., Supabase, Vercel, etc.).

B. What is Done / Not Done

- Bullet points: what you completed.
- Bullet points: what you didn’t complete due to time.

C.

Architecture in My Own Words

(5–10 lines)

Explain in simple language:

- How does login work in your app?
- How are tasks stored and fetched? (mention tables/entities)

- Where did you put the caching logic, and how does it work roughly?

Please avoid generic textbook answers – describe your actual implementation.

D. Short Reflections

Answer briefly (2–4 lines each):

1. Caching:
   How did you implement the 30-second cache? In which file/function?
2. Security:
   How do you ensure one user cannot access another user's tasks?
3. Bug you faced:
   Describe one bug or issue you faced while building this, and how you debugged it.
4. If you had 1 more hour:
   What would you improve or add next?