GenAI.Labs Challenge: SprintSync

The Scenario

A fast-moving AI consultancy needs a lean internal tool—**SprintSync**—so its engineers can log work, track time, and lean on an LLM for quick planning help. They also want the repo to double as a reference implementation of clear architecture, tight DevOps, and thoughtful UI. You've been asked to deliver an MVP that **works end-to-end in the cloud** and showcases how you reason about problems.

---

1 · Core Scope

Layer	What to build	Why it matters
Backend	*Your choice of stack (FastAPI, Express, Django, etc.).* • Entities: **User** (isAdmin flag) and **Task** (title, description, status, totalMinutes). • Endpoints: auth (JWT or secure cookie), CRUD for users & tasks, status transition, **/ai/suggest** (see AI below). • Auto-generated docs (Swagger/Redoc or GraphQL Playground).	Shows you can model data, expose clean APIs, and document them.
AI Assist	Implement **one** of: 1. *Draft task description* from a short title, **or** 2. *Return a concise daily plan* for the signed-in user. May call a live LLM (OpenAI, Anthropic, open-weights) or a stub that returns deterministic JSON.	Demonstrates prompt/response design and graceful degradation.

Frontend	SPA in React / Vue / Svelte. • **Task list view** with inline status change (To Do → In Progress → Done). • Form/modal to create & edit tasks. • Button to invoke **/ai/suggest** and display the result. • Basic responsive styling—clarity over flair.	Confirms you can connect front & back seamlessly and think about UX.
Observability	Structured logs for every API call (method, path, userId, latency). Log errors with stack traces.	Proves production-readiness and debugging hygiene.
DevOps & Deploy	• Dockerfile + docker-compose (app + DB). • Deploy to any free/low-cost cloud (Render, Railway, Vercel, Fly, AWS EC2 + RDS, etc.).	Validates you can run code outside of "localhost."
Database	Relational or NoSQL—seed with a few users & tasks for demo.	Shows schema thinking and data persistence.
Docs & Demo	• **README** with setup, tech choices, and live URL. • **estimates.csv** (see section 5). • **Loom / screen-share video** (≤ 5 minutes total; multiple clips OK) that must include: 1. End-user demo of the app 2. Walkthrough of architecture & deployment 3. Brief code tour showing repo structure and any noteworthy implementations	Lets us assess communication skills and decision rationale.

---

2 · Stretch Goals (optional extras)

You are welcome to explore including one or more of these features to showcase your best skillset if time allows:

- **UX / UI** – Kanban drag-and-drop board; analytics chart (time logged per day per user).
- **Backend** – /stats/top-users endpoint; admin can assign tasks to others.
- **AI** – Vector DB + simple RAG flow to auto-assign tasks to most suitable team member based on résumé snippets of the team members and the task description
- **Ops** – Separate logging stack (Loki/Grafana, ELK); CI pipeline; unit / integration tests.

---

3 · Submission Checklist

1. **GitHub Repository**
    - Public or private-invite repo containing all source code.
    - Meaningful commit history (≈ 10+ commits) that shows your thought process.
    - estimates.csv committed first, then updated with actual logged time as you work.
2. **Live Deployment**
    - URL that we can open without VPN or credentials (demo seed data is fine).
3. **Loom / Screen-Share Video**
    - Up to 5 minutes total (feel free to split into multiple clips).
    - Must clearly cover: product demo → architecture & deploy → quick code tour.
    - Include the link in the README and in your email reply.
4. **Email Reply**
    - Send the repo link, live app URL, and video link.
    - Add any notes or questions you'd like us to consider.

---

4 · Estimation, Versioning, and Thinking Out Loud

Feel very free—and encouraged—to think out loud. Ask questions. Commit as often as you usually would on any standard project.

Time Estimates

The first task is essential, and often overlooked as an important and difficult quality of advanced engineering. Spend some preliminary drawing board time to break this project into smaller pieces, and estimate how long each will take you. Update the attached CSV with your estimated information as your first commit. Time yourself and update this file as time goes on, with a new column showing actual logged time so it can be compared to the initial guess. Good engineers are bad at this sometimes, because of the notorious unpredictability of unknowns in technical projects. We are looking at your process, and promise we will not be phased by poor estimates.

Versioning and Git History

We encourage you to commit regularly and meaningfully to demonstrate how your thinking evolved. Use version tags (e.g., v0.2: data loading, v0.5: basic chatbot integration) to mark important milestones. A thoughtful commit history (typically around 10 commits or more) helps us follow your process — please avoid submitting everything in one dump at the end.

---

5 · Evaluation Criteria

Category	What we're looking for	Weight
System Design & Reasoning	Clarity of architecture, data modeling, and trade-off explanations. Do you understand why you chose X over Y?	30 %
Code Quality, Structure & Testability	Readable, modular code with clear separation of concerns. Easy to extend and reason about, even in prototype form.	25 %
Functionality & Output Quality	Does SprintSync run end-to-end? Does the AI endpoint return sensible results? Are edge cases handled?	20 %

UI Functionality & UX	Clean, intuitive task list; AI interaction surfaced well; responsive basics; any bonus UI like charts earns extra nods.	15 %
Process, Versioning & Thoughtfulness	Meaningful commits, reflective comments/questions, and overall communication clarity.	10 %

Note on Stretch Features & Evaluation Philosophy

Electing to include some of the stretch features will be considered in our evaluation criteria as well. They're a great way to show what you know, push your thinking a bit further, or highlight strengths that might not surface in the core flow. That said, they're not required, and they're not the point — you can do very well on this challenge without them.

We understand that candidates are juggling other commitments. If you absolutely need to simplify part of the project — *document what you did and why.* We're evaluating your thinking, not just your code.

What we care most about is how well your solution works, how clearly you understand it, and how thoughtfully you approach the problem.

A Note on AI Tool Usage

We actively encourage the use of AI coding assistants (like Cursor, GitHub Copilot, or others)—we use them daily ourselves. However, many applicants fail this challenge because they rely on AI-generated code without understanding it deeply. They can't explain their decisions live, debug issues, or extend their solution when asked.

Please feel free to use these tools—but make sure you understand your code inside and out. We're evaluating *your thinking*, not the AI's.

Questions? We're all ears. Have fun building **SprintSync**!