CodeStratLabs 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 (Required unless noted)	Why it matters
Backend	API in **FastAPI / Express / Django** (pick one). Entities: **User** (isAdmin) and **Task** (title, description, status, totalMinutes). Endpoints: auth (JWT or secure cookie), CRUD for users & tasks, status transition, **/ai/suggest**. Auto-generated docs (Swagger/Redoc or GraphQL Playground).	Shows data modeling, clean API design, and self-documenting interfaces.
AI Assist	/ai/suggest implements **either** a. Draft task description from a short title, **or** b. Return a concise daily plan for the signed-in user. Provide (a) live LLM call **and** (b) deterministic stub JSON for tests/CI.	Tests prompt/response design and graceful degradation.

Observability	Structured log per request (method, path, userId, latency). Include stack traces on errors.Expose **/metrics** (Prometheus-style JSON is fine) *or* OpenTelemetry exporter.	Proves production readiness and debugging hygiene.
DevOps & Deploy	Dockerfile + docker-compose (app + DB). Deploy to any free/low-cost cloud (Render, Railway, Fly, Vercel, simple AWS EC2 + RDS, etc.).	Validates you can run code outside "localhost."
Database	Relational or NoSQL. Supply schema.sql / migrations. Seed with demo data.	Demonstrates schema thinking & persistence.
Testing	≥ 2 unit tests (happy paths) **and** 1 integration test hitting /ai/suggest **stub**.	Shows reliability focus.
Layer	What to build (Required unless noted)	Why it matters

---

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:

AI

- Vector DB + RAG proof-of-concept that auto-assigns a new task to the "best-fit" user using résumé snippets.

UI / UX

- Tiny SPA (React / Vue / Svelte) that lists tasks and hits your API.

Backend / Data

- /stats/top-users aggregate endpoint (e.g., top 5 by minutes this week).
- Extra query such as “average cycle time per status.”

Ops

- CI pipeline (lint → tests → Docker build) on every push.
- Separate logging stack (Grafana/Tempo, Loki, or ELK) with a single dashboard panel.

---

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. Download a copy of the attached CSV and update it 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	Architecture clarity, data model quality, trade-off explanations	**35 %**
Code Quality & Testability	Readable, modular code; sensible separation of concerns; solid unit & integration tests	**30 %**
Functionality	End-to-end flow works, including AI stub; sensible edge-case handling	**20 %**

Observability & DevOps	Structured logs, metrics endpoint, Docker, successful cloud deploy	**10 %**
Process & Communication	Meaningful commits, clear Loom walkthrough	**5 %**

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**!