Laburen.com – Product Engineer Challenge

🎯 Objective

Build a functional AI Agent that demonstrates your skills as an AI Product Engineer and DevOps professional.
The agent must:

- Enable knowledge management through RAG
- Persist all conversation data in PostgreSQL using Prisma
- Deliver a real-time streaming experience with visual tool states

We want to see how you think end-to-end: from code architecture → automated deployment → user experience → system observability.

📋 Functional Requirements

1. Complete Streaming Chat Experience

- Real-time conversation with streaming
- Custom streaming components showing different tool states:
  - "Searching documents..."
  - "Uploading file to RAG..."
  - "Processing embedding..."
  - Visual progress for each operation
- Clear indicators when the agent is:
  - Thinking
  - Using a tool
  - Responding
- Display model reasoning in real-time
  - Show the model's reasoning / thinking process as it streams
  - Users must see how the model thinks and reasons live
- MANDATORY: Use Vercel AI SDK with streaming

2. Complete Persistence with Prisma + PostgreSQL

- Persist everything:
  - User messages
  - Agent responses
  - Tool calls
  - Tool states
  - Reasoning traces

- On page reload or returning to the chat → state must be identical:
    - Same message history
    - Same tool states displayed
    - Same conversation context
    - Reasoning / thinking process preserved
- Users must be able to create NEW chats from scratch
    - Clear UI to start a new conversation
    - Ability to switch between different chats
    - Each chat maintains its own independent history and context
- Well-designed Prisma schema covering:
    - Conversations / Chats
    - Messages (role, content, tool_calls, reasoning, etc.)
    - Tool states
    - Streaming metadata

3. RAG Tools (2 required)

Tool 1: Upload to RAG

- Upload files (PDF, TXT, MD, DOCX) to the RAG system
- Process and generate embeddings
- Save to vector database
- Show progress in streaming UI with custom component

Tool 2: Search in RAG

- Search for information in indexed documents
- Display sources / references clearly:
    - Document name
    - Exact fragment / chunk
    - Similarity score
    - Chunk metadata (if available)
- Render sources in a clean, readable way in the chat

Note on RAG: Use any vector database you prefer (pgvector, Pinecone, Chroma, Qdrant, etc.). What matters is that it works reliably.

4. User Experience

- Clear visualization of:
    - User messages
    - Agent responses
    - Tool calls in progress (custom components)
    - Information sources
    - Model reasoning / thinking process
- Well-handled loading states and errors
- Ability to create and switch between multiple chats
- Responsive and accessible design

🛠️ Required Tech Stack

Category	Required Technology
Frontend	Next.js (Pages Router)
AI Framework	Vercel AI SDK (mandatory)
UI Components	Vercel AI Elements (mandatory)
Database	PostgreSQL
ORM	Prisma
Vector Database	Your choice (pgvector, Pinecone, Chroma…)
LLM Provider	OpenAI, Anthropic, Azure OpenAI, Ollama…

🚀 DevOps Requirements

Task 1: Deploy with Maximum Automation
Show everything you know about DevOps in the deployment process.
Document in the README:

- Which automations you implemented
- Why you chose each tool / service
- How the deployment flow works

Task 2: DevOps Reflection on the Agent
After completing the challenge, write a document (`devops-proposal.md`) where you reflect and propose improvements for production.

Guiding Questions:

1. Observability: How would you monitor this agent in production? (metrics, logs, traces)
2. Scalability: What would you do if traffic increases 1000×?
3. Reliability: How would you ensure 99.9% uptime?
4. Cost optimization: How would you optimize costs for LLM calls, vector DB, and hosting?
5. Security: What vulnerabilities do you see and how would you mitigate them?
6. Testing: What testing strategy would you propose for this kind of system?
7. Advanced CI/CD: What improvements would you make to the pipeline you built?
8. Disaster recovery: What backup and recovery plan would you implement?

Expected format:

- Analysis of the current architecture
- Concrete proposals with pros/cons
- Diagrams if helpful (architecture, flows, etc.)
- Proposed tech stack for production
- Prioritized implementation roadmap

This is not mandatory to implement — we just want to see how you think as a DevOps engineer when facing an AI system in production.

📊 Evaluation Criteria

Criterion	Weight	What we evaluate
Architecture & Code Quality	15%	Clarity, patterns, organization, Prisma schema
AI SDK + Tools Integration	15%	Streaming, tool calling, state management, reasoning display
RAG + Source Citation	10%	Effective retrieval, clear & useful source presentation
Persistence & State Management	10%	Prisma usage, conversation recovery, consistency, multi-chat support
UX / Streaming Experience	10%	Custom components, fluid feel, reasoning visualization
DevOps & Deployment	15%	Automation, CI/CD, functional public deployment
DevOps Reflection	10%	Quality of analysis and realistic proposals
Conceptual Design Document	10%	Quality and clarity of conceptual_design.md
Extras / Creativity	5%	Bonus points for thoughtful additions

⏱ Estimated Time

7 calendar days from receiving the assignment.
Focus on delivering a functional MVP that covers the core requirements. Extras are welcome but optional.

📦 Final Delivery

1. GitHub Repository
   Must include:

   - Complete, well-documented code
   - README containing:
     • Setup instructions
     • Tech stack overview
     • Key architecture decisions
     • How to run locally
     • How to deploy
   - `devops-proposal.md` — your DevOps reflection
   - `conceptual_design.md` —(MANDATORY):
     It must describe the architecture of the agent you are building.
     It must include:
     1 Explanation of the design approach
     2 Reasoning and execution flow of the agent
     3 Definition of tools
     4 Diagrams using **Mermaid** (Markdown-based diagram syntax) representing:
       - System architecture
       - Agent flow
   - `.env.example` file
2. Demo Video (3–5 minutes)
   Record (Loom, YouTube, etc.) showing:

   - Full agent usage flow
   - Uploading a document to RAG
   - Searching with visible sources
   - Page reload → preserved conversation
   - Creating a new chat & switching between chats
   - Streaming components & tool states in action
   - Model reasoning / thinking process visible live
   - Deployment pipeline (optional but adds value)
3. Public Deploy

   - Working public URL to test the agent
   - Must be live and accessible
   - HTTPS enabled (free on Vercel, Railway, Render, etc.)

🎯 What We're Looking For in an AI Product Engineer

1. Product vision — you understand the problem and build a coherent solution
2. AI-first integration — comfortable with LLMs, streaming, tools & RAG
3. Full-stack ownership — frontend + backend + database + AI working together
4. DevOps mindset — thinking about deployment, automation, observability
5. Technical judgment — justified decisions, not just following tutorials
6. Execution speed — delivering a solid MVP under time pressure

Good luck!
We're excited to see what you build. 🚀