Fullstack test brief

Project Overview

Design and implement a scalable, real-time Kanban board application for managing candidate workflows, demonstrating your technical expertise and problem-solving approach.

The project you will start with is based on the stack that you would use if you join us.

Backend	Frontend
Elixir	React
Phoenix	Typescript
PostgreSQL	Vite / Vitest

Deliverables

Please submit your work as a Git repository that includes your complete implementation, comprehensive README documentation detailing your approach, a test suite demonstrating the functionality, with an optional live demo URL where we can see the application in action.

Time Management

While we appreciate thoroughness, focus on delivering a functional core solution first, then enhance with additional features as time permits. Quality over quantity is valued.

Notes

- Feel free to make assumptions where requirements are unclear, but document them
- Innovation in your approach is encouraged
- Consider real-world usage scenarios
- Focus on maintainability and scalability
- Document any known limitations or areas of improvement

ℹ️ **Not all upcoming requirements have to be implemented; you can detail your approach in a document (README, Notion, etc…)**

Requirements

1. Basic Functionality

- Implement drag-and-drop functionality for cards:
  - Within the same column
  - Between different columns
- Ensure proper handling of card positioning and ordering
- Maintain data consistency

2. Real-time Collaboration

- Implement real-time synchronization between users
- Handle concurrent operations gracefully
- Choose and justify your real-time solution (WebSocket, SSE, etc.)

3. Performance Optimization

- Design for scale: handle 10,000+ candidates efficiently
- Implement frontend optimizations (virtualization, pagination)
- Optimize backend operations and database queries
- Consider caching strategies

4. Customization & Extensibility

- Add candidate view
- Support dynamic column creation and management
- Design for future feature additions

Evaluation Criteria

🧑‍💻 Code Quality & Architecture

- Clean, maintainable, and well-documented code
- Proper error handling and edge cases
- Comprehensive test suite and high coverage
- Thoughtful architectural decisions

🎛️ Version Control

- Clear, atomic commits with meaningful messages
- Organized branching strategy

📝 Technical Documentation

Detailed README including: Setup instructions (If there is no live demo URL provided, we should be able to set up and run the application within a minute.), architecture overview, technical decisions and trade-offs, future improvements, etc...

✨ Bonus

- Online demo deployment
- Docker containerization
- E2E testing (e.g., Cypress)
- CI/CD pipeline setup
- Modern tooling and best practices
- Security considerations