InterviewDeck

Senior Frontend Engineer

Aperture Pay (fictional)

Interview: 2026-05-20T10:00:00Z

Walk in tomorrow having predicted the likely questions. Every answer below is grounded only in what you actually told us about yourself or what the job posting actually says.

Likely questions (15)

Behavioral

* Walk me through how you led a design-system migration effort from start to finish — what was your approach to planning, sequencing, and de-risking the work?

Tied to: "leading design-system migration efforts"

* Tell me about a time you shipped a large refactor behind a feature flag with a progressive rollout. How did you decide when to promote each stage?

Tied to: "own the design-system migration end-to-end"

* Describe your experience participating in an on-call rotation. How did you reduce alert noise and improve the signal-to-noise ratio for your team?

Tied to: "participate in the platform on-call rotation"

* Tell me about a time you mentored a junior engineer. What did you focus on and how did you measure their growth?

Tied to: "Prior work mentoring junior engineers"

* Have you done any public technical writing or open-source contributions? Tell me about a piece of work you're proud of and the impact it had.

Tied to: "Public technical writing or open-source contributions"

Technical screen

* What are your strong opinions about React performance? Give me a concrete example where you identified and resolved a performance bottleneck.

Tied to: "Strong opinions about TypeScript, React performance, and testing"

* How do you approach TypeScript in a large codebase — what patterns do you enforce and what tradeoffs do you accept?

Tied to: "Strong opinions about TypeScript, React performance, and testing"

* Walk me through how you would migrate a shared component library from styled-components to a zero-runtime CSS-modules pipeline without breaking consumers.

Tied to: "migration of our shared component library from styled-components to a modern zero-runtime CSS-modules pipeline"

* How do you think about testing strategy for a shared component library — what layers of testing do you prioritise and why?

Tied to: "Strong opinions about TypeScript, React performance, and testing"

Role-specific

* How have you defined and maintained reliability SLOs for a user-facing product? What signals did you choose and why?

Tied to: "set reliability SLOs for our public dashboards"

* Have you worked in payments or financial infrastructure before? How did that domain shape the way you think about reliability and correctness?
Tied to: "Experience with payments or financial infrastructure"

* Describe a situation where you had to build consensus across multiple teams to adopt a new frontend standard or tooling change.
Tied to: "lead the migration of our shared component library"

* How do you approach setting performance budgets for a frontend application, and how do you enforce them over time?
Tied to: "5+ years building production frontend at scale"

* How do you handle an incident during an on-call shift where the root cause is unclear and multiple teams are involved?
Tied to: "participate in the platform on-call rotation"

Culture fit

* Aperture Pay describes reliability as its north-star metric. How does that value align with the way you personally prioritise your engineering work?
Tied to: "Comfort with reliability SLOs and on-call rotations"

STAR drafts (4)

STAR 1 — Walk me through how you led a design-system migration effort from start to finish — what was your approach to planning, sequencing, and de-risking the work?

Grounded in: "Led the migration of our shared component library from styled-components to CSS-modules across 14 packages, shipping behind a feature flag with progressive rollout. Zero customer-visible regressions."

Situation:
At Lattice, the shared component library had grown across 14 packages all using styled-components, creating a performance and maintainability burden for the platform team.

Task:
I was asked to lead the end-to-end migration of the library to CSS-modules while ensuring no customer-visible regressions.

Action:
I planned the migration package-by-package, shipping each change behind a feature flag with a progressive rollout so that internal teams could opt in incrementally. I coordinated with consuming teams to validate each package before promoting the flag, and set up automated regression checks to catch visual or functional breakage early.

Result:
We completed the migration across all 14 packages with zero customer-visible regressions, validating the progressive rollout strategy.

STAR 2 — Describe your experience participating in an on-call rotation. How did you reduce alert noise and improve the signal-to-noise ratio for your team?

Grounded in: "Owned reliability SLOs and on-call rotations for the platform team. Reduced critical-alert noise by half by introducing severity tiers."

Situation:
While on the platform team at Lattice, the on-call rotation was generating a high volume of alerts, many of which did not require immediate action, leading to engineer fatigue.

Task:
I was responsible for owning reliability SLOs and the on-call rotation and needed to make the alert load sustainable.

Action:
I introduced severity tiers to classify alerts by their actual customer impact, which allowed the team to distinguish critical pages from informational noise. I audited existing alerts, reclassified them under the new tiers, and updated runbooks to match the new severity model.

Result:
Critical-alert noise was reduced by half, making the on-call rotation significantly more manageable

STAR 3 — Tell me about a time you mentored a junior engineer. What did you focus on and how did you measure their growth?

Grounded in: "Mentored two junior engineers across two quarters."

Situation:

At Earlier Co., I was a software engineer working on a React-based dashboard while two junior engineers joined the team.

Task:

I was asked to mentor both engineers across two quarters to help them ramp up and become productive contributors.

Action:

I worked closely with both engineers through code reviews, pair programming sessions, and regular one-on-ones, focusing on React patterns, code quality, and debugging approaches. I gave structured feedback on their pull requests and helped them take on progressively more complex tasks.

Result:

Both engineers grew into independent contributors over the two quarters, able to own features end-to-end with minimal guidance.

STAR 4 — Tell me about a time you shipped a large refactor behind a feature flag with a progressive rollout. How did you decide when to promote each stage?

Grounded in: "shipping behind a feature flag with progressive rollout. Zero customer-visible regressions."

Situation:

The Lattice component library migration needed to reach all 14 packages without disrupting any of the teams consuming those packages in production.

Task:

I needed to design a rollout strategy that let us ship incrementally and roll back safely if anything went wrong.

Action:

I implemented a feature flag system that allowed each package's migration to be toggled independently, enabling a progressive rollout where teams could opt in at their own pace. I monitored each promotion step for regressions before advancing to the next package, and kept stakeholders informed of progress throughout.

Result:

The progressive rollout approach meant we could catch and address any issues before they reached all consumers, ultimately achieving zero customer-visible regressions across the full migration.

Company context

Aperture Pay is a developer-focused payments infrastructure company whose customers build financial products on top of its APIs. The company treats reliability as its core engineering value and is actively scaling across multiple teams. It invests in design-system tooling, performance budgets, and learning from incidents.

* Aperture Pay recently launched its developer SDK, signalling an active phase of product expansion and developer ecosystem growth.

From: "We launched our developer SDK in March."

* Reliability is explicitly the company's north-star engineering metric, meaning SLO ownership and on-call discipline will be central to this role.

From: "Engineering values reliability as our north-star metric"

* The company is hiring across three teams simultaneously, suggesting significant growth and the likelihood of cross-team collaboration on shared infrastructure.
From: "we're hiring across three teams to scale our payments infrastructure"

* Dashboard reliability is treated as a direct dependency for customers' ability to run their own businesses, raising the stakes for frontend SLO ownership.
From: "the reliability of our dashboards directly affects their ability to support their own customers"

* Aperture Pay explicitly invests in design-system tooling, which aligns directly with the migration work described in the JD and Maya's Lattice experience.
From: "We invest in design-system tooling, performance budgets, and incident-driven learning."

* The company uses incident-driven learning as a named practice, suggesting a blameless postmortem culture and an expectation that engineers contribute to systemic improvements after outages.
From: "incident-driven learning"

* Performance budgets are a stated investment area, meaning candidates should be prepared to discuss how they set, monitor, and enforce frontend performance constraints.
From: "performance budgets"

Questions to ask the interviewer

* How does the team currently define and measure the SLOs for the public dashboards, and what tooling do you use to track them?
(Tied to: the job description)

* What does the on-call rotation look like today — how many engineers share it, what is the typical alert volume, and how do you run postmortems?
(Tied to: the job description)

* You mention incident-driven learning as a core practice — can you walk me through how a recent incident shaped a change in how the team builds or operates?
(Tied to: the company context)

* How far along is the design-system migration, and what are the biggest technical or organisational blockers the team has encountered so far?
(Tied to: the job description)

* Since reliability is the north-star metric, how does the engineering organisation balance shipping new features against maintaining and improving reliability?
(Tied to: the company context)

Honest gaps to acknowledge

Experience with payments or financial infrastructure
Acknowledge that your background has been in HR and people-management SaaS rather than payments, but highlight that the reliability and scale challenges you have solved — SLOs, on-call, large-scale migrations — transfer directly, and express genuine enthusiasm for learning the payments domain.