Puredot India

Forward Deployed Engineer — Interview Scorecard

Executive Briefing: Why This Scorecard Matters

Forward Deployed Engineers sit at the intersection of product, engineering, and the customer relationship. They are often the first technical presence a customer experiences post-sale, and their judgment directly shapes retention, expansion, and the credibility of the entire technical organization in the field. A mis-hire in this role is expensive in ways a bad hire elsewhere is not: it shows up as stalled deployments, eroded customer trust, and engineering effort spent on rework rather than reusable product value.

This scorecard operationalizes the five traits that most reliably separate strong FDEs from the rest: **Customer Empathy**, **Radical Ownership**, **Problem Decomposition**, **Product Sense**, and **Ambiguity Handling**. Each is paired with proven interview prompts and concrete signals, so evaluations are consistent across interviewers, defensible in debrief, and comparable across candidates and cohorts.

For leadership, the payoff is threefold: **faster, more confident hiring decisions** through a shared evaluation language; **lower mis-hire risk** in a role with outsized customer and revenue impact; and **a hiring bar that compounds**, since every completed scorecard sharpens calibration for the next panel. In short, this guide turns FDE hiring from an intuition call into a structured, repeatable decision.

Candidate: _____ **Interviewer:** _____ **Date:** _____

Rating scale: 1 = Strong No | 2 = No | 3 = Mixed | 4 = Yes | 5 = Strong Yes

---

1. Customer Empathy

Sample questions:

- Tell me about a time a customer's stated request wasn't actually their real problem. How did you find out?
- Describe a time you had to sit with a customer's workflow before building anything. What did you learn that changed your approach?
- How do you validate that a shipped solution actually solved the customer's pain?

Look for:

- Goes beyond the ticket/spec to understand context and root need
- Can describe the problem in the customer's language, not just technical terms
- Considers downstream/indirect stakeholders, not just the requester
- Follows up post-delivery to confirm impact

Signal	Rating (1-5)	Notes
Understood root need vs. stated ask		
Considered full set of affected users		
Validated outcome post-delivery		

Overall Customer Empathy score: ___/5

---

2. Radical Ownership

Sample questions:

- Tell me about a time something broke in production (or with a customer) that wasn't technically "your fault." What did you do?

- Describe a project where you had to work outside your defined role to get the customer to a good outcome.
- Tell me about a time you had to escalate a problem. What made you decide to escalate rather than solve it quietly?

Look for:

- Speaks in terms of "I" accountability, not "the team" or "they"
- Proactively surfaces risk instead of waiting to be asked
- Fills gaps between functions (support, sales, product) without being told to
- Owns mistakes directly, focuses on fix over blame

Signal	Rating (1-5)	Notes
Accountability language ("I" vs "they")		
Proactive risk/blocker surfacing		
Follow-through to actual resolution		

Overall Radical Ownership score: ___/5

---

3. Problem Decomposition

Sample questions:

- Walk me through how you'd break down a vague request like "help us integrate with the customer's data pipeline."
- Tell me about a messy, ambiguous technical problem you had to structure before solving.
- How do you decide what the smallest valuable first step is?

Look for:

- Breaks vague asks into concrete, testable sub-problems
- Identifies smallest slice that delivers real value (not "build everything" first)
- Separates data/UX/integration/process concerns clearly
- Sequences work to de-risk early

Signal	Rating (1-5)	Notes
Structured breakdown of ambiguous problem		
Identified minimal valuable first step		
Clear, non-technical explanation of plan		

Overall Problem Decomposition score: ___/5

---

4. Product Sense

Sample questions:

- Tell me about a time you built something for one customer that could (or should) become a general product feature.
- Describe a situation where you pushed back on a customer request because it wasn't the right long-term solution.
- How do you decide between a scrappy one-off fix and a durable, reusable solution?

Look for:

- Thinks about generalization and reuse, not just the immediate fix
- Constructively pushes back on requests that hurt the product long-term
- Weighs maintainability, adoption, and support burden

- Feeds field learnings back to core product/eng teams

Signal	Rating (1-5)	Notes
Considered generalizability of solution		
Weighed long-term cost vs. short-term ask		
Closed the loop with product/eng teams		

Overall Product Sense score: ___/5

---

5. Ambiguity Handling

Sample questions:

- Tell me about a project that started with almost no spec. How did you get moving?
- Describe a time your plan had to change mid-project because new information came in.
- How do you decide when to ask a clarifying question vs. just make a call?

Look for:

- Starts work productively with incomplete information
- Makes and documents reasonable assumptions instead of stalling
- Recognizes when to ask vs. when to decide independently
- Communicates uncertainty honestly rather than overselling confidence

Signal	Rating (1-5)	Notes
Made progress despite incomplete spec		
Adapted quickly to new/contradicting info		
Communicated uncertainty transparently		

Overall Ambiguity Handling score: ___/5

Summary

Competency	Score (1-5)
Customer Empathy	
Radical Ownership	
Problem Decomposition	
Product Sense	
Ambiguity Handling	
Total	**___/25**

Overall recommendation:

■ Strong Hire ■ Hire ■ No Hire ■ Strong No Hire

Key strengths:

Key concerns:

Additional notes: