**Frontend Developer Interview Scorecard**

Candidate Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Role Interviewed For: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Interviewer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Dimension** | **Guidance** | **Score (1–5)** |
| Frontend Implementation | 1-2: Struggles to convert designs into working UI; frequent layout or interaction bugs.
3: Implements components from designs using framework conventions and handles common props/state.
4: Builds responsive, accessible components and handles edge cases and error states.
5: Creates reusable component patterns, improves UI performance, and mentors peers on frontend best practices. |  |
| Backend Implementation | 1-2: Cannot implement or explain basic routes, data flow, or persistence; frequent runtime errors.
3: Implements endpoints and database interactions following established patterns and handles validation.
4: Designs clear APIs, handles errors and edge cases, and reasons about data models and performance.
5: Anticipates scaling concerns, proposes schema or API improvements, and drives reliable backend choices. |  |
| Problem Solving & Algorithms | 1-2: Cannot decompose simple problems or produces incorrect solutions with no testing.
3: Breaks problems into steps and implements correct solutions with reasonable complexity.
4: Selects efficient approaches, explains trade-offs, and handles edge cases proactively.
5: Simplifies complex problems, proposes robust algorithms, and anticipates future constraints. |  |
| Code Quality & Testing | 1-2: Delivers unstructured code with minimal or no tests and unclear naming.
3: Writes readable code, follows style conventions, and adds unit tests for main logic.
4: Produces well-factored code, includes integration tests, and refactors to reduce duplication.
5: Introduces clear testing patterns, improves codebase maintainability, and mentors on best practices. |  |
| Collaboration & Communication | 1-2: Provides unclear updates, misses context, and rarely asks clarifying questions.
3: Communicates status, asks for help when blocked, and documents basic decisions.
4: Clearly explains trade-offs, proactively syncs with teammates, and writes useful PR descriptions.
5: Facilitates team alignment on tasks, drives clear design discussions, and improves team processes. |  |
| Learning & Ownership | 1-2: Avoids unfamiliar tasks and rarely incorporates feedback.
3: Seeks feedback, learns new technologies, and completes assigned tasks reliably.
4: Takes ownership of small features, iterates on feedback, and improves processes.
5: Proactively identifies gaps, drives improvements beyond assigned scope, and mentors other juniors. |  |
| Tools & Deployment Basics | 1-2: Cannot use version control or run the app locally without heavy help.
3: Uses git effectively, runs local environment, and opens clear PRs.
4: Diagnoses CI issues, understands basic deployment steps, and improves dev scripts.
5: Automates repetitive tasks, contributes to CI/CD stability, and documents deployment processes. |  |

 **Overall Evaluation**

**Strengths Observed:**

**Concerns / Weaknesses:**

**Recommendation (Yes / No / With Reservations):**

**Final Score (Avg / Weighted):**