EchorTech – Developer Guidelines & Engineering Standards

Version: 1.0 | Purpose: This document serves as the authoritative onboarding and reference guide for all engineers at EchorTech. It defines the minimum engineering standards, expectations, and behaviors required to build reliable, scalable, and secure systems.

1. Engineering Philosophy

- Engineering at EchorTech prioritizes fundamentals over frameworks, clarity over cleverness, and ownership over delegation.
- Strong fundamentals mean the ability to read, reason about, debug, test, and evolve existing systems confidently.
- Every engineer is accountable for the quality, stability, and long-term health of the code they contribute.

2. Responsibility & Ownership

- Begin development only after fully understanding the requirements and edge cases. Clarify ambiguity before writing code.
- Developers own their changes end-to-end: design, implementation, testing, deployment awareness, and post-release behavior.
- If a defect escapes to production, the focus is on understanding why it happened and preventing recurrence, not assigning blame.
- Engineers must be able to explain what their code does, why it exists, how it can fail, and how it can be debugged.

3. Testing & Quality Standards

- Testing is a mandatory developer responsibility; it is not delegated or optional.
- Appropriate testing must be applied based on context: unit tests, integration tests, end-to-end tests, or documented manual verification.
- Code is considered incomplete if its failure modes and edge cases have not been evaluated.
- A feature that passes review but fails basic scenarios is treated as unfinished work.

4. Responsible LLM Usage

- LLMs are productivity tools, not sources of truth. All outputs must be reviewed and validated.
- Use clear task descriptions, step-by-step prompting, and constrained output formats.
- Generate tests or validation steps before relying on generated code.
- Break large tasks into smaller chunks and provide only relevant context.
- Never include secrets, credentials, tokens, or sensitive data in prompts.

- LLM-generated outputs must be validated by execution, testing, or reasoning through concrete examples before use.

5. Frontend Engineering Standards

- Proficiency with browser developer tools and systematic debugging techniques.
- Strong understanding of CSS fundamentals and layout behavior.
- Deep knowledge of JavaScript fundamentals including promises, hoisting, closures, and object behavior.
- Clear understanding of HTTP status codes, request lifecycles, and CORS behavior.
- Correct usage and understanding of browser storage: cookies, localStorage, sessionStorage, IndexedDB.
- React and React Native fundamentals: component lifecycle, state management, rendering behavior.
- Ability to explain why a component re-renders and how memoization (useCallback, useMemo) affects performance.
- Understanding of React context, complex state management patterns, and trade-offs.

6. Backend, Data & Systems Engineering

- Strong SQL fundamentals including table design, normalization, indexing, and query optimization.
- Understanding the differences and trade-offs between SQL and NoSQL databases.
- MongoDB fundamentals and appropriate use cases.
- Redis fundamentals, including caching strategies and consistency implications.
- Asynchronous processing concepts such as Kafka, event-driven systems, and AWS SQS.
- Clear understanding of async and concurrency models relevant to NodeJS, Python, Go, and Rust.
- Design systems with failure in mind: retries, idempotency, partial failures, and rollback strategies.

7. Cloud, DevOps & Delivery Standards

- Foundational understanding of networking concepts and basic DNS records.
- Clear understanding of request flow from DNS to load balancer to service to database.
- Ability to debug backend services in development and production environments.
- Test-driven development principles and appropriate test selection.
- Clear distinction between unit tests and end-to-end tests.
- Understanding and usage of GitHub Actions for CI/CD.

- Knowledge of deployment processes, environment separation, rollback strategies, and monitoring.

8. Communication, Security & Professional Conduct

- Reading and acknowledging messages in mandatory channels (scrum, general) is required.
- Missed communication is not an acceptable reason for delivery or coordination failures.
- Work must be backed up securely, including environment configurations and secrets.
- Company laptops, licensed software, and paid tools are strictly for professional use only.

This document defines the minimum bar for engineering at EchorTech. It is referenced during onboarding, day-to-day development, and quarterly check-ins.