Artificial Intelligence Technical Architect

Table of Contents

---

1. Technical Leadership

- **Cross-Team Collaboration** – Orchestrate smooth coordination between engineering, DevOps, QA, UI/UX, and product teams.
- **Stakeholder Communication** – Present architecture decisions and trade-offs to both technical and business leadership.

2. Pre-Sales & Solutioning

- **Client Requirement Gathering** – Client requirement gathering and understanding the client's needs.
- **Client Calls & Technical Discussions** – Join and lead client calls to clarify requirements, present solutions, and address technical queries.
- **Solution Proposal Creation** – Prepare architecture diagrams, high-level designs, and technology stack options.
- **Effort Estimation** – Deliver accurate estimates, including resourcing and timelines.

- **Risk & Feasibility** – Identify risks early and suggest mitigation paths.

- **Proof of Concepts (POC)** – Build rapid prototypes to validate feasibility and secure client buy-in.

- **Technical Presentations & Demos** – Showcase solutions with confidence to technical and non-technical audiences.

- **RFP/RFI Support** – Write technical sections for bids, detailing architecture, integrations, and delivery approaches.

3. Post-Sales & Solutioning

- **Pre-Development Analysis (PDA)** – Validate feasibility, constraints, and dependencies.

- **Design Gap Analysis** – Compare Figma/UI design against technical constraints and propose adjustments.

- **System & Data Architecture** – Create modular, service-oriented designs with proper data modeling.

- **Diagrams Design** – Create System diagrams, Architecture Diagrams, DFD, UML, ER Diagrams, deployment diagrams, etc.

- **Approach Documents** – Outline the delivery methodology, rationale for the tech stack, and execution plan.

- **API Documentation** – Define endpoints, request/response formats, authentication, and versioning.

- **Integration Blueprints** – Define how different systems, APIs, and services will interact.

- **Non-Functional Requirements Mapping** – Map performance, security, and scalability needs into design.

4. Technical Execution & Delivery

- **Task Breakdown & Estimation** – Translate functional requirements into granular development tasks.

- **Coding Standards** – Define and enforce style guides, linting, and clean code principles.

- **Security Guidelines** – Apply OWASP, encryption, and secure coding practices.

- **Performance Guidelines** – Define thresholds for response times, throughput, and concurrency to ensure optimal performance.

- **Code Reviews & Audits** – Provide deep architectural feedback and ensure adherence to standards.

- **Release Validation** – Ensure releases align with approved designs and business objectives.

5. Strong Coding Skills

- **Prototyping & POC Development** – Write clean, production-grade code for proof-of-concepts.

- **Complex Problem Debugging** – Jump into codebases to identify and fix deep technical issues.

- **Reference Implementations** – Build model codebases to set the standard for teams.

- **Code Optimization** – Refactor for performance, scalability, and maintainability.

- **Framework & Library Mastery** – Deep knowledge of core frameworks (ReactJS, Node.js, Bun.js, etc.).

- **Multi-Stack Capability** – Comfortable working across front-end, back-end, and cloud-native stacks.

- **Secure Coding Practices** – Implement and demonstrate patterns that avoid common vulnerabilities.

- **Code Mentoring** – Review and guide junior developers on writing clean, testable, and efficient code.

- **Automated Testing** – Write unit/integration tests as part of deliverables.

- **DevOps Tooling** – Familiarity with Docker, Kubernetes, CI/CD scripting.

6. Security & Compliance

- **Security Compliance** – Ensure GDPR, HIPAA, PCI-DSS, and ISO27001 compliance where applicable.

- **Vulnerability Management** – Oversee regular security scans and patch management.

7. Mentoring & Team Growth

- **Knowledge Sharing –** Keep easy-to-understand documentation, guides, and videos so everyone can learn quickly.

- **Onboarding Support –** Build simple onboarding guides and checklists to help new team members settle in fast and understand the system easily.

8. Continuous Learning & Innovation

- **Technology Watch** – Stay updated with emerging tools, frameworks, and industry trends.
- **Self-Learning** – Invest in certifications and R&D experiments.
- **Tool Evaluation** – Pilot and recommend tools that improve delivery efficiency.

9. Management & Operational Responsibilities

- **Resource Planning** – Allocate and optimize team resources for efficiency.
- **Status Reporting** – Deliver transparent updates to stakeholders on risks, dependencies, and progress.
- **Budget Tracking** – Monitor costs vs. estimates throughout the project lifecycle.

10. Accountability & Ownership

- **End-to-End Delivery Ownership** – From concept to go-live, ensure solution delivery meets agreed goals.
- **Post-Go-Live Support** – Oversee optimizations, bug fixes, and incident resolution.
- **Business Alignment** – Ensure every technical decision drives measurable business value.