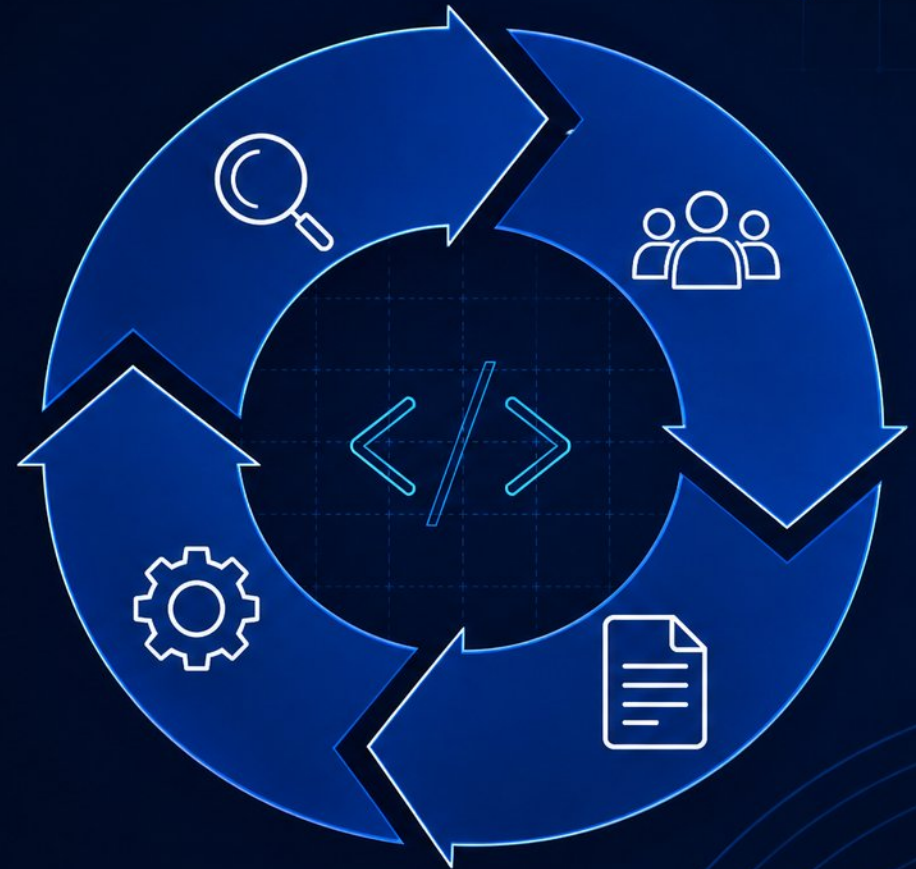


Web Development Process:

Stages, Roles, and Deliverables

- - Full lifecycle from discovery to ongoing maintenance
- - Structured process prevents scope creep and rework
- - Defines stages, roles, and deliverables for every project
- - Critical for developers, learners, and technical teams



Why Process Discipline Matters



- Most web project failures are process failures, not technical ones



- Unclear scope, weak decision ownership, and late content are top killers



- Process discipline improves cost control, timeline predictability, and maintainability



The End-to-End Lifecycle

- Seven core stages: discovery, planning, design, development, testing, deployment, maintenance
- Phases overlap, loop back, and run in parallel in modern iterative workflows
- Phase-gate checkpoints require formal sign-off before work advances
- Each stage produces specific deliverables and requires stakeholder review



Discovery and Requirements



- Define business goals, target users, and success metrics



- Deliverables: project brief, user personas, requirements document, prioritized backlog



- Rushing discovery causes scope creep and rework downstream



- Document technical constraints and integration requirements early



Planning and Architecture



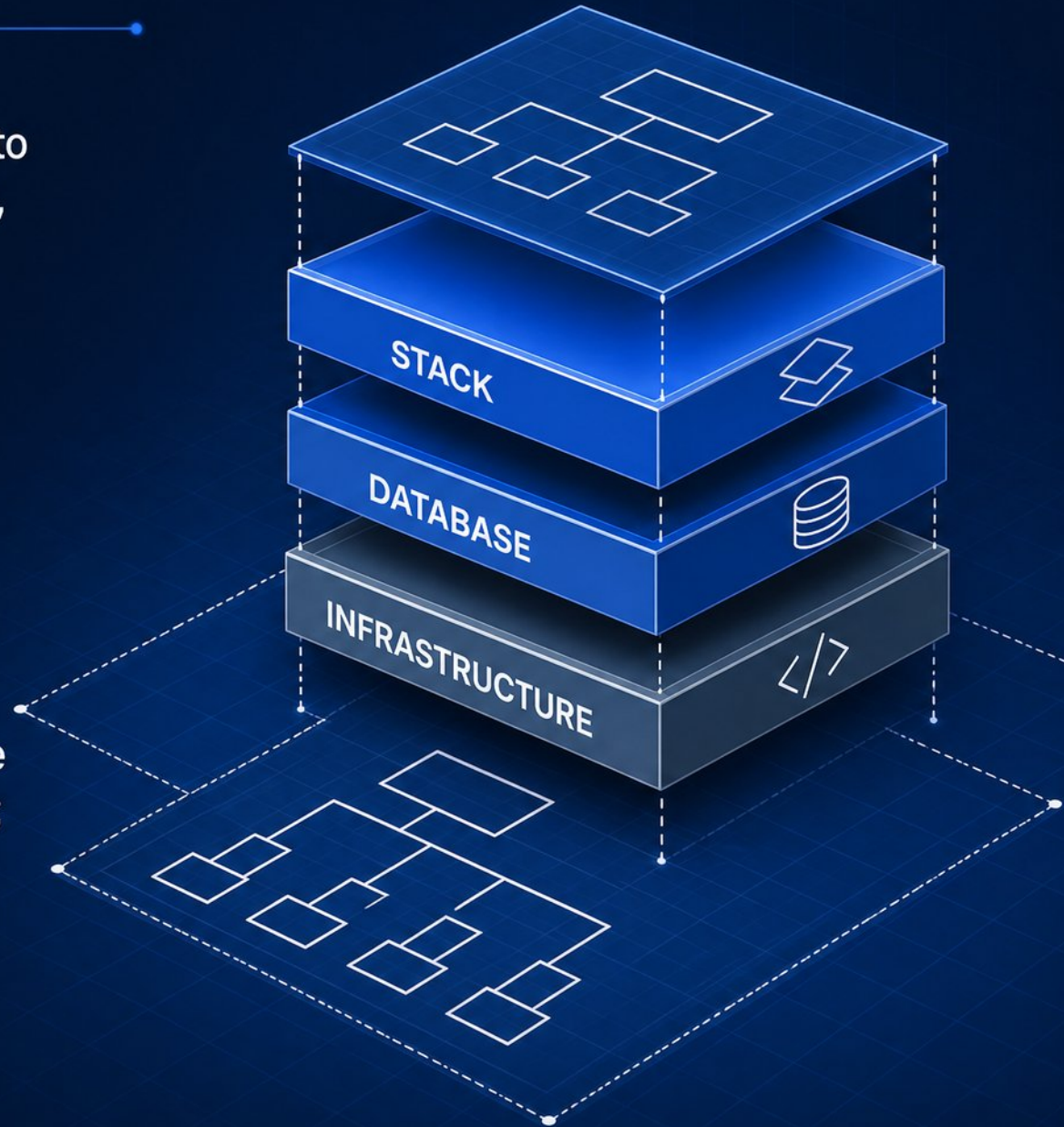
- Translate requirements into technical decisions: stack, database, API strategy, infrastructure



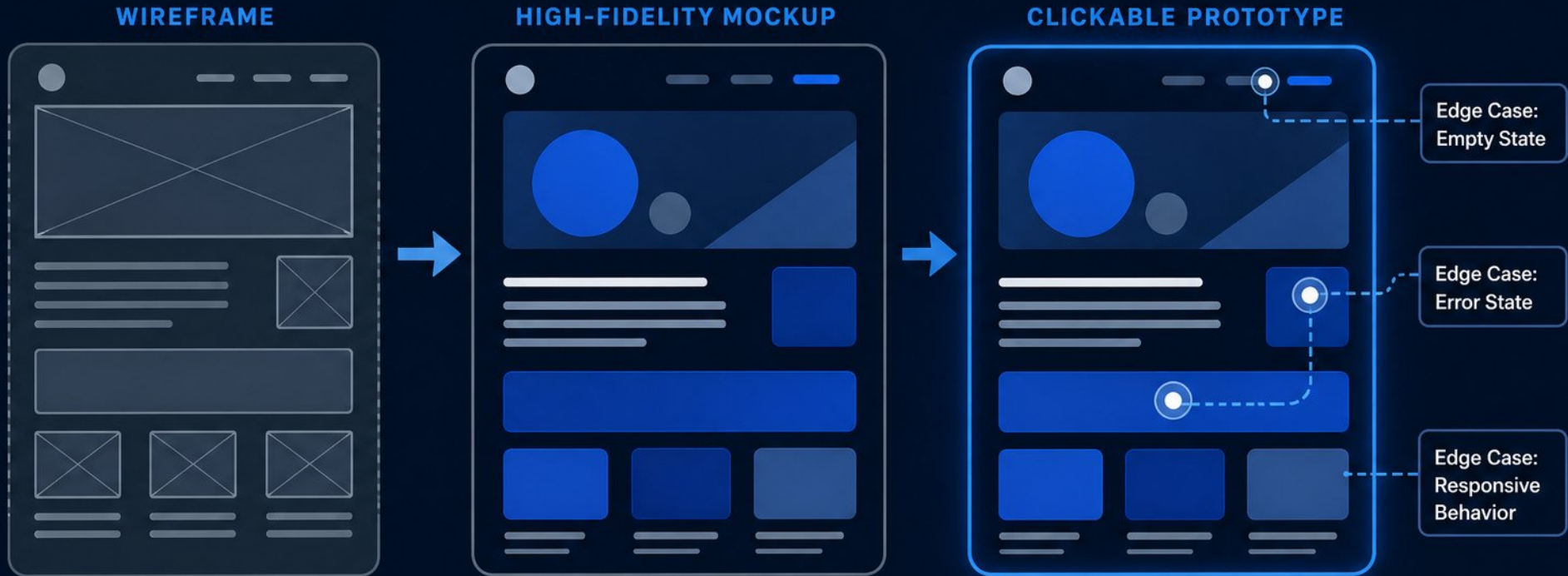
- Deliverables: sitemap, architecture document, ADRs, project timeline and milestones



- Architecture decisions are expensive to reverse later; make them early and deliberately



UX/UI Design and Prototyping



- From wireframes to high-fidelity mockups and clickable prototypes



- Deliverables: design system, component library, interactive prototype, style guide

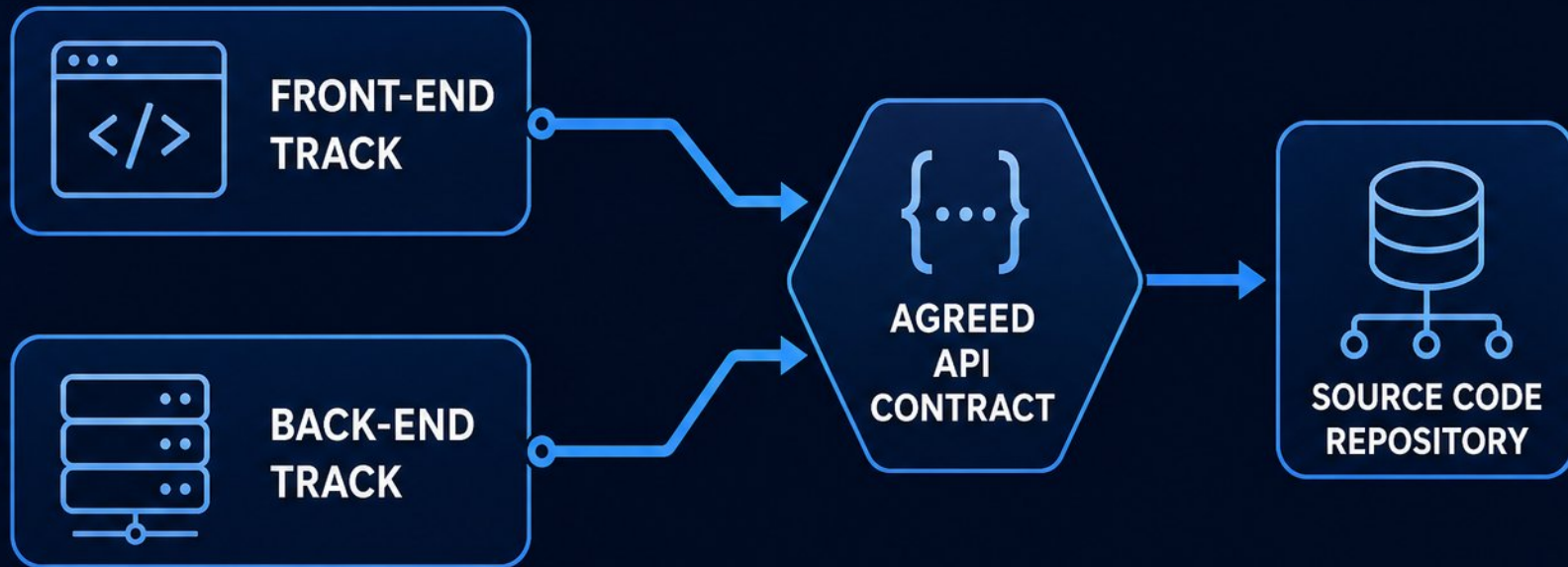


- Specify all states, edge cases, and responsive behavior in the design phase



- Approval gates before development to prevent late-stage rework

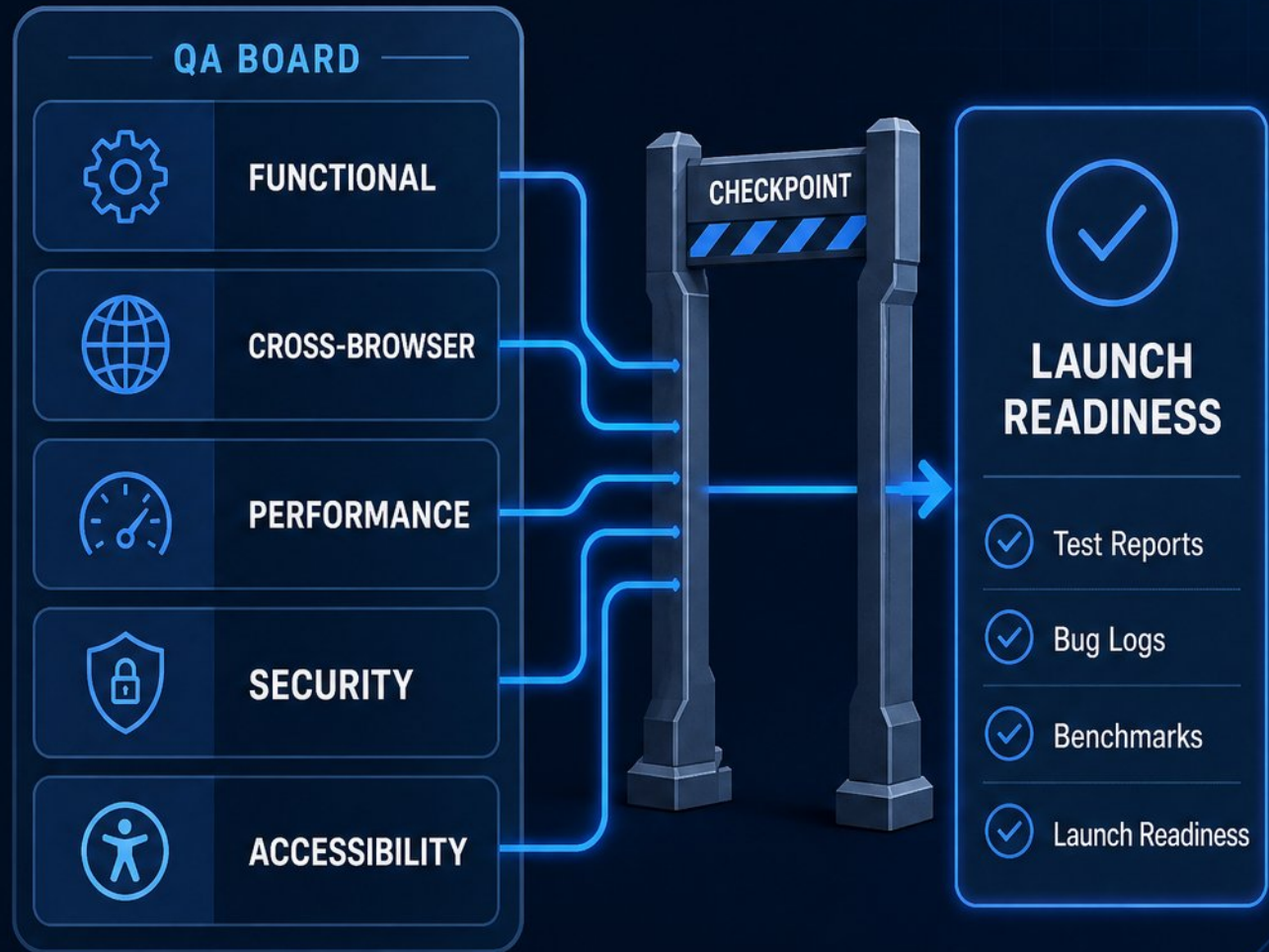
Development and Integration



- - Front-end and back-end tracks run in parallel against agreed API contracts
- - Deliverables: working software increments, source code repository, API documentation
- - CI/CD pipelines, code review, and automated tests are baseline practices

Testing and Quality Assurance

- ✓ QA runs alongside development, not as a final checkpoint
- ✓ Required: functional, cross-browser, performance, security, accessibility
- ✓ Test on staging, never on production
- ✓ Core Web Vitals: LCP < 2.5s, INP < 200ms, CLS < 0.1
- ✓ Deliverables: test reports, bug logs, benchmarks, launch readiness



Deployment and Launch

- - Launch is a checklist-driven sequence, not a single action
- - Essential items: DNS, SSL, redirects, analytics, sitemap, monitoring, rollback plan
- - Quality gates before launch include Core Web Vitals, security headers, and accessibility checks

	DNS	
	SSL	
	Redirects	
	Analytics	
	Sitemap	
	Monitoring	
	Rollback plan	

CHECKPOINT
GATE

Post-Launch Maintenance and Iteration

- Launch is the starting line, not the finish line
- Maintenance spans seven decaying layers: security, dependencies, performance, content, accessibility, infrastructure, SEO
- Monitoring and user data feed back into discovery for continuous improvement
- Use synthetic checks and real-user monitoring to catch regressions early
- Post-launch feedback loops drive the next iteration cycle



Key Roles and Ownership

- Seven core roles: Product Owner, PM, Architect, Designer, Developer, QA, DevOps
- Each role owns decisions, not just task lists
- Exactly one Accountable person per task
- RACI: Responsible, Accountable, Consulted, Informed
- Small teams combine roles without losing clarity



 Accountable Decisions	 Product Owner	 Project Manager	 Solution Architect	 Designer	 Developer	 QA	 DevOps
 What are we building and why?	A	C	C	C	C	C	I
 How will we deliver it?	C	A	C	C	C	C	C
 How should it be designed?	C	C	A	R	C	C	I
 How is it built and tested?	I	C	C	I	A	R	C
 How is it released and operated?	I	C	C	I	C	C	A

R Responsible

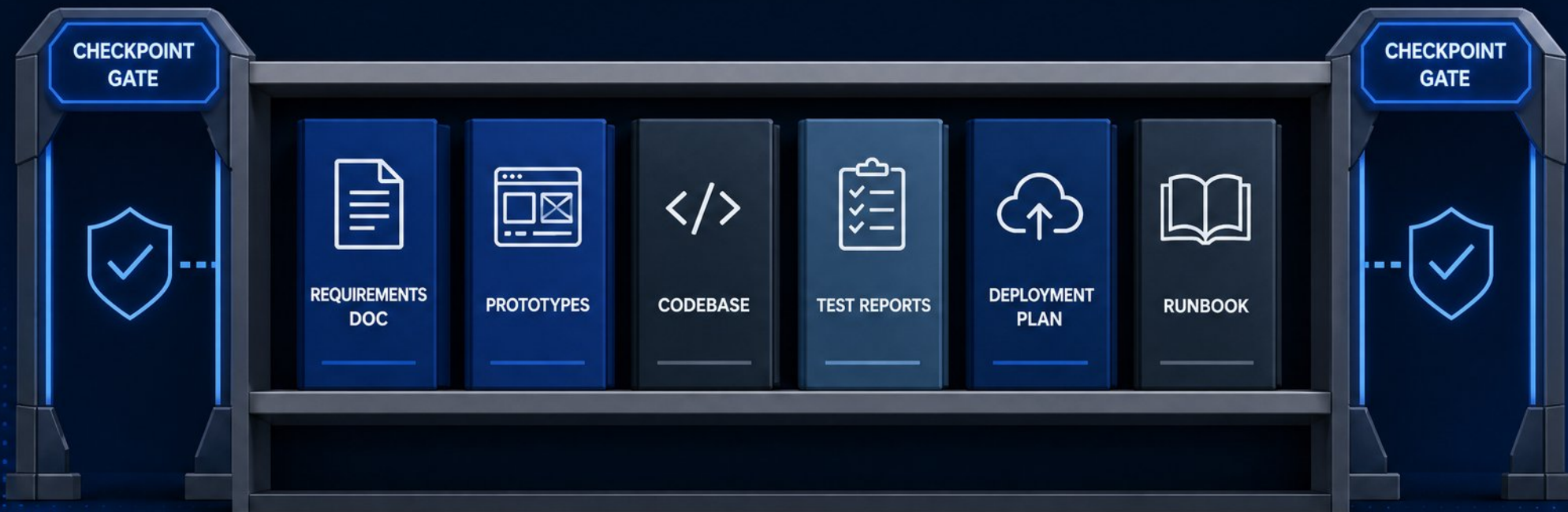
A Accountable

C Consulted

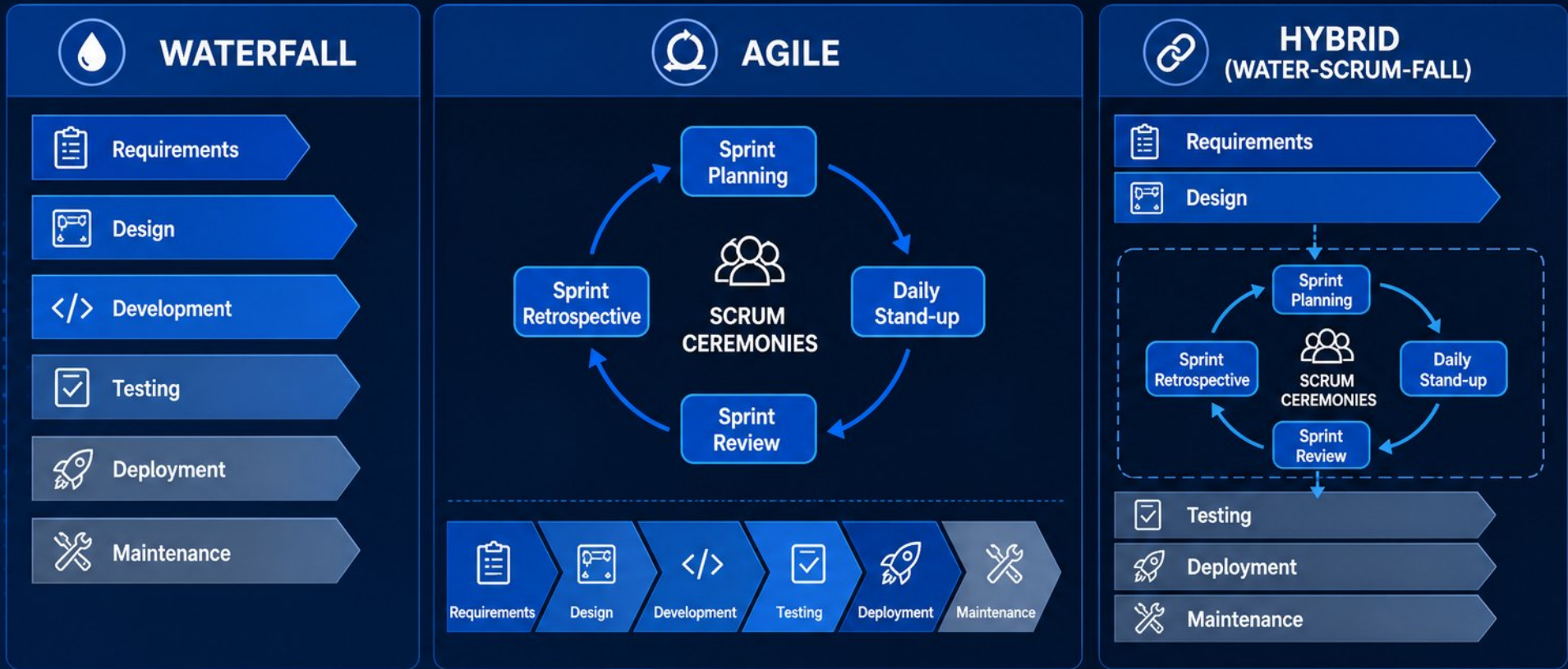
I Informed

Deliverables by Stage

- Every stage produces specific, reviewable, formally accepted outputs
- Examples: requirements doc, prototypes, codebase, test reports, deployment plan, runbook
- Missing deliverables signal incomplete work; deliverables are client assets, not agency property



Process Models: Waterfall, Agile, Hybrid



- Waterfall suits stable, well-defined requirements and fixed budgets
- Agile fits evolving scope with iterative feedback and client involvement
- Scrum ceremonies map to web development stages; Kanban fits continuous maintenance work
- Hybrid models (Water-Scrum-Fall) combine waterfall planning with agile execution

From Requirements to Technical Implementation



- ✓ – Break business goals into user stories, acceptance criteria, and technical tasks
- ✓ – ADRs record context, decision, and rationale for key architecture choices
- ✓ – ADRs are immutable once accepted; new decisions supersede old ones
- ✓ – Definition of done and handoff criteria clarify role ownership and prevent ambiguity

Collaboration and Handoffs

- Handoff is a contract, not a picture; use tokens, states, and specs
- Review gates and mid-sprint design reviews reduce miscommunication
- Structured tools capture CSS values, viewports, and browser metadata
- Synced design files and codebases prevent design debt and drift



Practical Takeaways



- Assign clear ownership for every task and deliverable



- Define done, acceptance criteria, and written sign-off at each phase gate



- Budget 15–25% of build cost annually for ongoing maintenance



- Plan for content, decisions, and post-launch operations before kickoff



- Treat launch as the starting line, not the finish line

PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/c6a2f574/public