AI CODING
IMPLEMENTATION CHECKLIST

www.strategyradar.ai

Before Using AI Code Generation

- [ ] **Created Business Context document**
  WHY we build: business priorities, strategic objectives, KPIs, trade-off guidelines
  template · *pack*
- [ ] **Created Functional Context document**
  WHAT it should do: actors, flows, entities, business rules, edge cases · *template* · *pack*
- [ ] **Created Technical Strategy document**
  HOW to build: tech stack, architecture patterns, coding standards, testing requirements · *template* · *pack*
- [ ] **Created Technical Context document**
  WHERE code goes: project structure, file naming, key reference files, integration points · *template* · *pack*
- [ ] **Set up AGENTS.md or CLAUDE.md**
  Entry point that references your four context documents for automatic AI loading
  template · *pack*
- [ ] **Established code review process**
  Define review criteria, responsibilities, and what to look for in AI-generated code
- [ ] **Trained team on structured AI workflows**
  Everyone understands context engineering and specs-driven approach (and avoids vibe coding)
  webinar
- [ ] **Prepared specialized agents (optional)**
  Configure agents for common tasks or team consistency (e.g., backend, frontend, docs) · *guide*

For Each Development Task

- [ ] **Described the changes**
  Keep it simple for straightforward tasks, add detail for complex features
  guide · *template*
- [ ] **Referenced relevant context documents**
  Point AI to existing Business, Functional, Technical Strategy, and Technical Context as needed
  guide · *template* · *example*
- [ ] **Defined acceptance criteria**
  Clear, measurable definition of "done" (leverage techniques like BDD)
  guide · *template* · *example*
- [ ] **Specified quality requirements**
  Non-functional requirements: performance, security, accessibility, etc.
  guide · *template*
- [ ] **Identified edge cases and constraints**
  Special cases, boundary conditions, and limitations
  guide · *template*
- [ ] **Added tech specs (for complex changes)**
  High-level or low-level design as needed, leverage diagrams as code and reference master specs
  guide · *template* · *example*

After AI Coding Generation

- [ ] **Reviewed for pattern consistency**
  Matches established architectural patterns and coding conventions from Technical Strategy
- [ ] **Checked for technical debt signals**
  Verbose code, scope creep (features not requested), unnecessary complexity
- [ ] **Validated against task specification**
  Meets all acceptance criteria without gold-plating or extra features
- [ ] **Tested edge cases explicitly**
  Not just happy path, verify error handling and boundary conditions
- [ ] **Updated context documents if needed**
  Document architectural changes, new patterns, or updated business rules in relevant context files
- [ ] **Verified security considerations**
  No injection vulnerabilities, proper input validation, secure dependencies

Ongoing Maintenance

- [ ] **Review and update context documents**
  Keep architecture documentation current as system evolves
- [ ] **Measure productivity impact**
  Track actual results vs expectations (velocity, code quality, bug rates)
- [ ] **Monitor technical debt accumulation**
  Regular code quality assessments, watch for code bloat trends
- [ ] **Refine AI guidelines based on learnings**
  Continuous improvement, update AGENTS.md with better patterns as you discover them