Cursor Best Practices Integration Guide

Overview

This document outlines the integration of 10 best practices from “Mastering Cursor IDE” by Roberto Infante into our existing MDC repository structure.

Integrated Best Practices

1. Product Requirements Document (PRD) First

Integration: Added to `05-cursor-best-practices.mdc`
- **What**: Always start with comprehensive PRD generation
- **Why**: Provides “North Star” for development, aligns AI with goals
- **How**: Use Cursor Agent with structured prompts to generate PRD
- **Files**: Save as `instructions.md` or `PRD.md` for @File references

2. Agent Mode Selection Strategy

Integration: Added to `05-cursor-best-practices.mdc`
- **AGENT Mode**: For autonomous execution (implementing, refactoring, testing)
- **ASK Mode**: For consultation and planning (read-only, no modifications)
- **Decision Matrix**: “Do this” → AGENT, “Tell me about” → ASK

3. Model Selection Guidelines

Integration: Added to `05-cursor-best-practices.mdc`
- **Top-tier models**: Claude-4 Sonnet, OpenAI o3, Gemini 2.5 Pro
- **Context considerations**: Match model to project size and complexity
- **Cost optimization**: Use appropriate model for task complexity

4. @ References Mastery

Integration: Enhanced in `05-cursor-best-practices.mdc` and `25-context-optimization.mdc`
- **@File/@Files**: Include file contents in prompts
- **@Code**: Reference specific code snippets or symbols
- **@Web**: Pull real-time information from web
- **@Terminal**: Include runtime output and error logs
- **@Git**: Reference version history and commits

5. Detailed Prompt Engineering

Integration: New dedicated file `15-prompt-engineering.mdc`
- **OSCAR Framework**: Objective-Specification-Context-Acceptance-References
- **Prompt patterns**: Feature implementation, bug fixing, refactoring
- **Quality metrics**: Clarity, context completeness, specificity scores
- **Anti-patterns**: Avoid vague requests, context overload, assumption gaps

6. Quality Triad: Logging + Tests + Documentation

Integration: Enhanced existing `30-testing.mdc` and added to `05-cursor-best-practices.mdc`
- **Always request logging**: Include observability in every feature

- **Generate tests proactively**: Unit tests before moving forward
- **Documentation as first-class**: README, docstrings, API docs
- **Definition of Done**: Feature + Tests + Logs + Docs

7. Iterative Improvement Cycle

Integration: Added to `15-prompt-engineering.mdc`
- **3-Pass Method**: Structure/Logic → Quality/Standards → Polish/Documentation
- **Feedback loops**: Immediate feedback, structured review, refinement prompts
- **Progressive disclosure**: Start simple, add complexity iteratively

8. Smart Indexing with Ignore Files

Integration: New dedicated file `25-context-optimization.mdc`
- **Enhanced .cursorignore**: Complete exclusion patterns for various project types
- **Strategic .cursorindexignore**: On-demand access to documentation and legacy code
- **Performance optimization**: Token management and response time improvement

9. Context Management Strategies

Integration: Added to `25-context-optimization.mdc`
- **Progressive context building**: Layer context from high-level to specific
- **Reference optimization**: Prefer specific over general references
- **Token budget allocation**: Strategic distribution of context budget

10. Advanced Features (MCP Servers)

Integration: Added to `05-cursor-best-practices.mdc`
- **When to use**: Large projects, domain-specific knowledge, custom tools
- **Popular servers**: Context7, DeepWiki, framework-specific MCPs
- **Setup considerations**: Advanced feature for complex projects

File Structure Changes

New Files Created

1. `05-cursor-best-practices.mdc` - Core Cursor IDE best practices
2. `15-prompt-engineering.mdc` - Advanced prompt engineering techniques
3. `25-context-optimization.mdc` - Context and indexing optimization

Enhanced Existing Files

- Updated `.cursorignore` with comprehensive patterns
- Enhanced `30-testing.mdc` with quality triad concepts
- Improved `README.md` with best practices integration

Implementation Workflow

Daily Development Flow (Enhanced)

1. **Start**: Review PRD and current state (ASK mode)
2. **Plan**: Break down tasks, choose appropriate model
3. **Context**: Set up @references and optimize ignore files
4. **Implement**: Use AGENT mode with detailed, structured prompts
5. **Quality**: Request logging, tests, and documentation

6. **Iterate**: Review, refine, and improve using 3-pass method
7. **Document**: Update PRD, changelog, and project documentation

Project Setup Checklist (New)

- [ ] PRD created and saved as `instructions.md`
- [ ] Project rules configured (relevant `.mdc` files)
- [ ] `.cursorignore` and `.cursorindexignore` optimized
- [ ] Model selection strategy defined
- [ ] @ reference patterns established
- [ ] Quality standards documented (logging, testing, docs)

Integration Benefits

Improved Code Quality

- **Systematic approach**: PRD-first development ensures alignment
- **Quality triad**: Every feature includes tests, logs, and documentation
- **Iterative refinement**: 3-pass method ensures polished output

Enhanced Productivity

- **Smart context management**: Optimized token usage and response times
- **Effective prompting**: OSCAR framework and pattern library
- **Appropriate tool usage**: Right model and mode for each task

Better Maintainability

- **Comprehensive documentation**: PRD, README, API docs, code comments
- **Test coverage**: Proactive test generation with high coverage targets
- **Observability**: Logging standards for debugging and monitoring

Migration Guide

For Existing Projects

1. **Audit current ignore files**: Update with new comprehensive patterns
2. **Create PRD**: Generate Product Requirements Document for existing projects
3. **Review prompt patterns**: Adopt OSCAR framework for complex tasks
4. **Enhance quality practices**: Ensure logging, testing, and documentation standards

For New Projects

1. **Start with PRD**: Use Cursor Agent to generate comprehensive requirements
2. **Set up context optimization**: Configure ignore files from project start
3. **Apply prompt engineering**: Use structured prompts and reference patterns
4. **Implement quality triad**: Include logging, tests, and docs from day one

Compatibility Notes

- **Cursor Rules v2**: All new rules follow `.mdc` format with proper front-matter
- **Existing rules**: Enhanced without breaking changes to current structure
- **Cross-references**: New rules properly reference existing rules using `@ref:` syntax

- **Modular design**: New practices can be adopted incrementally

Success Metrics

Quality Indicators

- **First-pass accuracy**: >90% of AI responses require minimal iteration
- **Test coverage**: ≥80% coverage on modified modules
- **Documentation completeness**: All features have PRD, README, and API docs

Efficiency Indicators

- **Token utilization**: >80% relevance in referenced content
- **Response time**: Optimized context leads to faster AI responses
- **Development velocity**: Reduced iteration cycles through better prompting

Maintainability Indicators

- **Code consistency**: Adherence to established patterns and standards
- **Debugging efficiency**: Comprehensive logging enables faster issue resolution
- **Knowledge transfer**: Complete documentation supports team onboarding