6-Month Software Engineering Roadmap

Simple Daily Plan with Time

Month	Day	Time	Task
Month 1	Day 1	**30 min**	Answer 3 questions before coding: What problem? What solutions? Best approach?
	Day 2	**2-3 hrs**	Weekend: Read Clean Code Chapter 1. Refactor old code with better names
	Day 3	**2-3 hrs**	Weekend: Read Clean Code Chapter 2. Break large function into smaller ones
	Day 4	**30 min**	Think Before Code exercise #2. Document problem and solutions
	Day 5	**30 min**	Think Before Code exercise #3. Focus on SIMPLEST solution
	Day 6	**30 min**	Think Before Code exercise #4. Ask: Will this be clear in 6 months?
	Day 7	**30 min**	Think Before Code exercise #5. Explain code choice to teammate
	Day 8	**30 min**	Think Before Code exercise #6. Weekly review: Rate code quality (1-10)
	Day 9	**2-3 hrs**	Weekend: Read Clean Code Chapter 3. Improve code comments
	Day 10	**2-3 hrs**	Weekend: Read Clean Code Chapter 4. Refactor class structure
	Day 11	**30 min**	Think Before Code #7. Draw flowchart before coding
	Day 12	**30 min**	Think Before Code #8. Focus on edge cases in planning
	Day 13	**30 min**	Think Before Code #9. Think about system-wide impact
	Day 14	**30 min**	Think Before Code #10. What could break? How to prevent it?
	Day 15	**30 min**	Think Before Code #11. Weekly review
	Day 16	**2-3 hrs**	Weekend: Review all exercises. Identify thinking patterns
	Day 17	**2-3 hrs**	Weekend: Major refactoring - apply Clean Code principles
	Day 18	**30 min**	Think Before Code #12. Focus on naming variables/functions
	Day 19	**30 min**	Think Before Code #13. Break complex problems into sub-problems
	Day 20	**30 min**	Think Before Code #14. Think about testing before coding
	Day 21	**30 min**	Think Before Code #15. Consider alternative approaches
	Day 22	**30 min**	Think Before Code #16. Weekly review
	Day 23	**2-3 hrs**	Weekend: Re-read Clean Code Ch 1-2. Fix 3 violations
	Day 24	**2-3 hrs**	Weekend: Refactor large function (20+ lines)
	Day 25	**30 min**	Think Before Code #17. Document why NOT using certain solutions
	Day 26	**30 min**	Think Before Code #18. Think about maintainability
	Day 27	**30 min**	Think Before Code #19. Consider performance implications

	Day 28	**30 min**	Think Before Code #20. Is there a simpler way?
	Day 29	**30 min**	Think Before Code #21. Monthly review
Month 2	Day 30	**2-3 hrs**	Weekend: Watch Laracasts SRP. Find God class and split it
	Day 31	**2-3 hrs**	Weekend: Read about SRP. Refactor with single responsibilities
	Day 32	**30 min**	Learn SRP: Find code where class has multiple reasons to change
	Day 33	**30 min**	Apply SRP: Does each class have ONE clear purpose?
	Day 34	**30 min**	SRP Practice: Identify God class and plan to split
	Day 35	**30 min**	SRP: Practice explaining in simple terms
	Day 36	**30 min**	SRP Review: Fix ONE violation. Weekly review
	Day 37	**2-3 hrs**	Weekend: Watch Laracasts OCP. Refactor for extension
	Day 38	**2-3 hrs**	Weekend: Read OCP. Implement with interfaces
	Day 39	**30 min**	Learn OCP: Study code with if/switch statements
	Day 40	**30 min**	Apply OCP: Make code extensible
	Day 41	**30 min**	OCP Practice: Use interfaces for extensibility
	Day 42	**30 min**	OCP: Study plugin architectures
	Day 43	**30 min**	OCP Review: Refactor one feature. Weekly review
	Day 44	**2-3 hrs**	Weekend: Watch Laracasts LSP. Study inheritance
	Day 45	**2-3 hrs**	Weekend: Read LSP violations. Fix broken inheritance
	Day 46	**30 min**	Learn LSP: When does inheritance break?
	Day 47	**30 min**	Apply LSP: Can child classes replace parents?
	Day 48	**30 min**	LSP Practice: Prefer composition over inheritance
	Day 49	**30 min**	LSP: Learn contract design
	Day 50	**30 min**	LSP Review: Fix one issue. Weekly review
	Day 51	**2-3 hrs**	Weekend: Watch Laracasts ISP and DIP. Use interfaces
	Day 52	**2-3 hrs**	Weekend: Study ISP and DIP. Implement dependency injection
	Day 53	**30 min**	Learn ISP: Find fat interfaces. Split them
	Day 54	**30 min**	Apply ISP: Break down large interfaces
	Day 55	**30 min**	Learn DIP: High-level shouldn't depend on low-level
	Day 56	**30 min**	Apply DIP: Depend on abstractions
	Day 57	**30 min**	Month 2 Review: Explain all 5 SOLID principles
Month 3	Day 58	**2-3 hrs**	Weekend: Read Repository Pattern. Implement for one model
	Day 59	**2-3 hrs**	Weekend: Study Repository. Separate data from business logic

	Day 60	**30 min**	Repository: Study interface/implementation separation
	Day 61	**30 min**	Repository: Find direct Eloquent usage. Abstract it
	Day 62	**30 min**	Repository: Create interface and implementation
	Day 63	**30 min**	Repository: Move data access logic to repository
	Day 64	**30 min**	Repository: Test. Can you swap implementations? Weekly review
	Day 65	**2-3 hrs**	Weekend: Read Strategy Pattern. Implement for algorithms
	Day 66	**2-3 hrs**	Weekend: Study Strategy in Laravel (payment gateways)
	Day 67	**30 min**	Strategy: Find if/else chains for algorithms
	Day 68	**30 min**	Strategy: Design strategy interface
	Day 69	**30 min**	Strategy: Implement 2-3 concrete strategies
	Day 70	**30 min**	Strategy: Use context class to switch strategies
	Day 71	**30 min**	Strategy: Test all strategies. Weekly review
	Day 72	**2-3 hrs**	Weekend: Read Observer Pattern. Study Laravel events
	Day 73	**2-3 hrs**	Weekend: Implement Observer using Laravel events
	Day 74	**30 min**	Observer: Identify tight coupling needing events
	Day 75	**30 min**	Observer: Design events for business process
	Day 76	**30 min**	Observer: Create event and listener classes
	Day 77	**30 min**	Observer: Wire up observers and test
	Day 78	**30 min**	Observer: Review decoupling. Weekly review
	Day 79	**2-3 hrs**	Weekend: Read Factory Pattern. Implement for complex creation
	Day 80	**2-3 hrs**	Weekend: Study Factory variations. Implement one
	Day 81	**30 min**	Factory: Find complex object creation in code
	Day 82	**30 min**	Factory: Design factory class
	Day 83	**30 min**	Factory: Implement factory methods
	Day 84	**30 min**	Factory: Test factory flexibility
	Day 85	**30 min**	Month 3 Review: Explain all 4 patterns
Month 4	Day 86	**2-3 hrs**	Weekend: Read Clean Architecture. Draw project diagram
	Day 87	**2-3 hrs**	Weekend: Study layered architecture. Identify layers
	Day 88	**40 min**	Before features: Draw diagram - components, communication, data
	Day 89	**40 min**	Architecture: How would this work with 100x users?
	Day 90	**40 min**	Study: Domain, Application, Infrastructure layers
	Day 91	**40 min**	Practice: Draw architecture before implementing

	Day 92	**40 min**	Review: Did diagram help? Weekly review
	Day 93	**2-3 hrs**	Weekend: Read dependency rules. Diagram dependencies
	Day 94	**2-3 hrs**	Weekend: Identify architectural violations. Plan fixes
	Day 95	**40 min**	Study: What crosses layer boundaries?
	Day 96	**40 min**	Architecture: Is business logic framework-independent?
	Day 97	**40 min**	Practice: Move business logic from controllers
	Day 98	**40 min**	Diagram: Map data flow from request to response
	Day 99	**40 min**	Review: Is architecture clearer? Weekly review
	Day 100	**2-3 hrs**	Weekend: Study hexagonal architecture. Draw it
	Day 101	**2-3 hrs**	Weekend: Read screaming architecture. Check folder structure
	Day 102	**40 min**	Before coding: What components? How interact?
	Day 103	**40 min**	Think: What if switched from MySQL to MongoDB?
	Day 104	**40 min**	Practice: Create clear boundaries
	Day 105	**40 min**	Architecture: Design with dependency inversion
	Day 106	**40 min**	Review: Compare before/after. Weekly review
	Day 107	**2-3 hrs**	Weekend: Read microservices vs monoliths
	Day 108	**2-3 hrs**	Weekend: Study scalability patterns
	Day 109	**40 min**	Scalability: Handle 10x traffic? What breaks?
	Day 110	**40 min**	Architecture: Identify single points of failure
	Day 111	**40 min**	Study: Horizontal vs vertical scaling
	Day 112	**40 min**	Practice: Design with statelessness
	Day 113	**40 min**	Month 4 Review: Draw and explain architecture
Month 5	Day 114	**2-3 hrs**	Weekend: Study TDD basics. Write tests for new feature
	Day 115	**2-3 hrs**	Weekend: Practice Red-Green-Refactor cycle
	Day 116	**40 min**	TDD: Write failing test FIRST. Make it pass
	Day 117	**40 min**	TDD: Focus on test design. What behavior?
	Day 118	**40 min**	TDD: Write small focused tests
	Day 119	**40 min**	TDD: Refactor after green
	Day 120	**40 min**	Review: How does TDD change design? Weekly review
	Day 121	**2-3 hrs**	Weekend: Set up PHPStan level 5+. Fix errors
	Day 122	**2-3 hrs**	Weekend: Set up PHP CS Fixer. Auto-format
	Day 123	**40 min**	PHPStan: Run before every commit

	Day 124	**40 min**	Static analysis: Add type hints to all methods
	Day 125	**40 min**	Quality: Use PHPStan to catch bugs
	Day 126	**40 min**	Practice: Write code passing highest PHPStan level
	Day 127	**40 min**	Review: Bugs caught by static analysis? Weekly review
	Day 128	**2-3 hrs**	Weekend: Add tests to 0% coverage code. Aim for 80%
	Day 129	**2-3 hrs**	Weekend: Practice integration tests end-to-end
	Day 130	**40 min**	Testing: Write unit tests for business logic
	Day 131	**40 min**	Testing: Write feature tests for API endpoints
	Day 132	**40 min**	Testing: Use factories for test data
	Day 133	**40 min**	Testing: Practice mocking external services
	Day 134	**40 min**	Review: Check coverage. What's untested? Weekly review
	Day 135	**2-3 hrs**	Weekend: Study testing best practices. Refactor tests
	Day 136	**2-3 hrs**	Weekend: Set up CI pipeline. Auto-run tests
	Day 137	**40 min**	Quality: Tests as documentation
	Day 138	**40 min**	Practice: Test edge cases and error paths
	Day 139	**40 min**	Testing: Separate unit, integration, feature tests
	Day 140	**40 min**	Quality: Refactor based on test feedback
	Day 141	**40 min**	Month 5 Review: Compare quality Month 1 vs now
Month 6	Day 142	**2-3 hrs**	Weekend: Profile app. Find slow endpoints
	Day 143	**2-3 hrs**	Weekend: Study query optimization. Fix N+1 queries
	Day 144	**40 min**	Performance: Enable query logging. Find slow queries
	Day 145	**40 min**	Database: Add indexes to queried columns
	Day 146	**40 min**	Optimize: Fix N+1 with eager loading
	Day 147	**40 min**	Database: Use select() for needed columns only
	Day 148	**40 min**	Review: Measure query improvements. Weekly review
	Day 149	**2-3 hrs**	Weekend: Study Redis caching. Implement for queries
	Day 150	**2-3 hrs**	Weekend: Practice cache strategies. Implement
	Day 151	**40 min**	Caching: Identify expensive operations
	Day 152	**40 min**	Cache: Implement Redis for frequent data
	Day 153	**40 min**	Cache: Set TTL. Plan invalidation strategy
	Day 154	**40 min**	Practice: Cache aggregations, reports, API responses
	Day 155	**40 min**	Review: Measure response time. Weekly review

	Day 156	**2-3 hrs**	Weekend: Study Laravel queues. Set up worker
	Day 157	**2-3 hrs**	Weekend: Move slow tasks to queues (emails, reports)
	Day 158	**40 min**	Queues: Identify tasks that should be async
	Day 159	**40 min**	Async: Move email sending to queues
	Day 160	**40 min**	Queues: Implement job for slow reports
	Day 161	**40 min**	Practice: Use job batching for bulk operations
	Day 162	**40 min**	Review: How much faster? Weekly review
	Day 163	**2-3 hrs**	Weekend: Study horizontal vs vertical scaling
	Day 164	**2-3 hrs**	Weekend: Write scaling plan for 10x traffic
	Day 165	**40 min**	Scalability: Ensure app is stateless
	Day 166	**40 min**	Scale: What if multiple app servers?
	Day 167	**40 min**	Performance: Use pagination for large datasets
	Day 168	**40 min**	Optimization: Use chunk() for bulk operations
	Day 169	**40 min**	Final Review: Performance before/after Month 6
	Day 170	**2-3 hrs**	Weekend: Review entire 6-month journey
	Day 171	**2-3 hrs**	Weekend: Compare Month 1 vs Month 6 code
	Day 172	**40 min**	Reflection: What are strongest improvements?
	Day 173	**40 min**	Reflection: What needs more practice?
	Day 174	**40 min**	Achievement: List patterns you use naturally
	Day 175	**40 min**	Planning: Create 'next 6 months' plan
	Day 176	**40 min**	Celebration: You're a Software Engineer now!
	Day 177	**2-3 hrs**	Weekend: Share knowledge - write blog post
	Day 178	**2-3 hrs**	Weekend: Mentor someone - explain patterns
	Day 179	**40 min**	Keep habit: Continue 'Think Before Code' daily
	Day 180	**40 min**	Keep growing: Start advanced topics (DDD, CQRS)