Build-Ship-Iterate: AI-Augmented Product Development Methodology

Overview

Build-Ship-Iterate is a modern methodology for transforming product ideas into operational digital solutions through AI-augmented development. It represents the evolution of software development where strategic thinking and architectural decisions take precedence over manual code writing.

This approach leverages AI tools for implementation while maintaining full control over architecture, business logic, and quality—enabling superior productivity without sacrificing understanding or ownership of the final product.

Core Philosophy

The Evolution of Development

Modern product development isn't about writing every line of code—it's about making the right decisions and ensuring quality outcomes. Just as architects use CAD instead of hand-drawing, I use AI tools to implement my technical vision more efficiently.

Three Pillars

1. Architect with Intelligence

Focus on system design, data flow, and user experience. The architecture and business logic are where human insight adds the most value.

2. Build with AI Precision

Leverage Claude Code for rapid, consistent implementation while maintaining complete oversight of the generated solution.

3. Validate with Rigor

Test thoroughly, monitor actively, and iterate based on real usage. No amount of AI can replace human judgment in quality assurance.

The Build-Ship-Iterate Framework

My Role as Product Builder

What I Control (100%)

- System architecture and technical stack decisions
- Database schema design and relationships
- API structure and data flow
- State management patterns (Pinia stores, composables)
- User experience and interface flow
- Business logic and validation rules

- Deployment strategy and infrastructure

What AI Handles (Under My Direction)

- Code syntax and boilerplate generation
- Implementation of defined patterns
- Standard CRUD operations
- Component structure following my specifications
- Consistent code formatting and conventions

What I Validate (100%)

- Every feature works as intended
- Database operations execute correctly
- State management behaves properly
- API endpoints respond appropriately
- User flows are smooth and logical
- Performance meets requirements
- Security best practices are followed

Phase 1: Build Foundation

Product Discovery

- Define clear product vision and target users
- Create comprehensive Product Backlog with user stories
- Prioritize features using impact vs. effort matrix
- Document business logic and user flows in Notion

Technical Architecture

- Select appropriate tech stack based on project requirements
- Design scalable system architecture
- Set up development environment with AI-augmented tools
- Configure MCP (Model Context Protocol) servers for enhanced AI context

Development Setup

```
Repository Structure → GitHub
Frontend Framework → Vue.js + Vite
AI Development → Claude Code + Cursor IDE
Documentation → Notion workspace
```

Phase 2: Ship Incrementally

Sprint-Based Development

- Work in focused iterations on prioritized features
- Maintain continuous integration practices
- Deploy features as they're completed
- Keep main branch always deployable

Quality Assurance

- Test during development, not after
- Use AI tools to identify potential issues early
- Implement automated deployment pipelines
- Monitor performance from day one

Infrastructure Strategy

```
Frontend Hosting → Netlify (automatic deploys)
Backend Services → Render (containerized applications)
Database → Supabase (real-time capabilities)
Content Management → Strapi (when user control needed)
```

Phase 3: Iterate Intelligently

Feedback Loops

- Monitor actual usage patterns
- Collect user feedback systematically
- Track technical performance metrics
- Document lessons learned

Continuous Improvement

- Prioritize bugs by user impact
- Refactor code for maintainability
- Optimize performance bottlenecks
- Scale infrastructure as needed

Technical Stack & Tools

Development Environment

AI-Augmented Development Workflow

- **Cursor IDE**: Primary development environment with AI integration
- **Claude Code**: Generates implementation based on my architectural decisions
- **MCP Servers**: Provide context-aware assistance for specific services (Supabase, Render, etc.)

How I Work with AI:

1. I design the feature and define requirements
2. I instruct Claude Code with specific implementation details
3. AI generates the code following my patterns
4. I review critical business logic and data flows
5. I test everything thoroughly in real scenarios
6. I debug and refine based on actual behavior

This approach allows me to move at 10x speed while maintaining quality and understanding.

Frontend Technologies

Core Framework

- **Vue.js 3**: Progressive JavaScript framework
- **Vite**: Next-generation frontend tooling
- **Tailwind CSS**: Utility-first CSS framework

Design & Prototyping

- **Figma**: Interface design and prototyping
- **Component Libraries**: Reusable UI patterns

Backend & Infrastructure

Services

- **Supabase**: PostgreSQL database with real-time subscriptions
- **Render**: Backend deployment and hosting
- **Netlify**: Frontend hosting with CI/CD

Optional Components

- **Strapi CMS**: Headless content management
- **API Integrations**: Third-party service connections

Project Management

Documentation & Control

- **Notion**: Central documentation hub
  - Product requirements
  - Technical specifications
  - Progress tracking
  - Knowledge base

Version Control

- **GitHub**: Code repository and collaboration
- **Conventional Commits**: Clear change history
- **Branch Protection**: Code quality gates

Implementation Process

Real Workflow Example

Let me show you how I actually work on a feature:

Feature: User Authentication System

1. **I Design** (30 minutes):
   - Define user roles (admin, manager, instructor)
   - Plan database schema (users, roles, permissions)
   - Decide auth flow (Supabase Auth + RLS policies)
   - Sketch UI components needed
2. **AI Implements** (2 hours with my guidance):

```
Me: "Create a Vue composable for authentication using Supabase,
    handling login, logout, and session management. Include
    error handling and loading states."

Claude Code: [Generates complete composable with all methods]

Me: "Add role-based route guards for admin-only pages"

Claude Code: [Implements router middleware]
```

3. **I Validate** (1 hour):
   - Test login with valid/invalid credentials
   - Verify session persistence

- Check role-based access works
- Test edge cases (expired tokens, network errors)
- Verify database user records are created correctly

Result: Complete auth system in ~3.5 hours vs ~2 days traditional coding

Week-by-Week Breakdown

Week 1-2: Foundation

1. **Product Definition**
   - Clarify business objectives
   - Define user personas
   - Create initial user story map
   - Set success metrics
2. **Technical Setup**
   - Initialize repository structure
   - Configure development environment
   - Install and configure MCP servers
   - Set up CI/CD pipelines

Week 3-6: Core Development

3. **MVP Implementation**
   - Develop core user flows
   - Implement essential features
   - Create responsive UI components
   - Integrate backend services
4. **Quality Checks**
   - Conduct functionality testing
   - Ensure responsive design
   - Optimize initial performance
   - Review security basics

Week 7-8: Production Ready

5. **Deployment**
   - Configure production environments
   - Set up monitoring tools

- Deploy to production
- Create user documentation

6. **Handover & Training**
   - Document system architecture
   - Create operational runbooks
   - Transfer knowledge to stakeholders
   - Establish support processes

Week 9+: Evolution

7. **Continuous Operations**
   - Monitor system health
   - Address user feedback
   - Implement enhancements
   - Scale as needed

Key Differentiators

Why This Approach Works

Architectural Focus

By delegating syntax to AI, I concentrate on what matters: system design, user experience, and business value. This leads to better architectural decisions and more thoughtful solutions.

Rapid Iteration

AI-generated code allows for quick experimentation and iteration. I can test multiple approaches fast, keeping what works and refining what doesn't.

Consistent Quality

Claude Code doesn't have bad days, doesn't make typos, and follows patterns consistently. The code quality is uniformly high, reducing technical debt.

Technology Agnostic

I can work effectively across different stacks (Vue, React, Node, Python) because I understand the concepts, while AI handles syntax variations.

What I Bring to the Table

Technical Understanding

- Deep knowledge of web architectures and patterns
- Clear understanding of frontend frameworks (Vue.js, state management)
- Backend API design and database modeling
- DevOps and deployment strategies

- Security and performance considerations

Product Thinking

- 10+ years of business development experience
- User-centric design approach
- Pragmatic feature prioritization
- Cost-benefit analysis mindset
- Stakeholder communication skills

Quality Assurance

- Comprehensive testing methodology
- Database verification and monitoring
- Performance optimization
- User flow validation
- Production debugging capabilities

Success Metrics

Development Efficiency

- User story completion rate
- Time from commit to deployment
- Bug discovery to resolution time
- Code review turnaround

Product Quality

- User satisfaction scores
- System uptime percentage
- Performance benchmarks
- Security audit results

Business Impact

- Time to market
- Development cost efficiency
- Feature adoption rates
- Stakeholder satisfaction

Case Study: Activitier

Challenge: Build a B2B SaaS platform for boutique hotels to manage activities with external instructors.

My Approach with Build-Ship-Iterate:

Architecture & Design (My 100% focus):

- Designed complete database schema with 15+ tables
- Defined role-based access control system
- Mapped all user flows and state management
- Selected tech stack based on requirements

Implementation (AI-Augmented):

- Used Claude Code to generate Vue.js components following my patterns
- Directed implementation of Supabase real-time subscriptions
- AI wrote CRUD operations while I ensured business logic accuracy
- Generated responsive UI with Tailwind based on my specifications

Validation & Refinement (My 100% control):

- Tested every user flow manually
- Verified all database operations in Supabase dashboard
- Debugged edge cases and race conditions
- Optimized performance bottlenecks
- Fixed UI/UX issues based on actual usage

Results:

- 97% complete MVP in 8 weeks (working part-time)
- Fully functional booking system with real-time updates
- Complex features like availability management and instructor scheduling
- Clean, maintainable codebase that follows Vue.js best practices

Key Learning: AI tools excel at implementation speed, but architectural decisions and quality validation remain fundamentally human responsibilities. The combination yields superior results.

Frequently Asked Questions

"Do you actually know how to code?"

Yes, absolutely. I understand programming logic, data structures, algorithms, and system design. I can read, debug, and modify any code. What I've optimized is the writing process—similar to how a writer might use voice dictation instead of typing. The understanding is there; the execution is more efficient.

"What if AI makes a mistake?"

That's why validation is crucial. I test every feature thoroughly, verify database operations, and monitor system behavior. When issues arise, I can identify and fix them because I understand what the system should do. AI is a tool, not a replacement for technical judgment.

"Can you work without AI tools?"

Yes, though it would be less efficient. It's like asking an architect if they can work without CAD software—of course they can, but why would they? The value I provide is in designing solutions and ensuring quality, not in typing syntax.

"How do you ensure code quality?"

Through multiple layers:

1. Clear architectural patterns that AI must follow
2. Comprehensive testing of all features
3. Database verification for data integrity
4. Performance monitoring in production
5. Code review of critical business logic
6. Consistent debugging when issues arise

"What's your actual technical level?"

I'm a solutions architect who codes. I understand:

- How Vue.js components and lifecycle work
- State management with Pinia
- RESTful API design and implementation
- SQL and database relationships
- Authentication and authorization flows
- Deployment and DevOps practices
- Performance optimization techniques

The difference is I implement these concepts efficiently using AI tools rather than typing everything manually.

"Why should we hire you over a traditional developer?"

Speed: I deliver working MVPs in weeks, not months **Perspective**: I think like a product owner AND a developer **Quality**: AI-generated code is consistent and follows best practices **Adaptability**: I can quickly work across different tech stacks **Focus**: My time goes to solving business problems, not syntax issues **Communication**: I bridge the gap between technical and business teams

Conclusion

Build-Ship-Iterate represents the future of product development—where human creativity and strategic thinking

combine with AI efficiency. This isn't about cutting corners or avoiding "real" programming. It's about evolving with the tools available to deliver better solutions faster.

I bring a unique combination:

- **Product mindset** from 10+ years in business development
- **Technical understanding** to architect robust solutions
- **Modern tooling** to implement efficiently
- **Quality focus** to ensure everything works properly

The result? Digital products that are well-architected, quickly delivered, and actually solve business problems.

For Potential Clients

You get someone who speaks your language (business value) while ensuring technical excellence. I won't waste time on unnecessary complexity, and I'll deliver working software you can see and test quickly.

For Potential Employers

You get a multiplier for your team—someone who can rapidly prototype, validate ideas, and build production-ready solutions while collaborating effectively with both technical and business stakeholders.

Let's discuss how this approach can accelerate your next project. Whether you need an MVP, a technical proof of concept, or someone to bridge product and development, I'm ready to deliver.