SPECSTORY

Top 25 Lessons Learned: Agentic Development

> **What This Is**: Hard-won lessons from building multiple production micro-SaaS products using AI agents. These insights cover the entire journey from ideation to deployment, focusing on what actually works in practice.

> **Who This Is For**: Those using AI coding assistants (Cursor, Claude Code, Copilot, ETC) to ship real products. Whether you're building your first AI-assisted project or optimizing your workflow, learning from these lessons will accelerate your success.

🎯 Part 1: Ideation & Requirements Gathering

The Power of Unstructured Capture

1. Freeform Brainstorming Works

When starting, don't try to write a perfect spec. Instead:

- Get all related ideas down in one place
- Don't worry about structure initially
- Let ideas flow naturally and connect them later

2. Voice is Your Superpower

- **Talk it out**: Speak for 5-10 minutes about your idea. Use a product like SuperWhisper or Wispr Flow
- **Don't stop**: Keep talking even when you think you're done
- **Convert**: This audio becomes valuable text input for agents

3. Leverage Existing Content

- **Team recordings**: Use interview transcripts and meeting recordings (with light editing)
- **Visual inspiration**: Add screenshots of apps you like
- **Documentation**: Include any existing background materials

4. User Research Multiplier

> "It's a 10x improvement if you can talk to half a dozen potential users"

- Real conversations beat imagined scenarios
- Focus on problem space before solution space
- Use both open and closed-ended questions strategically

🏗️ Part 2: Technology Stack Decisions

Critical Human-in-the-Loop Decisions

5. Package Selection Requires Human Review

Before accepting AI package suggestions:

- **Check health signals**:
  - Recency of commits
  - Download numbers
  - GitHub stars
  - Active maintainers
- **Review documentation quality**
- **Look for working demos**
- **Save good documentation URLs** for agent context

6. Version Control for Packages is Critical

- AI often suggests outdated package versions
- Always verify latest stable versions
- Check compatibility between packages

7. API & Service Provider Selection

Consider these factors when choosing services:

Factor	Questions to Ask
Provider Type	Major cloud? Minor cloud? AI service? Independent?
Cost	Pay-per-use? Monthly minimums? Free tier limits?
Configuration	How complex is setup? Clear documentation?
Integration	Well-supported SDKs? Good examples?

8. **Form Factor First**

Choose early: Web, mobile, desktop, or watch app

- Without a choice, AI will wander or decide for you
- You can change later, but starting focused saves time

9. **Language & Framework Selection**

Less important than you think, except for:

- **AI capability**: Avoid obscure languages with poor AI support
- **Package ecosystem**: Rich ecosystems = faster development
- **Your familiarity**: Know enough to spot obvious errors

📐 Part 3: Architecture Fundamentals

Know Your Building Blocks

10. **Web Architecture Essentials**

- Client (Browser)
- Backend Server
- REST APIs
- Databases
- Routing
- Authentication
- Build/Deployment & Hosting

11. Mobile Architecture Components

- Client (Mobile App)
- Backend Server & APIs
- Local Storage
- Authentication & Push Notifications
- App Lifecycle Management
- Platform considerations (iOS/Android/Cross-platform)
- Device APIs & Permissions
- App Distribution

12. Desktop Architecture Elements

- Client (Desktop App)
- Native vs. Multi-platform decisions
- Local "Backend" considerations
- Local Storage patterns
- Authentication flows
- Update mechanisms
- OS Integration (file system, clipboard)
- Distribution strategies

🤖 Part 4: Working with AI Agents

Context is King

13. Context Priming Strategy

- Ask questions about the relevant area before writing specs
- Load agent memory with domain knowledge
- Provide examples of what you want

14. Strategic Ambiguity

Sometimes leaving things unclear is useful:

- Let AI help think through choices
- See what it comes up with during implementation
- Roll back if you don't like it, roll forward if promising

15. **Multi-Modal is Powerful**

- **Screenshots are your friend**
- Visual examples often communicate better than text
- Use diagrams for architecture decisions

Managing Agent Behavior

16. **Know When to Reset**

Ask the agent to step back and rethink when:

- **Dead loops**: Same error 2-3 times
- **Wrong structure**: Files/directories look off
- **Confusion signs**: Agent's summary doesn't make sense

17. **Development Environment Awareness**

Understand your tools:

- When do changes need a server restart?
- When does hot-reloading work?
- When to do a clean rebuild?

Pro tip: When in doubt, ask the AI: "Do I need to restart the server for this change?"

🔧 Part 5: Development Workflow

Productivity Patterns

18. **Parallel Project Management**

> "Pretend you're managing a pair of engineers"

- Work on two projects simultaneously
- Switch context while agents work
- Avoid idle time watching output

19. **Database Schema Management**

For migrations and schema changes:

1. Dump entire schema from your database
2. Feed as context to agent
3. Generate accurate migrations
4. Review before applying

Quality Control

20. **Common Error Patterns**

Watch for:

- Off-by-one errors
- Incorrect assumptions about state
- Package version mismatches
- Missing error handling

21. **Documentation Strategies**

- Use `CLAUDE.md` or similar for project-specific instructions
- Document gotchas and patterns
- Include adversarial review notes in slash commands

🚀 Part 6: Advanced Techniques

Problem-Solving Patterns

22. **The Simplification Strategy**

When debugging complex issues:

1. Have AI write a simplified version of the feature
2. Load context with the "clean" approach
3. Apply learnings to fix the complex version

23. **Feature Reduction Exercise**

Periodically review and remove unnecessary features:

- Improves performance
- Reduces complexity
- Makes maintenance easier

Production Readiness

24. **Performance Optimization**

- Use tools like PageSpeed Insights
- Focus on Core Web Vitals
- Implement SEO best practices

25. **Product Polish**

- Convert apps to product videos using tools like Remotion
- Create compelling demos
- Polish user experience details

📋 Quick Reference Checklist

Before Starting

- [ ] Define form factor (web/mobile/desktop)
- [ ] Choose language/framework based on AI support
- [ ] Set up voice capture for brainstorming

During Development

- [ ] Review all package suggestions
- [ ] Verify package versions
- [ ] Check API documentation quality
- [ ] Maintain context with screenshots
- [ ] Work on parallel tasks

When Stuck

- [ ] Ask agent to step back and rethink
- [ ] Try the simplification strategy
- [ ] Check if server restart needed
- [ ] Review file/folder structure

Before Shipping

- [ ] Run performance analysis
- [ ] Remove unnecessary features
- [ ] Create product demo
- [ ] Document lessons learned

🎓 Key Takeaways

1. **Human judgment matters most** in technology selection and architecture decisions
2. **Context management** is the key to effective agent collaboration
3. **Parallel work** maximizes productivity with agents
4. **Visual communication** (screenshots) often beats text
5. **Know when to reset** rather than pushing through errors
6. **Real user feedback** is worth 10x imagined scenarios
7. **Simplification** is a powerful debugging strategy

Questions about these lessons? Let us know!

- greg@specstory.com
- jake@specstory.com

Remember: These lessons come from real-world experience building micro-SaaS products with AI agents. Adapt them to your context and continue learning from your own experiments.