Steve AI - Creative Coding Agent Development Brief

Executive Summary

Steve is a creative coding AI agent built on the open-source Bolt.diy framework, specialized in generating visually stunning creative projects that current AI tools fail to produce. While tools like Replit generate "functional but ugly" code, Steve produces professional-quality visual experiences through enhanced prompting, real-time research, and curated creative libraries.

The Problem We're Solving

Current State: AI coding tools (Replit, Lovable, etc.) generate functional but aesthetically terrible creative projects

- Three.js solar systems that are just black screens
- Data visualizations that look like high school projects
- Web experiences with no visual polish or creative flair

User Pain Point: "I want to vibe code cool shit, but every AI tool spits out garbage"

Our Solution: Steve AI

Positioning: "Free tools build functional apps. Steve builds apps that make people say 'holy shit, how did you make that?'"

Core Differentiators

1. **Creative Specialization** - Focus on visual projects (Three.js, P5.js, data viz, creative web)
2. **Real-time Research** - Actively scrapes current best practices and techniques
3. **Curated Creative Stack** - Pre-loaded with premium creative coding libraries
4. **Visual QA Loop** - Tests output, detects "black screens," iterates until it looks amazing
5. **Enhanced Prompting** - Transforms "solar system" into detailed creative specifications

Technical Architecture

Foundation: Bolt.diy Fork

- **Base:** Open-source Bolt.diy (MIT licensed)
- **Infrastructure:** File management, LLM integration, terminal, preview system
- **Modification Approach:** Add creative intelligence layers to existing architecture

Steve's Creative Intelligence Layers

Layer 1: Vision Translation Engine

```
User Input: "Build me a Three.js solar system"
↓
Steve's Logic:
- Solar system = planets + starfield + bloom + realistic lighting + smooth orbits
- Research current Three.js best practices
- Add modern visual enhancements
↓
Enhanced Prompt: "Build Three.js solar system with 10k particle starfield,
post-processing bloom effects, realistic planet textures, smooth orbital
animations using 2024 best practices..."
```

Layer 2: Real-time Research System

- **Web scraping** for current creative coding techniques
- **Library discovery** - find latest Three.js tools, shader libraries
- **Best practices extraction** from dev blogs, GitHub repos, CodePen
- **Trend analysis** - what makes modern creative projects look good

Layer 3: Creative Component Libraries (Pre-loaded)

Three.js Ecosystem:

- React Three Fiber, Drei, Leva
- Post-processing (bloom, film grain, chromatic aberration)
- Particle systems, shader libraries

Creative Coding Stack:

- P5.js for generative art
- D3.js for data visualization
- GSAP for animations
- Modern design tokens and styling

Layer 4: Visual QA & Iteration Loop

- **Execution monitoring** - run code, take screenshots
- **Quality detection** - identify black screens, boring output

- **Iterative improvement** - generate fixes until output meets creative standards
- **Aesthetic validation** - built-in rules for what looks good

Curated Tech Stack Strategy

Instead of supporting everything, Steve specializes in:

- **Frontend:** React/Next.js + creative libraries
- **3D/Graphics:** Three.js + React Three Fiber ecosystem
- **Styling:** Tailwind + modern design patterns
- **Animation:** GSAP, Framer Motion
- **Data Viz:** D3.js + interactive components

Development Timeline: Sprint Approach

Week 1: Core Foundation

Days 1-2: Setup & Research

- Fork Bolt.diy repository
- Set up with Gemini API integration
- Study codebase architecture
- Identify modification points

Days 3-5: Enhanced Prompting System

- Modify prompt enhancement logic in `app/lib/.server/llm/prompts.ts`
- Add creative decision rules for different project types
- Implement vision translation engine

Days 6-7: Research Integration

- Add web scraping capabilities
- Build research-to-prompt pipeline
- Test with basic Three.js examples

Week 2: Creative Libraries & QA

Days 8-10: Component Library Integration

- Pre-load Three.js ecosystem into build system

- Add creative coding templates
- Configure enhanced tech stack

Days 11-14: Visual QA System

- Implement execution monitoring
- Add screenshot analysis capabilities
- Build iteration logic for quality improvement

Development Team Structure

Jeff (Project Lead/Implementer)

- Final decision making
- Code implementation using RootCode assistance
- Testing and validation
- Handles actual development work

Claude (Senior Architect)

- Technical strategy and architecture
- File modification guidance
- System design and integration planning
- Debugging and optimization strategies

RootCode (Structured Development Assistant)

- Code generation and modification
- Codebase navigation and understanding
- Detailed implementation following instructions
- Systematic task breakdown and execution

Business Model

Bootstrap Strategy

- **Development:** Use existing $100 Gemini API credits
- **Launch:** Users provide their own API keys (like current Bolt.diy model)
- **Revenue:** $20-50/month subscription for enhanced features

Go-to-Market

1. **Build 5-10 impressive demos** showing Steve vs Replit comparisons
2. **Document everything** with screen recordings
3. **Launch "Steve Beta"** to creative coding community
4. **Iterate based on user feedback**

Target Market

- Indie developers who care about aesthetics
- Creative professionals learning to code
- Portfolio projects requiring visual wow-factor
- Client work where design quality matters

Competitive Advantage

1. **Creative Specialization** - Only tool focused on visual quality over functionality
2. **Real-time Research** - Always current with latest techniques
3. **Curated Experience** - Pre-configured for creative success
4. **Quality Assurance** - Actually tests that output looks good
5. **Local Privacy** - Code stays on user's machine

Success Metrics

- **Technical:** Steve consistently produces better-looking output than free alternatives
- **User:** "Holy shit" reactions to Steve-generated projects
- **Business:** $10K+ MRR within 90 days of launch
- **Market:** Become go-to tool for creative coding projects

Risk Mitigation

- **Competition:** Focus on creative niche vs general development
- **Technical:** Build on proven Bolt.diy foundation
- **Resources:** Bootstrap approach using existing API credits
- **Market:** Solve real problem we're personally experiencing

Next Steps

1. **Immediate:** Fork Bolt.diy and set up development environment

2. **Week 1:** Implement enhanced prompting system
3. **Week 2:** Add research and QA capabilities
4. **Week 3:** Build demo projects and document results
5. **Week 4:** Launch MVP to creative coding community

"The goal isn't to build another AI coding tool. It's to build the first AI that makes creative coding actually look creative."