# AI-Powered Web Development: Vibe-Coding with Windsurf, GitHub, and Vercel

## Table of Contents

---

## What is Vibe-Coding?

**Vibe-coding** is a modern development approach where you use natural language prompts to guide AI in building software, rather than writing every line of code manually. With Windsurf's Cascade AI, you describe what you want, and the AI generates the implementation.

### Traditional vs Vibe-Coding

| Traditional | Vibe-Coding with AI |
|---|---|
| Write code manually | Describe what you want |
| Search Stack Overflow | Ask AI for solutions |
| Debug for hours | AI suggests fixes instantly |
| Read documentation | AI explains and implements |
| Slow iteration | Rapid prototyping |

### Why It Works

- **Speed**: Build prototypes in minutes, not hours
- **Learning**: AI explains concepts as it codes
- **Exploration**: Try ideas without deep technical knowledge
- **Focus**: Concentrate on design and user experience

---

## Setting Up Your Project Folder in Windsurf

### Creating a New Folder in the Documents Directory

Before starting your project, you need to create a dedicated folder to store all your files. Here's how to do it within Windsurf:

**Method 1: Using the Explorer Panel (Recommended)**

1. **Open Windsurf**
   - Launch the Windsurf IDE from your desktop or Start menu
2. **Access the Explorer Panel**
   - Look for the file icon in the left sidebar (or press `Ctrl/Cmd + Shift + E`)
   - This shows your current workspace structure
3. **Navigate to Documents**
   - Click on `File` → `Open Folder`
   - Navigate to `C:\Users\YourUsername\Documents` (Windows) or `/Users/YourUsername/Documents` (Mac)
   - Click `Select Folder` or `Open`
4. **Create Your Project Folder**
   - Right-click in the Explorer panel on the Documents folder
   - Select `New Folder` from the context menu
   - Type your project name (e.g., `my-website`)
   - Press `Enter` to confirm
5. **Open the New Folder**
   - Click `File` → `Open Folder` again
   - Navigate to your newly created folder
   - Click `Open`
   - Windsurf will reload with your project folder as the workspace

**Method 2: Using the Terminal**

1. **Open Terminal in Windsurf**
   - Press `Ctrl+`` (backtick) or go to `Terminal` → `New Terminal`
   - The terminal appears at the bottom of the screen
2. **Create the Folder**

```
# Navigate to Documents
cd ~/Documents

# Create project folder
mkdir my-website

# Navigate into the folder
cd my-website

# Open this folder in Windsurf
code .
```

3. **Windsurf Opens the Folder**
   - The folder is now your active workspace

  - You're ready to start your project

**Method 3: Using Cascade (AI)**

1. **Open Cascade Panel**
   - Press `Ctrl/Cmd + L`
2. **Ask AI to Create the Folder**

   ```
   Create a new folder called "my-website" in my Documents directory
   and open it as the current workspace.
   ```

3. **Cascade Will:**
   - Create the folder for you
   - Set it as the active workspace
   - Confirm the action

---

## Managing Projects with Workspace Files

### What is a Workspace File?

A workspace file ( `.code-workspace` ) is a configuration file that saves your project settings, open files, and folder structure. It allows you to quickly reopen entire projects with all your preferences intact.

### Benefits of Using Workspace Files

- **Quick Access**: Open previous projects instantly
- **Saved State**: Remember open files, terminal history, and settings
- **Multiple Projects**: Switch between different projects easily
- **Consistency**: Same environment every time you open the project
- **Team Sharing**: Share workspace configurations with teammates

### Creating a Workspace File

**Method 1: Save Current Workspace**

1. **Open Your Project Folder**
   - Ensure your project is loaded in Windsurf
   - Check the Explorer panel shows your project files
2. **Save Workspace**
   - Click `File` → `Save Workspace As...`
   - Navigate to a location (e.g., `Documents/Workspaces/` )
   - Name the file: `my-project.code-workspace`
   - Click `Save`
3. **Workspace File Created**
   - A `.code-workspace` file is saved
   - Contains paths to your project folders and settings

**Method 2: Create from Scratch**

1. **Create New Workspace**
    - Click `File` → `Open Workspace from File...`
    - Or `File` → `New Window` → then add folders
2. **Add Folders to Workspace**
    - Click `File` → `Add Folder to Workspace`
    - Select your project folder
    - Repeat for multiple projects
3. **Save the Workspace**
    - `File` → `Save Workspace As...`
    - Name and save the file

## Opening a Previous Project

### Option 1: Double-Click Workspace File

1. Navigate to where you saved `.code-workspace` file
2. Double-click the file
3. Windsurf opens with your complete project setup

### Option 2: From Within Windsurf

1. Click `File` → `Open Workspace from File...`
2. Navigate to your `.code-workspace` file
3. Click `Open`
4. Your project loads with all previous settings

### Option 3: Recent Workspaces

1. Click `File` → `Open Recent`
2. Select from list of recent workspaces
3. Project opens instantly

## Workspace File Structure

A workspace file looks like this:

```json
{
  "folders": [
    {
      "path": "my-website"
    }
  ],
  "settings": {
    "editor.fontSize": 14,
    "editor.tabSize": 2
  },
  "extensions": {
    "recommendations": [
      "esbenp.prettier-vscode",
      "bradlc.vscode-tailwindcss"
    ]
  }
}
```

### Best Practices for Workspaces

1. **Organize by Client/Project**

   ```
   Workspaces/
   ├── client-a-website.code-workspace
   ├── personal-blog.code-workspace
   ├── ecommerce-project.code-workspace
   └── tutorial-learning.code-workspace
   ```

2. **Include Multiple Related Folders**

   - Frontend and backend in same workspace
   - Shared components library + main project

3. **Version Control Workspaces**

   - Commit `.code-workspace` files to Git
   - Team members get same setup

4. **Customize Per Project**

   - Different color themes for different clients
   - Specific extensions for project type

---

## Project Planning Before Opening Windsurf

### Why Plan Before Coding?

Planning before opening your IDE helps you:

- Clarify project goals and requirements
- Make better use of AI assistance
- Avoid rework and refactoring
- Stay focused on the end result
- Communicate clearly with Cascade

### The Pre-Development Checklist

#### Step 1: Define the Project Purpose

**Questions to Answer:**

- What problem does this project solve?
- Who is the target audience?
- What is the primary goal? (inform, sell, entertain, educate)
- What makes this project unique?

**Document Your Answers:**

```
Project: Portfolio Website
Purpose: Showcase my work to potential clients
```

```
Audience: Small business owners looking for web design
Goal: Generate leads through contact form
Unique Factor: Interactive project gallery with live demos
```

**Step 2: Plan the Content Structure**

**Outline Your Pages:**

```
Home
├── Hero section (headline, CTA)
├── Services overview
├── Featured projects (3-4 items)
├── Testimonials
└── Contact CTA

About
├── Bio/Story
├── Skills/Tech stack
├── Experience timeline
└── Download resume button

Portfolio
├── Project grid
├── Filter categories
├── Project detail modals
└── Live demo links

Contact
├── Contact form
├── Social links
├── Email/phone info
└── Location map
```

**List Required Assets:**

- Logo (SVG or PNG)
- Project screenshots (16:9 ratio)
- Profile photo (square)
- Resume PDF
- Favicon

**Step 3: Define the Design System**

**Color Palette:**

```
Primary: #DC2626 (Red)
Secondary: #1F2937 (Dark Gray)
Accent: #10B981 (Green for success)
Background: #FFFFFF (White)
Text: #111827 (Near Black)
```

**Typography:**

```
Headings: Inter, sans-serif
Body: Inter, sans-serif
Code: JetBrains Mono, monospace
```

**Component Ideas:**

- Navigation: Sticky header, mobile hamburger
- Buttons: Primary (filled), Secondary (outline)
- Cards: Rounded corners, subtle shadow
- Forms: Clean labels, inline validation

**Step 4: Technical Decisions**

**Framework & Tools:**

- Next.js with App Router
- TypeScript for type safety
- Tailwind CSS for styling
- shadcn/ui for components
- Framer Motion for animations

**Integrations:**

- Contact form → Email service (Resend/Formspree)
- Analytics → Google Analytics or Plausible
- Hosting → Vercel
- Domain → Custom domain (optional)

**Data Structure:**

```typescript
// Project data structure
interface Project {
  id: string;
  title: string;
  description: string;
  image: string;
  tags: string[];
  liveUrl: string;
  githubUrl?: string;
}
```

**Step 5: Create a Prompt Strategy**

**Prepare Initial Prompt:**

```
Create a portfolio website for a web developer with:
- Dark red (#DC2626) primary color
- Clean, modern design
- Hero section with "John Doe - Full Stack Developer"
- Services: Web Design, Development, SEO
- Portfolio grid with 6 projects
- Contact form with validation
- Responsive design
- Dark mode support
```

**Plan Follow-Up Prompts:**

1. "Add animations to the hero section"
2. "Create project detail modals"
3. "Add form validation and error handling"
4. "Optimize for SEO with meta tags"
5. "Add loading states and transitions"

**Step 6: Set Up Your Environment**

**Before Opening Windsurf:**

1. ✅ Project folder created
2. ✅ Planning documents ready
3. ✅ Assets collected in project folder
4. ✅ Color palette documented
5. ✅ Content outline complete

**Organize Assets:**

```
my-website/
├── assets/
│   ├── logo.svg
│   ├── profile-photo.jpg
│   ├── project-1.jpg
│   ├── project-2.jpg
│   └── resume.pdf
└── planning/
    ├── content-outline.md
    ├── design-system.md
    └── prompt-strategy.md
```

## Using Your Plan with Cascade

**Start with Context:**

```
I'm building a portfolio website. Here's my plan:
- Target: Small business clients
- Pages: Home, About, Portfolio, Contact
- Style: Modern, professional, red accent color
- Features: Dark mode, contact form, project gallery

Let's start by creating the project structure and home page.
```

**Reference Your Plan:**

```
According to my design system (primary color #DC2626),
create a navigation component with:
- Logo on left
- Links: Home, About, Portfolio, Contact
- CTA button "Hire Me" in red
- Mobile hamburger menu
```

### Quick Reference: Planning Template

```
# Project Plan: [Name]

## Purpose
- Problem: [What it solves]
- Audience: [Who it's for]
- Goal: [What success looks like]

## Pages & Content
1. [Page Name]
   - [Section 1]
   - [Section 2]

## Design
- Primary: #[COLOR]
- Style: [Modern/Classic/Playful]
- Fonts: [Font names]

## Tech Stack
- Framework: Next.js
- Styling: Tailwind CSS
- Components: shadcn/ui
- Animations: Framer Motion

## Assets Needed
- [ ] Logo
- [ ] Images
- [ ] Copy/text

## Prompt Strategy
1. [First prompt]
2. [Second prompt]
3. [Third prompt]
```

---

## Understanding the Technologies

Before diving into development, let's understand the key technologies that power your workflow:

### Next.js

**What is Next.js?** Next.js is a React framework that enables server-side rendering, static site generation, and full-stack web applications. It provides the foundation for building modern, fast, and SEO-friendly websites.

**Key Features:**

- **App Router**: Modern routing system with nested layouts
- **Server Components**: Render components on the server for better performance
- **Static Generation**: Pre-build pages for instant loading
- **API Routes**: Build backend endpoints within your Next.js app
- **Image Optimization**: Automatic image resizing and optimization

- **TypeScript Support**: Built-in type safety

**Why Use It?**

- Faster page loads through server-side rendering
- Better SEO than traditional single-page apps
- Automatic code splitting for smaller bundles
- Developer-friendly with hot reloading
- Production-ready out of the box

**Example:**

```
// app/page.tsx
export default function Home() {
  return (
    <main>
      <h1>Welcome to Next.js</h1>
    </main>
  )
}
```

---

## Git

**What is Git?** Git is a distributed version control system that tracks changes in your code over time. It allows you to save snapshots of your project, collaborate with others, and revert changes when needed.

**Key Concepts:**

- **Repository**: A folder containing your project and its history
- **Commit**: A snapshot of your files at a specific point in time
- **Branch**: An independent line of development
- **Staging**: Preparing files to be committed
- **History**: Complete record of all changes made

**Why Use It?**

- Track every change to your code
- Experiment safely with new features
- Collaborate without overwriting others' work
- Revert mistakes instantly
- Maintain multiple versions of your project

**Essential Commands:**

```
# Start tracking a project
git init

# Save changes with a message
git add .
git commit -m "Added hero section"

# View history
git log
```

```
# Check current status
git status
```

---

## GitHub

**What is GitHub?** GitHub is a web-based platform that hosts Git repositories. It provides a centralized place to store your code, collaborate with others, and manage projects.

**Key Features:**

- **Remote Repositories**: Store code in the cloud
- **Collaboration Tools**: Pull requests, code reviews, issues
- **Actions**: Automated workflows for testing and deployment
- **Pages**: Host static websites directly from repositories
- **Teams**: Organize contributors and permissions

**Why Use It?**

- Backup your code safely in the cloud
- Share projects with others
- Collaborate on team projects
- Showcase your work to potential employers
- Integrate with deployment platforms like Vercel

**Workflow:**

```
# Connect local to remote
git remote add origin https://github.com/username/repo.git

# Upload changes
git push -u origin main

# Download changes
git pull origin main
```

---

## Vercel

**What is Vercel?** Vercel is a cloud platform for static sites and serverless functions. It provides instant deployment, automatic scaling, and global CDN distribution.

**Key Features:**

- **Zero-Config Deployment**: Push to GitHub, auto-deploy
- **Preview URLs**: Every branch gets its own URL
- **Serverless Functions**: API endpoints without managing servers
- **Edge Network**: Global CDN for fast loading worldwide
- **Analytics**: Built-in performance monitoring
- **Environment Variables**: Secure configuration management

**Why Use It?**

- Deploy websites with a single `git push`
- Automatic HTTPS and custom domains

- Instant rollbacks to previous versions
- Preview deployments for testing
- Optimized for Next.js applications
- Free tier for personal projects

**Deployment Process:**

1. Connect GitHub repository to Vercel
2. Push code to main branch
3. Vercel automatically builds and deploys
4. Site is live at `your-project.vercel.app`

---

## Windsurf IDE

**What is Windsurf?** Windsurf is an AI-powered integrated development environment (IDE) that combines code editing with artificial intelligence assistance. It's built on top of VS Code with added AI capabilities through Cascade.

**Key Features:**

- **Cascade AI**: Natural language coding assistant
- **Codeium Integration**: AI-powered autocomplete
- **VS Code Compatibility**: All VS Code extensions work
- **Integrated Terminal**: Command line access within the editor
- **File Explorer**: Visual file and folder management
- **Debugger**: Built-in debugging tools
- **Git Integration**: Visual diff and commit tools

**Why Use It?**

- Write code faster with AI assistance
- Get instant help with errors and bugs
- Generate code from natural language descriptions
- Refactor and improve code with AI suggestions
- Explain complex code in plain English
- All the power of VS Code plus AI

**Cascade in Action:**

```
User: "Create a navigation component with mobile hamburger menu"

AI: Generates complete React component with:
- Desktop navigation bar
- Mobile hamburger menu
- Responsive design
- TypeScript types
- Styling with Tailwind CSS
```

**Essential Shortcuts:**

- `Ctrl/Cmd + L` - Open Cascade AI panel
- `Ctrl/Cmd + Shift + E` - Open Explorer
- `` Ctrl+` `` - Open Terminal
- `Ctrl/Cmd + P` - Quick file navigation
- `Ctrl/Cmd + Shift + F` - Search across files

---

## The AI Prompting Workflow

### Understanding Cascade (Windsurf's AI)

Cascade is your AI pair programmer. Access it:

- **Shortcut**: `Ctrl/Cmd + L`
- **Command**: Type `@` to reference files
- **Context**: AI sees your current file and workspace

### The Vibe-Coding Loop

```
1. PROMPT → Describe what you want
2. REVIEW → AI suggests changes
3. ACCEPT → Apply the changes
4. TEST → See if it works
5. ITERATE → Refine with follow-up prompts
```

---

## Project Lifecycle with AI

### Phase 1: Project Initialization with AI

**Prompt:**

```
Create a new Next.js project with:
- TypeScript support
- Tailwind CSS for styling
- App Router enabled
- shadcn/ui components
- Dark mode support
- A hero section on the home page
- Navigation with links to Home, About, and Contact
```

**What Cascade Does:**

1. Runs `npx create-next-app@latest`
2. Configures TypeScript and Tailwind
3. Installs shadcn/ui
4. Sets up dark mode with next-themes
5. Creates initial page components
6. Builds a responsive navigation

**Your Action:**

- Review each file Cascade creates
- Accept or reject changes
- Ask for modifications if needed

---

### Phase 2: Building Features with Prompts

**Example 1: Creating a Hero Section**

**Prompt:**

```
Create a hero section for the home page with:
- Large headline: "Build Faster with AI"
- Subheadline explaining our service
- Call-to-action button "Get Started"
- Background gradient animation
- Responsive design for mobile
- Use framer-motion for subtle entrance animations
```

**Cascade Generates:**

- `app/components/hero.tsx`
- Animations with Framer Motion
- Responsive Tailwind classes
- Gradient background component

**Example 2: Building a Contact Form**

**Prompt:**

```
Create a contact form on the contact page with:
- Fields: name, email, message
- Form validation using React Hook Form and Zod
- Submit button with loading state
- Success message after submission
- Send data to /api/contact endpoint
- Styled with Tailwind and shadcn Input components
```

**Cascade Generates:**

- Form component with validation
- API route for form submission
- Error handling
- Loading states
- Success feedback

**Example 3: Adding a Portfolio Gallery**

**Prompt:**

```
Add a portfolio section to showcase projects:
- Grid layout with 3 columns on desktop, 1 on mobile
- Each card shows project image, title, description
- Hover effects with scale and shadow
- Click to open project details in a modal
- Use project data from a projects.ts file
- Include featured project: Raccoon Muncher game at raccoon-muncher.vercel.app
```

**Cascade Generates:**

- Portfolio grid component
- Project card with hover effects
- Modal for project details
- Data structure for projects

- Responsive grid layout

---

## Phase 3: Styling and Refinement

### Prompt for Design System

**Prompt:**

```
Create a consistent design system:
- Color palette: primary red (#DC2626), dark gray for backgrounds
- Typography scale using Tailwind defaults
- Button variants: primary (red), secondary (outline), ghost
- Card component with border and hover effects
- Spacing consistency (4px grid)
- Apply to all existing components
```

**Cascade:**

- Updates `tailwind.config.ts` with custom colors
- Creates reusable button component
- Standardizes spacing across components
- Applies design system to existing pages

### Prompt for Responsive Design

**Prompt:**

```
Make the entire website fully responsive:
- Mobile-first approach
- Navigation becomes hamburger menu on mobile
- Hero text scales down on smaller screens
- Grid layouts collapse to single column
- Test breakpoints: sm (640px), md (768px), lg (1024px)
- Ensure no horizontal scrolling
```

---

## Phase 4: Adding Interactivity

### Prompt for Animations

**Prompt:**

```
Add scroll animations throughout the site:
- Fade in sections as user scrolls
- Stagger animation for list items
- Smooth scroll to sections
- Use Framer Motion or AOS library
- Keep animations subtle and professional
```

### Prompt for State Management

**Prompt:**

```
Add a theme toggle (light/dark mode):
- Toggle button in navigation
- Persist user preference in localStorage
- Apply dark: variants to all components
- Smooth transition between themes
- Use next-themes for SSR compatibility
```

---

## Advanced AI Prompting Techniques

### 1. Progressive Disclosure

Start broad, then narrow down:

**Step 1 - Broad:**

```
Create an about page for my digital agency
```

**Step 2 - Specific:**

```
Add to the about page:
- Team section with 3 member cards
- Each card has photo, name, role, and short bio
- Photos use placeholder avatar
- Grid layout: 3 columns desktop, 1 column mobile
```

**Step 3 - Polish:**

```
Add hover effects to team cards:
- Scale up slightly on hover
- Show social icons (LinkedIn, Twitter) on hover
- Smooth transition animations
```

### 2. Reference Existing Code

Use `@` to point to specific files:

```
Update @app/page.tsx to include a testimonials section below the hero.
Use the same card style as @components/ui/card.tsx.
```

### 3. Context-Aware Prompts

Cascade remembers previous context:

```
Follow-up: Also add a pricing section with 3 tiers:
- Basic: $99/month
- Pro: $199/month
- Enterprise: Contact us

Use the same styling as the features section we just created.
```

### 4. Debug with AI

When something breaks:

**Prompt:**

```
The build is failing with this error:
"Type error: Cannot find name 'Badge'"

Fix the import issue in @client/src/pages/home.tsx
```

**Cascade:**

- Identifies the missing import
- Adds `import { Badge } from "@/components/ui/badge"`
- Explains why the error occurred

### 5. Refactoring Prompts

**Prompt:**

```
Refactor the navigation component:
- Extract mobile menu into separate component
- Use custom hook for menu state
- Add TypeScript interfaces
- Keep all existing functionality
```

---

## From Prompt to Production

### The Complete AI-Guided Workflow

**Step 1: Initial Setup (AI-Powered)**

```
Prompt: "Initialize a Next.js project with all the modern defaults"
```

**Cascade Actions:**

- Creates project structure
- Sets up TypeScript
- Configures Tailwind
- Initializes shadcn/ui
- Creates basic layout

**Step 2: Build Core Pages (Iterative Prompting)**

```
Prompt 1: "Create a landing page with hero, features grid, and CTA"
Prompt 2: "Add an about page with company story and team"
Prompt 3: "Build a contact page with form and map"
```

**Step 3: Add Features (Specific Prompts)**

```
Prompt: "Add a blog section with:
- List of posts on /blog
- Individual post pages at /blog/[slug]
- MDX support for rich content
- Syntax highlighting for code blocks"
```

**Step 4: Polish and Optimize**

```
Prompt: "Optimize the site for production:
- Add SEO meta tags to all pages
- Generate sitemap.xml
- Add loading states
- Optimize images with next/image
- Add analytics integration"
```

**Step 5: Deploy to Vercel**

```
Prompt: "Help me deploy this to Vercel:
- Connect to GitHub repository
- Set up automatic deployments
- Configure environment variables
- Add custom domain (optional)"
```

**Cascade guides you through:**

1. Creating GitHub repo
2. Pushing code
3. Importing to Vercel
4. Configuring settings
5. Deploying

---

## Best Practices for AI Collaboration

### DO:

✅ **Be Specific**

```
Good: "Create a blue primary button with white text, rounded corners,
       that changes to darker blue on hover"

Vague: "Make a button"
```

✅ **Provide Context**

```
Good: "Following the design system in @tailwind.config.ts,
       create a card component with our brand colors"
```

✅ **Iterate Incrementally**

```
Prompt 1: "Create basic form structure"
Prompt 2: "Add validation to the form"
Prompt 3: "Style the form errors"
```

✅ **Review AI Output**

- Check for security issues
- Verify TypeScript types

- Test functionality

✅ **Learn from AI**

```
Prompt: "Explain how the useEffect hook works in the component you just created"
```

### DON'T:

❌ **Don't blindly accept** - Always review AI-generated code

❌ **Don't make prompts too complex** - Break into smaller tasks

❌ **Don't ignore errors** - Ask AI to explain and fix issues

❌ **Don't skip testing** - Verify locally before pushing

---

## Prompt Templates by Task

### Website Structure

```
Create a [business type] website with:
- Home page with hero, features, testimonials, CTA
- About page with story and team
- Services page with [number] service cards
- Contact page with form and contact info
- Navigation and footer on all pages
- Responsive design
- [Color scheme] color palette
```

### Component Creation

```
Create a [component name] component that:
- Accepts props: [list props]
- Renders: [describe output]
- Has states: [list states]
- Uses: [specific libraries if any]
- Matches style: [reference existing components]
```

### API Integration

```
Create an API route at /api/[endpoint] that:
- Accepts [method] requests
- Validates [input data]
- Connects to [service/database]
- Returns [response format]
- Handles errors with [error format]
```

### Styling Tasks

```
Apply these design updates:
- Color scheme: [colors]
- Typography: [font preferences]
- Spacing: [spacing scale]
- Components to update: [list]
- Breakpoints: [responsive requirements]
```

### Bug Fixes

```
Fix this issue: [describe problem]
Error message: [paste error]
File: @[file path]
Expected behavior: [what should happen]
Current behavior: [what actually happens]
```

---

## Troubleshooting with AI

### Common Scenarios

#### Build Error - Missing Import

**Error:** `Module not found: Can't resolve '@/components/ui/button'`

**Prompt:**

```
Fix this import error in @app/page.tsx:
"Cannot find module '@/components/ui/button'"

Check if the file exists and fix the import path.
```

#### TypeScript Error

**Error:** `Type 'string' is not assignable to type 'number'`

**Prompt:**

```
Fix the TypeScript error in @[file]:
"Type 'string' is not assignable to type 'number'"

The error is on line [X]. Convert the value properly.
```

#### Styling Issue

**Problem:** Component not responsive on mobile

**Prompt:**

```
The hero section looks bad on mobile (under 768px).
Current issues: [describe]

Fix the responsive design using Tailwind breakpoints.
```

#### Deployment Issue

**Problem:** Vercel build fails

**Prompt:**

```
The Vercel deployment is failing with:
[paste build log]

Identify the issue and provide the fix.
```

---

## Example: Complete Project Walkthrough

### Project: Digital Agency Website

#### Step 1: Setup

**Prompt:**

```
Create a new Next.js project called "digital-agency" with:
- TypeScript
- Tailwind CSS
- shadcn/ui
- App Router
- Dark mode support
- Place it in ~/projects/
```

#### Step 2: Home Page

**Prompt:**

```
In the digital-agency project, create a home page with:
1. Navigation with logo and links (Home, Services, About, Contact)
2. Hero section:
   - Headline: "We Build Digital Experiences"
   - Subheadline about web development services
   - CTA button linking to /contact
   - Animated background
3. Features section with 3 cards:
   - Web Design
   - Development
   - Marketing
4. Testimonials slider with 3 quotes
5. Footer with contact info and social links
```

#### Step 3: Services Page

**Prompt:**

```
Create a services page at /services with:
- Page title and description
- 6 service cards in a 3x2 grid:
  1. Web Design (icon: Palette)
  2. Development (icon: Code)
  3. SEO (icon: Search)
  4. Marketing (icon: TrendingUp)
  5. Branding (icon: Brush)
  6. Support (icon: Headphones)
- Each card: icon, title, description, learn more link
- Consistent styling with home page features
```

**Step 4: About Page**

**Prompt:**

```
Create an about page with:
- Company story section with text and image
- Mission statement
- Team section:
  - 4 team members
  - Circular photos
  - Names and roles
  - Social media links
- Timeline of company milestones
- Values section with icons
```

**Step 5: Contact Page**

**Prompt:**

```
Create a contact page with:
- Contact form:
  - Name (required)
  - Email (required, validated)
  - Phone (optional)
  - Message (required, textarea)
  - Submit button
- Contact information sidebar:
  - Address
  - Phone
  - Email
  - Business hours
- Map placeholder (we'll add real map later)
- Form validation with error messages
- Success state after submission
```

**Step 6: Polish**

**Prompt:**

```
Polish the entire website:
1. Add smooth scroll behavior
2. Add page transition animations
3. Create a loading component
4. Add SEO meta tags to all pages
5. Ensure all links work correctly
6. Test responsive design on all pages
7. Optimize images with next/image
```

**Step 7: Deploy**

**Prompt:**

```
Help me deploy this digital-agency website:
1. Initialize git repository
2. Create GitHub repo and push
3. Deploy to Vercel with GitHub integration
4. Configure build settings
5. Set up custom domain (optional)

Provide the exact commands to run.
```

**Cascade provides:**

```
# Git setup
git init
git add .
git commit -m "Initial commit"
gh repo create digital-agency --public --source=. --push

# Vercel setup
vercel
# Follow prompts to link to GitHub and deploy
```

---

## Key Takeaways

### The Vibe-Coding Mindset

1. **Describe, Don't Write** - Use natural language to describe what you want
2. **Iterate Rapidly** - Make small prompts, review, refine
3. **Trust but Verify** - AI accelerates, but you guide quality
4. **Learn Continuously** - Ask AI to explain its choices
5. **Focus on Product** - Spend more time on user experience, less on syntax

### AI Collaboration Levels

| Level | Your Role | AI Role |
|---|---|---|
| **Beginner** | Describe features | Writes all code |
| **Intermediate** | Design architecture | Implements components |
| **Advanced** | Set patterns & review | Assists with implementation |
| **Expert** | Strategic decisions | Accelerates execution |

### Getting Started Today

1. **Install Windsurf** - Download from codeium.com
2. **Open a project** - Any existing or new folder
3. **Open Cascade** - Press `Ctrl/Cmd + L`
4. **Type your first prompt** - "Create a simple landing page"
5. **Iterate** - Keep refining with follow-up prompts

6. **Deploy** - Push to GitHub, deploy to Vercel

---

## Resources

### Prompt Libraries

- Keep a file of prompts that worked well
- Share prompts with your team
- Document which prompts produce best results

### Learning Path

1. Start with simple component prompts
2. Progress to full page generation
3. Learn to debug with AI
4. Master the iteration cycle
5. Build complete projects independently

### Community

- **Windsurf Discord** - Share prompts and tips
- **GitHub Discussions** - Next.js and Vercel communities
- **Twitter/X** - Follow #vibecoding for inspiration

---

## Conclusion

Vibe-coding with Windsurf, GitHub, and Vercel transforms web development from manual coding to collaborative creation. By mastering AI prompting, you can:

- **Build faster** - Prototypes in hours, not days
- **Explore more** - Try ideas without technical barriers
- **Focus on value** - Spend time on user experience
- **Ship confidently** - AI-assisted code + automated deployment

**Your new workflow:**

```
Prompt → AI Generates → You Review → Iterate → Push → Auto-Deploy
```

Start your first AI-powered project today!

---

**Document Version:** 2.2 - AI & Vibe-Coding with Workspace Management and Planning **Last Updated:** March 2025