Vibe Coding
AI-Assisted Development

CSCI 571

Principles of Software Development

Marco Papa, Ph.D.

What is Vibe Coding?

> *"You just see stuff, say stuff, run stuff, and copy/paste stuff, and it mostly works."*
> — Andrej Karpathy, 2025

Vibe coding is collaborative approach where you describe what you want in natural language, and AI generates the code.

AI as Your Pair Programmer
Not replacement — collaboration

Describe → Generate → Test
Iterative refinement through prompts

Human Judgment Required
You validate, test, and refine

Setup: GitHub Copilot in Eclipse

✓ Prerequisite: Visual Studio Code is already installed

1

GitHub Student

education.github.com

2

VS Code Marketplace

Settings → Extensions

3

Search & Install

GitHub Copilot

4

Sign In

Click Copilot icon

Choose Your LLM

Use any of these for writing and refining your prompts:

ChatGPT-5
OpenAI

chat.openai.com

Gemini 3
Google

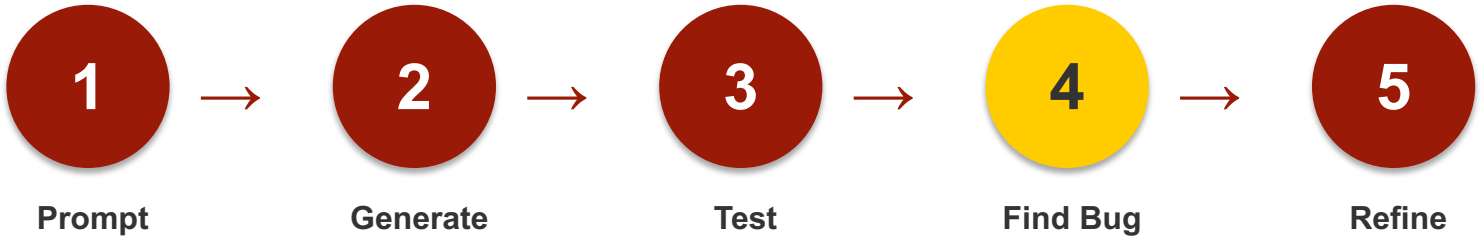
gemini.google.com

Claude Opus
Anthropic

claude.ai

We'll use a different LLM for each fix to show they all work!

The Vibe Coding Process



↻ Repeat until code handles all edge cases

Key Insight: AI gets the 'happy path' right, but YOU must identify edge cases.

EXAMPLE 1: SIMPLE CALCULATOR

Take two numbers and an operator, print the result

Calculator: Initial Prompt

THE PROMPT:

```
Write a Java program that takes two numbers
and an operator (+, -, *, /) from the user,
performs the calculation, and prints the result.
```

AI GENERATES:

```
double result = switch(op) {
  case '+' -> num1 + num2;
  case '-' -> num1 - num2;
  case '*' -> num1 * num2;
  case '/' -> num1 / num2;
};
```

TEST IT:

✓ **Happy Path Works!**

10 + 5 → 15.0 | 20 * 3 → 60.0

✗ **Bug Found!**

10 / 0 → Infinity (should show error!)

⚠ **More Edge Cases**

Invalid operator? Non-numeric input?

Next: Fix division by zero →

Calculator Fix 1: Division by Zero

ChatGPT-5

REFINED PROMPT:

```
The calculator returns 'Infinity' when dividing
by zero. Fix it to display an error instead.
```

AI ADDS THIS CHECK:

```
case '/' -> {
  if (num2 == 0) {
    System.out.println("Error: Divide by zero");
    yield 0;
  }
  yield num1 / num2;
}
```

TEST RESULTS:

✓ **Division by Zero Fixed!**

10 / 0 → Error message

✓ **Normal Division Works**

10 / 2 → 5.0

✗ **New Bug Found!**

10 % 3 → Exception! Invalid operators crash

Calculator Fix 2: Invalid Operator

Gemini 3

REFINED PROMPT:

```
The calculator crashes with invalid operators
like '%' or 'x'. Add a default case.
```

AI ADDS DEFAULT CASE:

```
double result = switch(op) {
  case '+', '-', '*', '/' -> ...
  default -> {
    System.out.println("Invalid op");
    yield 0;
  }
};
```

TEST RESULTS:

✓ **Invalid Operator Handled!**
10 % 3 → Error: Invalid op

✓ **Previous Fixes Work**
10 / 0 → Error message

✗ **Another Bug Found!**
'abc' + 5 → InputMismatchException!

Calculator Fix 3: Non-Numeric Input

Claude Opus

REFINED PROMPT:

```
The calculator crashes with InputMismatchException
when user types letters. Add try-catch.
```

AI WRAPS IN TRY-CATCH:

```
try {
  double num1 = sc.nextDouble();
  // ... rest of code ...
} catch (InputMismatchException e) {
  System.out.println("Enter valid numbers");
}
```

FINAL TEST RESULTS:

✓ Non-Numeric Input Handled!

'abc' + 5 → Error message

✓ All Previous Fixes Work!

Div by zero ✓ | Invalid op ✓

🎉 **Calculator Complete!**

3 prompts across 3 LLMs

EXAMPLE 2: PALINDROME CHECKER

Check if a word reads the same forwards and backwards

Palindrome: Initial Prompt

THE PROMPT:

```
Write a Java program that takes a word and
checks if it's a palindrome. Print the result.
```

AI GENERATES:

```
String word = sc.nextLine();

String reversed = new StringBuilder(word)
    .reverse().toString();

if (word.equals(reversed))
    System.out.println("Palindrome!");
```

TEST IT:

✓ Happy Path Works!

radar → Palindrome! | hello → Not

✗ Bug Found!

Radar → Not a palindrome (case matters!)

⚠ More Edge Cases

What about phrases with spaces?

Palindrome Fix 1: Case Sensitivity

ChatGPT-5

REFINED PROMPT:

```
'Racecar' shows as not a palindrome because
of uppercase R. Make it case-insensitive.
```

AI ADDS LOWERCASE:

```
String word = sc.nextLine()
    .toLowerCase();  // <-- NEW

String reversed = new StringBuilder(word)...
```

TEST RESULTS:

✓ **Case Sensitivity Fixed!**

Radar → Palindrome! | LEVEL → Palindrome!

✗ **New Bug Found!**

'A man a plan...' → Not (spaces break it!)

Palindrome Fix 2: Handle Spaces

Gemini 3

REFINED PROMPT:

```
The checker fails on phrases with spaces.
Remove spaces before checking.
```

AI REMOVES SPACES:

```
String word = sc.nextLine()
    .toLowerCase()
    .replaceAll(" ", "");  // <-- NEW
```

TEST RESULTS:

✓ **Spaces Handled!**

'A man a plan...' → Palindrome!

✗ **Another Bug Found!**

'Was it a car...?' → Not (punctuation!)

Palindrome Fix 3: Ignore Punctuation

Claude Opus

REFINED PROMPT:

```
The checker fails with punctuation marks.
Remove all non-letter characters.
```

AI USES REGEX:

```
String word = sc.nextLine()
    .toLowerCase()
    .replaceAll("[^a-z]", ""); // only a-z
```

FINAL TEST RESULTS:

✓ Punctuation Handled!
'Was it a car...?' → Palindrome!

✓ All Edge Cases Work!
Racecar ✓ | 'No lemon, no melon' ✓

🎉 **Palindrome Complete!**
3 iterative improvements

EXAMPLE 3: TEMPERATURE CONVERTER

Convert between Fahrenheit and Celsius

Temperature: Initial Prompt

THE PROMPT:

```
Write a Java program that takes a temperature
in Fahrenheit from the user and converts it
to Celsius. Print the result.
```

AI GENERATES:

```
double fahrenheit = sc.nextDouble();

double celsius = (fahrenheit - 32) * 5/9;

System.out.printf("%.2f°F = %.2f°C",
    fahrenheit, celsius);
```

TEST IT:

✓ **Happy Path Works!**

32°F → 0.00°C | 212°F → 100.00°C

✗ **Bug Found!**

'hot' → InputMismatchException crash!

⚠ **More Edge Cases**

What about Celsius to Fahrenheit?

Next: Handle invalid input →

Temperature Fix 1: Non-Numeric Input

ChatGPT-5

REFINED PROMPT:

```
The converter crashes when user types letters
instead of numbers. Add error handling.
```

AI ADDS TRY-CATCH:

```
try {
  double fahrenheit = sc.nextDouble();
  double celsius = (fahrenheit - 32) * 5/9;
  System.out.printf("%.2f°C", celsius);
} catch (InputMismatchException e) {
  System.out.println("Enter a valid number");
}
```

TEST RESULTS:

✓ Invalid Input Handled!

'hot' → Enter a valid number

✓ Normal Input Works

98.6°F → 37.00°C

⚠ Feature Request

Users want Celsius → Fahrenheit too!

Temperature Fix 2: Bidirectional

Gemini 3

REFINED PROMPT:

```
Users want to convert both directions.
Add a menu: F→C or C→F.
```

AI ADDS MENU:

```
System.out.println("1. F to C");
System.out.println("2. C to F");
int choice = sc.nextInt();
if (choice == 1)
  celsius = (fahrenheit - 32) * 5/9;
else
  fahrenheit = celsius * 9/5 + 32;
```

TEST RESULTS:

✓ F→C Works!

212°F → 100.00°C

✓ C→F Works!

0°C → 32.00°F

✗ Bug Found!

-500°F accepted (below absolute zero!)

Temperature Fix 3: Absolute Zero

Claude Opus

REFINED PROMPT:

```
The converter accepts impossible temperatures
below absolute zero (-459.67°F / -273.15°C).
Add validation to reject these.
```

AI ADDS VALIDATION:

```
final double ABS_ZERO_F = -459.67;
final double ABS_ZERO_C = -273.15;

if (fahrenheit < ABS_ZERO_F)
  System.out.println("Below absolute zero!");
```

FINAL TEST RESULTS:

✓ Absolute Zero Validated!

-500°F → Below absolute zero!

✓ All Previous Fixes Work!

Invalid input ✓ | Bidirectional ✓

🎉 **Temperature Converter Complete!**

3 prompts across 3 LLMs

Key Takeaways

1. **AI excels at the 'happy path'**
   Initial code works for typical inputs but fails on edge cases.
2. **Testing is your superpower**
   Always test with empty inputs, invalid data, and unusual cases.
3. **Iteration is the process**
   Specific prompts about observed problems lead to better fixes.
4. **Human judgment is irreplaceable**
   Understanding what to test requires your expertise.
5. **Any LLM can be your partner**
   ChatGPT, Gemini, Claude — pick one and learn to prompt effectively.

Top Coding Models

- Gemini 3 Pro
- Claude Opus 4.5
- GPT-5.2

Top Coding IDEs

IDE	Core Support	Extended Support	AI Plug-ins	Cost
Eclipse	Java		GitHub Copilot Pro, Windsurf, Tabnine, AssistAI	Free
JetBrains IntelliJ IDEA Ultimate	Java (1), Kotlin, HTML, CSS, YAML, SCSS, Less	Python (PyCharm), JavaScript and TypeScript (Webstorm)	JetBrains AI Assistant, GitHub Copilot Pro, Gemini Code Assist, Windsurf, CodeGPT, ProxyAI, Tabnine, Claude Code	Free (except Claude Code)
Microsoft VS Code	JavaScript, TypeScript, HTML, CSS	C++, C#, Python, Java, JSON, and many more	GitHub Copilot Pro, OpenAI Codex, Gemini Code Assist, Windsurf, CodeGPT, Tabnine, Claude Code	Free (except Claude Code)
Anysphere Cursor Pro	JavaScript, TypeScript, HTML, CSS, Python, Java, C#, PHP, Ruby, Swift, Kotlin, YAML, Terraform, Docker		No plug-ins needed; OpenAI Codex	Free (for 1 year)
Anthropic Claude Code	Java, Python, JavaScript	C++, C#, Ruby, Go, Rust, SQL, Swift, Kotlin	No plug-ins needed;	Free
Google Antigravity	Supports 22+ languages	Extensions through Open VSX Marketplace	No plug-ins needed; Supports Gemini 3 Pro & Flash, Claude Sonnet 4.5, Claude Opus 4.5, GPT-OSS (Open Source)	Free

Effective Prompting Basics

1. **Be Specific**
   State exactly what language, framework, and constraints you need.

2. **Provide Context**
   Explain the problem domain, inputs, and expected outputs.

3. **State Constraints**
   "No external libraries", "Must handle null input", "Under 50 lines".

4. **Show Examples**
   Give sample input/output pairs when behavior is complex.

5. **Ask for Explanation**
   "Explain each step" helps you learn, not just copy.

The Iterative Refinement Workflow

The Vibe Coding Cycle

1. DESCRIBE — Write a clear prompt explaining what you want
2. GENERATE — Let AI create the initial code
3. TEST — Run with normal inputs (the "happy path")
4. OBSERVE — Find edge cases that break it
5. REFINE — Write a specific prompt about the observed bug

Key Insight

Each iteration should address ONE specific problem.

Vague prompts like "fix all bugs" produce vague results.

Process Log Methodology

Why Document Your Process?

- Shows your reasoning and problem-solving approach
- Proves YOU identified the bugs and fixes (not just copied)
- Required for grading — 10% of assignment score

What to Include

- Each prompt you sent to the AI
- What you tested and what broke
- Your analysis of why it broke
- The refined prompt and result

Non-AI Users: Describe your development process similarly.

Process Log Example

Iteration 1

Prompt: "Write Java calculator..."

Test: 10 + 5 → 15.0 ✓

Test: 10 / 0 → Infinity ✗

Analysis: Division by zero not handled

Iteration 2

Prompt: "Fix division by zero..."

Test: 10 / 0 → Error ✓

Test: 10 % 3 → Crash ✗

Analysis: Invalid operator crashes

Iteration 3

Prompt: "Add default case..."

Test: 10 % 3 → Error ✓

Test: "abc" + 5 → Crash ✗

Analysis: Non-numeric input crashes

Iteration 4

Prompt: "Add try-catch..."

Test: "abc" + 5 → Error ✓

All edge cases handled!

4 iterations = complete solution

Vibe Coding for Web Projects

Multi-file development with HTML, CSS, JavaScript, and backend APIs

Multi-File Project Structure

Frontend Stack

index.html — Structure

styles.css — Presentation

app.js — Behavior

Backend Stack

Python/Flask or Node/Express

Acts as proxy to external APIs

Handles authentication, CORS

AI Handles Each Differently

HTML: Usually solid structure

CSS: Often needs layout fixes

JS: Watch for async issues

Prompting Strategy

Generate files separately

Reference other files by name

"In app.js, call the /api endpoint..."

Example 4: Responsive Landing Page

HTML + CSS + JavaScript with mobile breakpoints

Landing Page: Initial Prompt

PROMPT:

> Create an HTML landing page with a hero section, 3 feature cards, and a contact form. Include CSS for styling and make it responsive.

AI GENERATES:

```
<div class="hero">...</div>
<div class="features">
<div class="card">...</div>
</div>
/* CSS */
.features { display: flex; }
.card { width: 33%; }
```

TEST RESULTS:

✓ Desktop layout works!

✓ Cards side-by-side

✗ Mobile: cards overflow!

⚠ No media queries generated

Landing Page Fix 1: Mobile Breakpoints

Gemini 3

PROMPT:

> The feature cards overflow on mobile screens. Add CSS media queries to stack them vertically below 768px.

AI GENERATES:

```
@media (max-width: 768px) {
  .features {
    flex-direction: column;
  }
  .card {
    width: 100%;
    margin-bottom: 1rem;
  }
}
```

TEST RESULTS:

✓ Mobile: cards stack ✓

✓ Tablet: looks good

✗ Touch targets too small

⚠ Color contrast issues

Landing Page Fix 2: Accessibility

Claude Opus

PROMPT:

> The page has accessibility issues: buttons are too small on mobile, and text contrast is poor. Fix for WCAG 2.1 AA compliance.

AI GENERATES:

```
button, .btn {
  min-height: 44px;
  min-width: 44px;
  padding: 12px 24px;
}
body { color: #333; } /* 7:1 ratio */
a { color: #0056b3; } /* 4.5:1 */
img { alt="descriptive text"; }
```

TEST RESULTS:

✓ Touch targets 44px+ ✓

✓ Contrast ratios pass

✓ Alt text on images

✓ Landing page complete!

Example 5: Flask API Proxy

Backend that proxies requests to external RESTful APIs

Flask Proxy: Initial Prompt

PROMPT:

Create a Flask API endpoint /api/weather that takes a city parameter and proxies the request to OpenWeatherMap API, returning JSON.

AI GENERATES:

```
@app.route('/api/weather')
def get_weather():
    city = request.args.get('city')
    url = f"{API_URL}?q={city}&appid={KEY}"
    resp = requests.get(url)
    return jsonify(resp.json())
```

TEST RESULTS:

✓ Valid city works!

✓ "London" → weather data

✗ "" (empty) → 500 error

✗ "<script>" → passes through!

Flask Fix 1: Error Handling

ChatGPT-5

PROMPT:

> The endpoint crashes with empty city or when the external API fails. Add proper error handling with appropriate HTTP status codes.

AI GENERATES:

```
@app.route('/api/weather')
def get_weather():
    city = request.args.get('city')
    if not city:
        return jsonify({"error": "Missing city"}),
400
    try:
        resp = requests.get(url, timeout=5)
        resp.raise_for_status()
        return jsonify(resp.json())
    except requests.RequestException as e:
        return jsonify({"error": str(e)}), 502
```

TEST RESULTS:

✓ Empty city → 400 ✓

✓ API timeout → 502 ✓

✗ XSS still possible!

Flask Fix 2: Input Validation

Gemini 3

PROMPT:

The city parameter is not sanitized. Add input validation to prevent injection attacks and limit to alphanumeric + spaces.

AI GENERATES:

```
import re

def sanitize_city(city):
    # Only letters, spaces, hyphens
    if not re.match(r'^[a-zA-Z\s\-]+$', city):
        return None
    return city.strip()[:100]  # Limit length

@app.route('/api/weather')
def get_weather():
    city = sanitize_city(request.args.get('city',
''))
    if not city:
        return jsonify({"error": "Invalid city"}),
400
```

TEST RESULTS:

✓ "<script>" → 400 ✓

✓ "New York" → works ✓

✓ Flask proxy complete!

Debugging with AI

What to Include in Debug Prompts

- The exact error message (copy/paste, don't paraphrase)
- The relevant code snippet (not the entire file)
- What you expected vs. what happened
- What you've already tried

Example Debug Prompt

"I'm getting 'TypeError: Cannot read property map of undefined'

on line 15 where I call data.results.map(). The API response

looks correct in the Network tab. Why might results be undefined?"

Code Review Workflows

Use AI to Review Your Code

- "Review this code for security vulnerabilities"
- "What edge cases am I missing?"
- "How can I make this more readable?"
- "Is there a more efficient approach?"

Before Submitting

- Ask AI to explain any code you don't fully understand
- Request alternative implementations to compare
- Have AI check for common mistakes in your language

AGENTS.md Introduction

What is AGENTS.md?

A file in your project that tells AI assistants how to work with your codebase.

Why It Matters

- Gives AI context about your project structure
- Defines coding conventions and patterns to follow
- Lists important files and their purposes
- Specifies testing and deployment requirements

Adopted by: Cursor, GitHub Copilot, VS Code, Devin, and 60,000+ projects

JavaScript Pitfalls AI Misses

async/await Issues

Forgetting await on async calls

Not handling Promise rejections

Race conditions in parallel calls

this Binding

this changes in callbacks

Arrow functions preserve this

bind() often forgotten

Type Coercion

"5" + 3 = "53" (not 8)

[] == false (but [] is truthy!)

Always use === not ==

Your Job

Test async flows manually

Check type conversions

Verify this in callbacks

CSS Specificity & Layout

Common AI CSS Issues

Overly specific selectors

Missing responsive breakpoints

Hardcoded pixel values

Z-index conflicts

What to Check

Test at multiple screen sizes

Verify in multiple browsers

Check for overflow issues

Validate color contrast

Flexbox/Grid Gaps

Often generates older syntax

May miss gap property

Alignment issues on Safari

Best Practice

Ask AI to use CSS variables

Request mobile-first approach

Specify browser support needs

Backend Patterns for GCP

Python/Flask

requirements.txt for dependencies

app.yaml for App Engine config

Environment variables for secrets

gunicorn for production

Node.js/Express

package.json with start script

Use PORT from env variable

Handle graceful shutdown

CORS configuration

Prompt Tip

"Generate Flask app ready for Google Cloud App Engine with proper configuration files"

Deployment Context

Tell AI your target platform

"This will run on Cloud Run"

"Configure for App Engine flex"