



VIBE CODING

Vibe On: Code Ignited

Feel the flow, code the glow.




Vibe Coding


Igniting Intuitive Coding


Vibe Coding - How it's different!





Vibe Coding


 **Start with a Compass, Not a Map** – Have a rough goal or vision before diving in. Direction matters more than detail.


 **Flow Over Formality** – Code in a stream-of-consciousness style; let ideas guide structure.


 **Prototype, Don't Perfect** – Build fast, test fast, break fast.

 **Energy-Led Development** – Let excitement and intuition lead your session, not a strict plan.


 **Ideas First, Refactor Later** – Prioritise expressing concepts before tidying up code.


 **Snap Pieces Together** – Like LEGO blocks: try components, swap them, remix them freely.


 **Ride the Momentum** – When it clicks, go deep. Don't stop to overthink.


 **Embrace the Unknown** – Explore, improvise, and find the solution along the way.


Rigid Coding (Plan-Driven)

 **Detailed Blueprint** – Begin with a complete plan before writing any code.


 **Strict Conformity** – Follow prescribed class designs, naming conventions, and documentation rules.


 **Deliver Only What's Specified** – No building without defined requirements and approved use cases.

 **Milestone-Led Work** – Prioritise timelines, checklists, and structured outputs.

 **Refactor As You Go** – Maintain clean, production-ready code from the outset.

 **Pre-Designed Modules** – Use components exactly as specified with minimal remixing.

 **Steady Progress** – Follow task lists regardless of creative flow.

 **Reduce Variance** – Avoid untested paths or speculative coding.

What do you need?



Tools

- ❖ **Fast, minimal editor:** VSCode, Sublime, or anything quick to open and use.
- ❖ **Terminal access:** For running scripts, tests, and debugging without leaving flow.
- ❖ **Auto-reload / hot-reload setup:** See changes instantly (e.g. nodemon, streamlit, or browser dev server).

Mindset

- ❖ **Loose goal:** Know roughly what you're building, not every detail.
- ❖ **Permission to be messy (initially):** Clean code comes later.
- ❖ **Bias to action:** Try first, analyse after.

Environment

- ❖ **Distraction-free time:** No meetings, no TikTok, no notifications.
- ❖ **Music or ambient sound:** Lo-fi, synthwave, or silence – whatever helps you zone in.
- ❖ **Coffee:** Lecturer recommendation.

Optional

- ❖ **Scratchpad** – Quick notes or sketches, digital or physical.
- ❖ **Previous snippets or playground repo** – So you can start experimenting immediately.
- ❖ **LLM or search tab open** – For frictionless lookup or prototyping.

What to Expect



Mindset

- ❖ Small projects can be done fast, especially with no backend. (e.g., Databases)
- ❖ Vibe coding focused and fluid, although initial mess is often tolerated.
- ❖ Start rough, improve as you go, structure comes with progress.
- ❖ Larger builds need more sessions, same method, more time.
- ❖ **Non-deterministic** - The same inputs/prompts will give different results the second time.

You Might Build

- ❖ A static site with:
 - Google Maps iFrame,
 - Weather API call and display,
 - GitHub repo set up and committed.
- ❖ Copilot helps with layout, fetch logic, and polish.

What It Feels Like

- ❖ Quick wins (e.g., “Map works!”).
- ❖ Creative and responsive.
- ❖ Occasional blockers (API keys, formatting).
- ❖ Progress guided by momentum, not a checklist.



GitHub Co-Pilot Agent Mode

**Code Smarter, Faster: GitHub Copilot
Agent Mode in VSCode**

Copilot Free with Visual Studio Code



Select user to authorize



GitHub Co-Pilot Features



Use Natural Language for Code Generation

- ❖ What it does: Converts plain English prompts into code.
- ❖ How to use: Open the Copilot Chat panel and type, for example ...
Create a function that fetches weather data from OpenWeatherMap.

Code Explanation

- ❖ What it does: Explains selected code in plain language.
- ❖ How to use: Select code → Right-click → Ask Copilot → Explain this code
- ❖ Or just type:
What does this function do?

Bug Detection and Fix Suggestions

- ❖ **What it does:** Analyses code and suggests how to fix errors.
- ❖ **How to use:** Ask:
Why is this function crashing?
Fix the bug in this code.

GitHub Co-Pilot Features



Test Case Generation

- ❖ What it does: Writes tests for selected functions.
- ❖ How to use: Select a function → Ask:
Write Jest tests for this function.



Code Refactoring

- ❖ What it does: Suggests or performs refactors to improve code.
- ❖ How to use:
Refactor this to be more readable.
Refactor this to use async/await.



Multi-file Awareness (Preview / Beta)

- ❖ What it does: Understands relationships across multiple files in your workspace.
- ❖ How to use:
Where is this function used?
How does routing work in this project?

GitHub Co-Pilot Features



API and Library Help

- ❖ What it does: Explains how to use third-party libraries or APIs.
- ❖ How to use:
How do I use the Google Maps API in JavaScript?

Project Scaffolding


- ❖ What it does: Creates starter code or full project structure based on a description.
- ❖ How to use:
Scaffold a React app with Tailwind and a weather API.


Prompt Iteration


- ❖ What it does: Lets you ask follow-ups and evolve code through natural dialogue.
- ❖ How to use:
Now add input validation.
Now write a CSS animation for the loading spinner.


Lecturer Demo




 **Prepare 1:** Create a new directory called `dev-c240`. Open that directory in VSCode.

 **Prepare 2:** Start VSCode and start a new terminal. Minify the debug console. Check `git`, `node` and `npm` are installed.

 **Prepare 3:** Open Github Copilot Chat, select Agent mode and Model to Claude Sonnet 4.

 **Prompt 1:** Create a new Node.js project called ``demo_node`` by running ``npm init -y``, then add a basic ``index.js`` file with a placeholder function and comments. Create a ``.gitignore`` file with typical Node.js exclusions. Do not install any extra packages yet.

 **Prompt2:** Initialise this project folder as a Git repository, make an initial commit with all files, then guide me to create a new GitHub repository called ``demo_node``, add it as the remote origin, and push the commit to GitHub.

Read, Don't just click



- ❖ This message (or others like it) may appear.

2. Add GitHub Repository as Remote Origin

After creating the repository, GitHub will show you the setup instructions. You'll need to copy your repository URL and run these commands:

```
✓ cd c:\dev-c240\demo_node &&  
  echo "Ready to add remote  
  origin. Please replace  
  YOUR_USERNAME with your  
  actual GitHub username:"
```

Please replace `YOUR_USERNAME` with your actual GitHub username and run this command:

```
git remote add origin https://gi
```



Play Quick GitHub Co-Pilot Demo

Vibe Coding

Planning

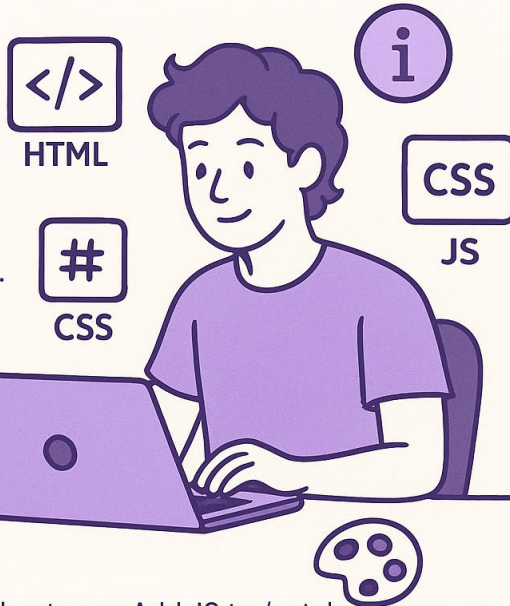


Why you still need a plan



Vibe Coding Plan

- 🔍 **Clarify & Configure**
 - AI Kickstart: Prompt Copilot/ChatGPT...
 - Review & Commit: Verify file structure,...



- ✂️ **Prototype & Flow**
 - Stub UI: Hardcode a simple HTML form...
 - AI Assist: Accept Copilot suggestions...

- 🔊 **Integrate & Validate**
 - Real API Swap: Replace demo data...
 - AI-Test Prompt: "Write JS code to..."

- ⌘ **Refine & Ship**
 - Style It: Tweak CSS for layout...
 - Robustness: Add JS try/catch,...

- ❖ Vibe Coding is **anti-rigid waterfall thinking**, however it is not ***without* thinking!**
- ❖ Vibe Coding is **creative, exploratory** and **non-linear**
- ❖ Vibe Coding can **benefit** from a suitable Product Requirements Documents (**PRD**). Such a PRD has been supplied.
- ❖ The PRD given
 - **Sets the direction** without rigidity
 - **Light-weight, gamified and modular** catering for human-in-the-loop approach
 - Includes **personas, success metrics and core features** making it easy to build prompts at each stage
 - Encourages **creative flow** while **staying grounded**.

How to use the PRD



Your Mission

Use it to define your North Star.

When vibe coding, you don't need a rigid spec, but you do need ***purpose***. This section gives you just that: a name, a one-liner, and a reason your app matters. Refer back when decisions get fuzzy.

Objectives & XP

Set loose goals to guide iteration.

The "Objectives" and "Success Metrics" tell you what ***matters most*** (e.g., fast load times, uptime). Keep these in mind as you build, but don't let them block early creativity. Validate them later in testing or deployment.

Who's Playing

Build empathy, not bureaucracy.

Personas help you ***vibe with your users***. You don't need 20-page user journeys, just a quick mental model of who benefits. Use this to guide UI design, language tone, and feature priorities.

How to use the PRD



Must-Haves & Bonus Rounds

Start minimal, grow naturally.

These act as a lightweight feature list. You can hardcode or prototype core features first, then decide if and when to explore the bonus rounds, based on how the build is going.

Team & Tools

Get into flow quickly.

Everything here, VS Code, Copilot, Live Server, is tailored for fast feedback. Use this to set up your working environment in minutes, not hours. It's designed for exactly the kind of iterative loop vibe coding thrives on.

How to use the PRD



Vibe Coding Plan

This section explicitly outlines the vibe workflow:

- *Clarify & Configure* with AI prompts
- *Prototype & Flow* with stubs and Copilot
- *Integrate & Validate* with real APIs
- *Refine & Ship* based on feel and polish

Each phase is both a **suggestion** and a **launch point** for Copilot prompts. You're not following steps—you're riding the rhythm.



Milestones & Level Ups

Structure without stress.

Use these weekly markers as **loose rhythms**, not deadlines. They help you pace the project and avoid last-minute scrambles, while staying flexible and responsive.



Test It!

AI-aided confidence checks.

Let Copilot or ChatGPT help you write test plans and validations. Again—exploratory, fast, just enough to build trust in your output.



Co-Pilot Vibe

Activity



Activity: Vibe Coding



❖ Make sure you have installed git, node, npm →

❖ Read through how **we used** the **PRD**

PRD Shore Squad.docx

This document shows

- **Prompts Used**
- **Results Obtained**
- **CoPilot Output and Handling**

❖ If you get stuck - there is a **Vibe Coding Hints** document. Try reading it and see if you can solve the issue for yourself.

❖ **Live Demo Site** is here: <https://codeblazar.github.io/ShoreSquad/>

❖ Site Features:

- Google Map iFrame Inclusion
- Live Weather from NEW

```
C:\dev-c240 > git --version  
git version 2.48.1.windows.1
```

```
C:\dev-c240 > node --version  
v22.15.0
```

```
C:\dev-c240 > npm --version  
10.9.2
```





Lesson 13

Summary of Material Covered

Lesson 13 Review



- ❖ **Vibe Coding** is a key theme. It's an alternative to rigid, plan-driven coding.
- ❖ Encourages **creativity, exploration, and flow-based development**.
- ❖ Emphasizes a **lightweight Product Requirements Document (PRD)**:
 - Provides direction without rigid specs.
 - Includes personas, success metrics, and core features.
 - Supports creative prompt-building with structure.
- ❖ A **"plan" is still needed**, but it should be flexible and purpose-driven.
- ❖ Encourages using **Copilot and AI tools** to assist with prototyping, testing, and refining.
- ❖ Promotes **momentum-driven iteration** instead of strict milestones.

Thank you

School of Infocomm

C240 AI Essentials and Innovations

© Republic Polytechnic 2025: All Rights Reserved