



# Pair programming with AI



by Jarosław Michalik



# Jarosław Michalik

Founder

@ theimpl.com

@ trainingpro.app

@ nowoczesny.dev

@ AndroidPro.io

Google Developer Expert / Android

# 1950 - 1970

1950-1970

**coding == science**

**and military**



70'

COBOL -> business dev



# 80's

"Computer in every office"

(at least in US)



# 90' early 00'

.com bubble

“programmer” becomes separate profession

```
length, iN;
dblTemp;
ain = true;

again) {
= -1;
in = false;
line(cin, sInput);
stem("cls");
ingstream(sInput) >> dblTemp;
length = sInput.length();
(iLength < 4) {
again = true;
continue;
else if (sInput[iLength - 3] != '.') {
again = true;
continue;
while (++iN < iLength) {
if (isdigit(sInput[iN])) {
continue;
else if (iN == (iLength - 3)) {

```

# 2010'

specialization era

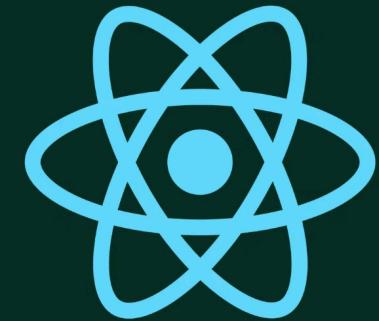
Backend

Frontend

Mobile

Data Science

Devops



# building software is becoming more and more accessible

## 1 Low-level Coding Era

Engineers worked directly with machine-level instructions.

## 2 Frameworks Era

Libraries and frameworks accelerated development.

## 3 Abstraction Era

No code tools, automation platforms.

## 4 AI Code Generation

"We are here" - AI now generates code from descriptions.



# As tech advances...



A screenshot of a Reddit post from the subreddit r/ProgrammerHumor. The post has 3,5k upvotes and 42 comments. The top comment, by a user with a green profile picture, says: "Well... you can try html and js. It's not really coding, everyone here will agree." A reply to this comment, by a user with a black profile picture, says: "Oh javascript isn't coding now either? You're not allowed in the club if you cant write your own machine learning algorithm from scratch in x86 assembly huh?" Below the replies, there is a button labeled "Say Happy Cake Day!"

← r/ProgrammerHumor 📒 ⋮

↑ 3,5k ↓    42 Share ⚡

⋮ ⚡ Reply ↑ 1 ↓

Well... you can try html and js. It's not really coding, everyone here will agree.

⋮ ⚡ Reply ↑ -38 ↓

Oh javascript isn't coding now either? You're not allowed in the club if you cant write your own machine learning algorithm from scratch in x86 assembly huh?

Say Happy Cake Day!

⋮ ⚡ Reply ↑ 55 ↓

# And we will always have gatekeepers.

Ignore them.

# AI assisted coding is not "real" programming!

Who cares?. Let's build stuff.

# GPT 3 - open beta

Model

gpt-3.5-turbo

text.format: `text` temp: `1.00` tokens: `2048` top\_p: `1.00`

store: `true`

# gpt-3.5-turbo

We can ask for code snippets.

Context window is small.

Copy/paste

**Format:** + followed by 1–15 digits

```
r
```

 Copy code

```
^\+[1-9]\d{1,14}$
```

## ✓ Valid examples

```
diff
```

 Copy code

```
+14155552671
```

```
+919876543210
```

```
+447911123456
```

## ✗ Invalid examples

```
scss
```

 Copy code

```
14155552671 (missing +)
```

```
+0123456789 (country code cannot start with 0)
```

```
+12345678901234567 (too long)
```

# ChatGPT

## News Viewmodel

Your ViewModel has been created. If you'd like to add dependency injection (Hilt/Koin), expose loading states, or include more RxJava operators (switchMap, debounce, etc.), just let me know!

🔗 ↻ ⚡ ...

- possibility to ask follow up question
- "Stack Overflow As A Service"

Ok, but model state with StateFlow

News Viewmodel · other

Copy Edit Download

```
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.launch
import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.disposables.CompositeDisposable
import io.reactivex.schedulers.Schedulers

class NewsViewModel(private val newsRepository: NewsRepository) : ViewModel() {

    private val disposables = CompositeDisposable()

    private val _newsState = MutableStateFlow<List<News>>(emptyList())
    val newsState: StateFlow<List<News>> = _newsState.asStateFlow()
```

# Claude Sonnet 3.7

Business Process Mapping Consultation

- Elementy dodatkowe z możliwością określenia ilości (szuflady, półki)
- Warunki montażu ze współczynnikami trudności
- Optymalizacja zespołu monterów
- Automatyczny bufor bezpieczeństwa (+15%)

Designerskie elementy:

- Nowoczesny, responsywny design z gradientami
- Animacje i efekty hover dla lepszego UX
- Wskaźnik procesu podczas obliczeń
- Szczegółowe wyjaśnienie kalkulacji
- Intuicyjny interfejs z ikonami

Logika biznesowa:

- Bazowe czasy: Szafa (6h), Kuchnia (12h), Łazienka (8h), Inne (4h)
- Współczynniki warunków: 1.0-1.5x
- Optymalizacja zespołu: 1-3+ monterów
- Automatyczne obliczenia w czasie rzeczywistym

Aplikacja jest gotowa do użycia i można ją łatwo dostosować do konkretnych potrzeb warsztatu meblarskiego.

Reply...  
Sonnet 4.5

Claude is AI and can make mistakes. Please double-check responses.

### Kalkulator Czasu Montażu

Precyzyjna kalkulacja czasu montażu mebli u klienta

**Typ mebla**  
Wybierz kategorię mebla:  
-- Wybierz typ --

**Elementy dodatkowe**

Szuflady (2h/szt)    Półki regulowane (0.5h/szt)  
 Oświetlenie LED (3h)    Uchwyty/Klamki (1h)

Liczba szuflad:

Liczba półek regulowanych:

**Warunki montażu u klienta**

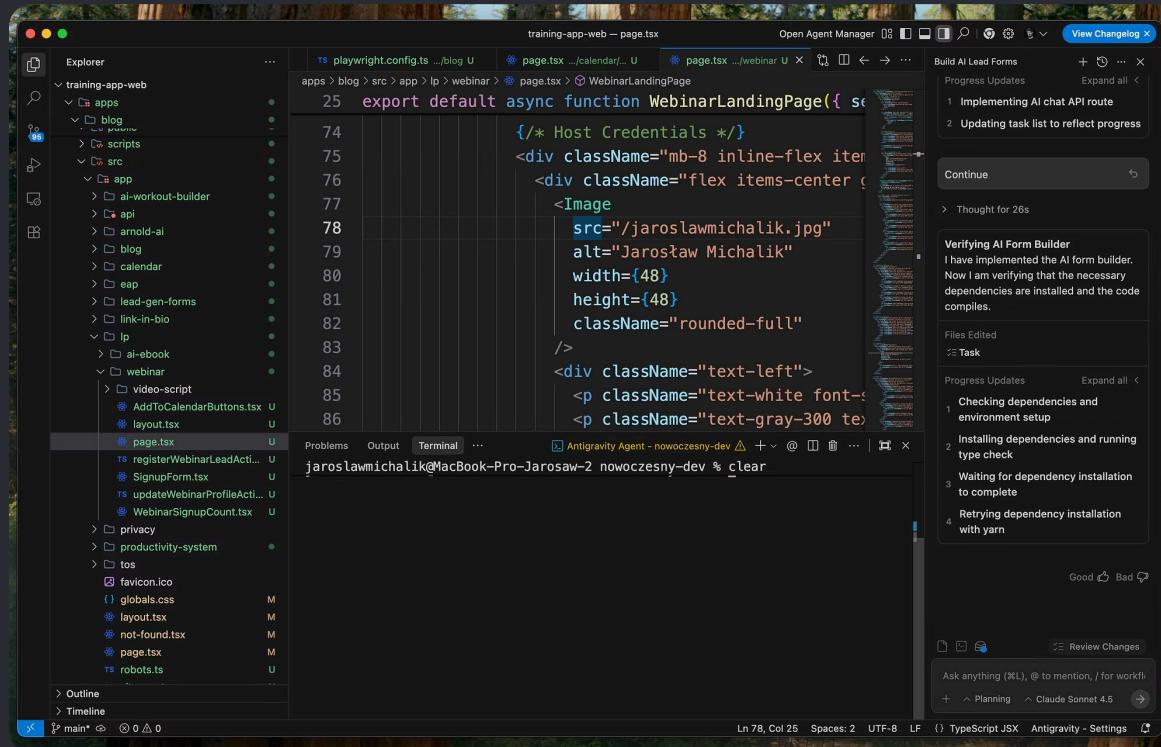
Standardowe  
Dobry dostęp,

Trudne  
Ograniczony

Bardzo trudne  
Wielkie schody

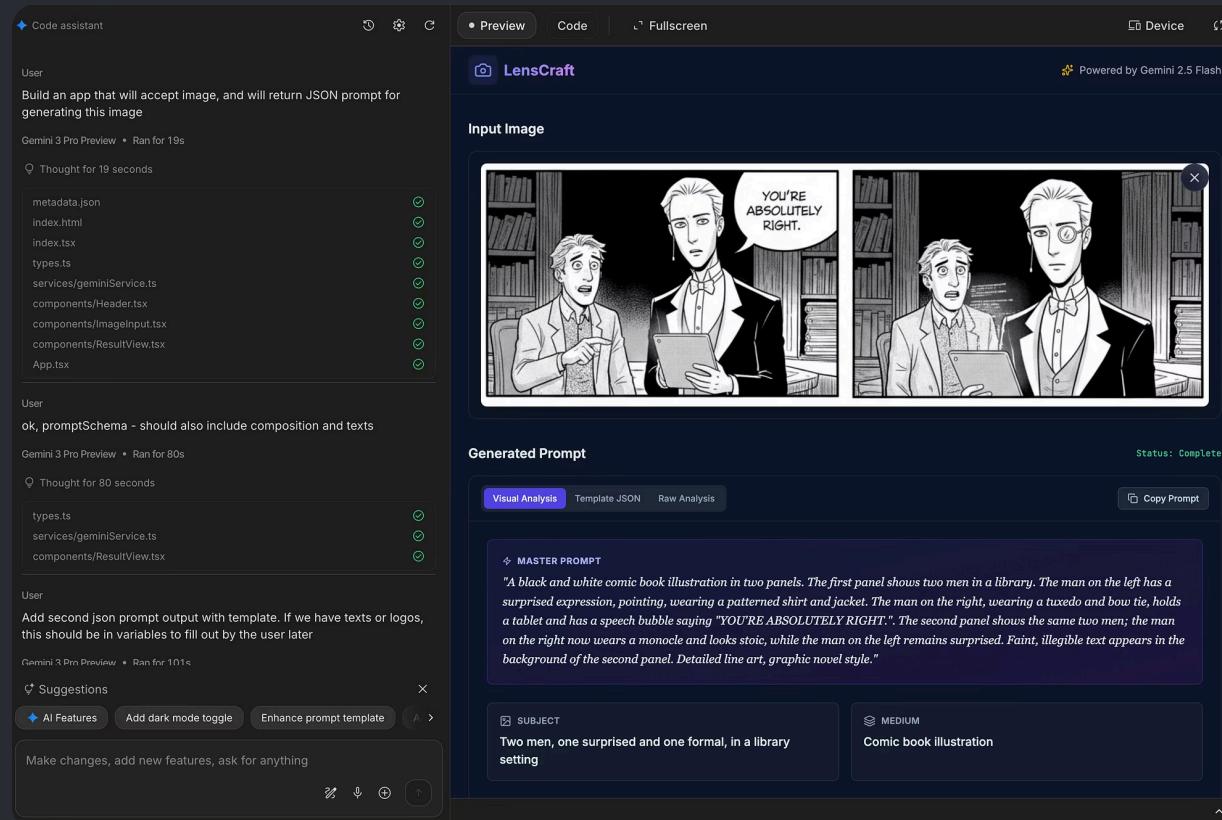
- best coding model (at the time)
- built-in artifacts

# AI-native IDEs (Cursor, Antigravity)



- Integrated Dev Environment
- Built-in coding agent

# Vibe coding tools



## Tools available in browser

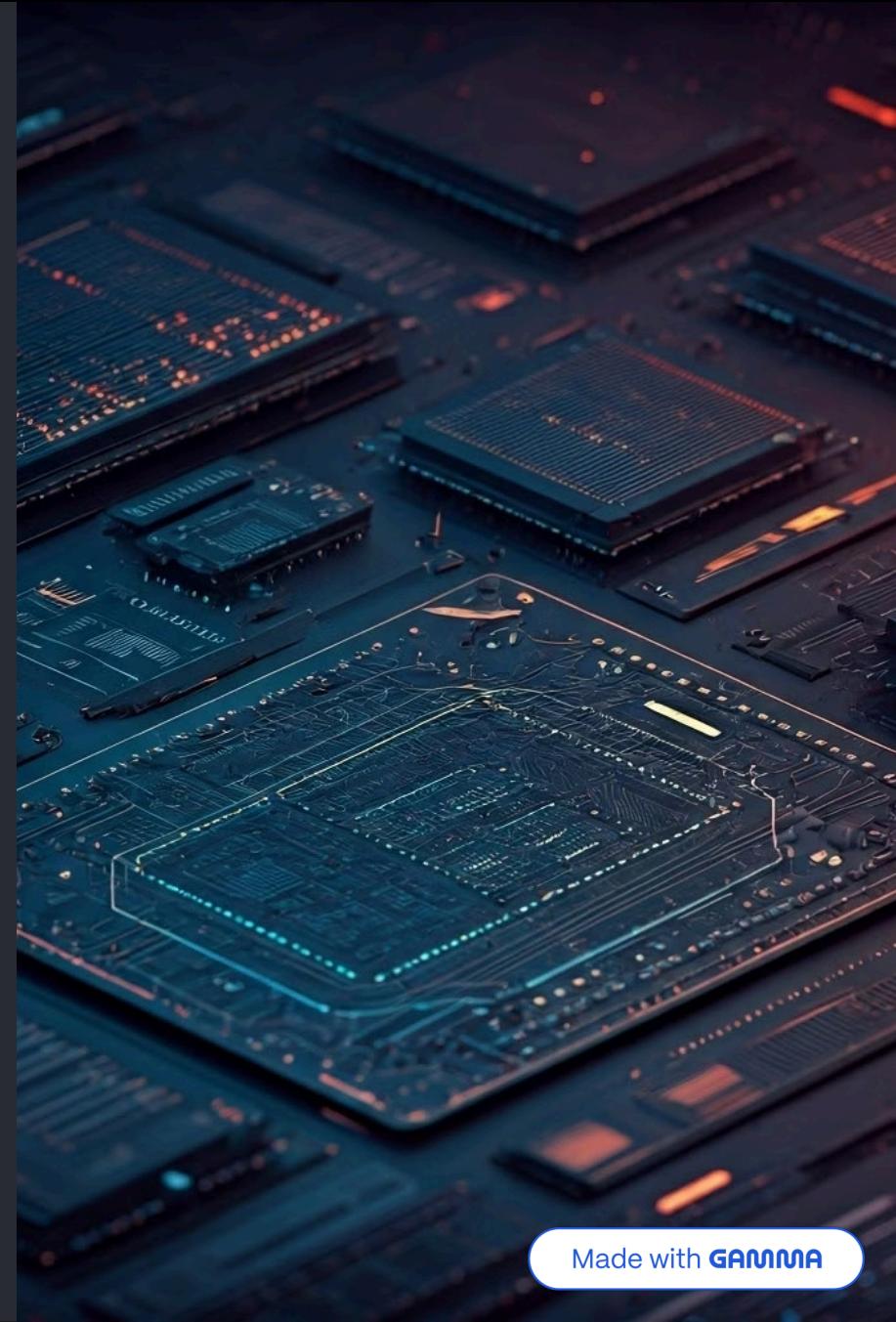
"Describe your idea with prompt"

- Google AI Studio
- Lovable
- bolt.new
- DreamFlow

**Tools are here. How to use them wrong?**

# Bad Practices - Regardless of AI model

Poor software engineering habits persist whether you're writing code or prompting AI. These anti-patterns undermine quality regardless of your tools.



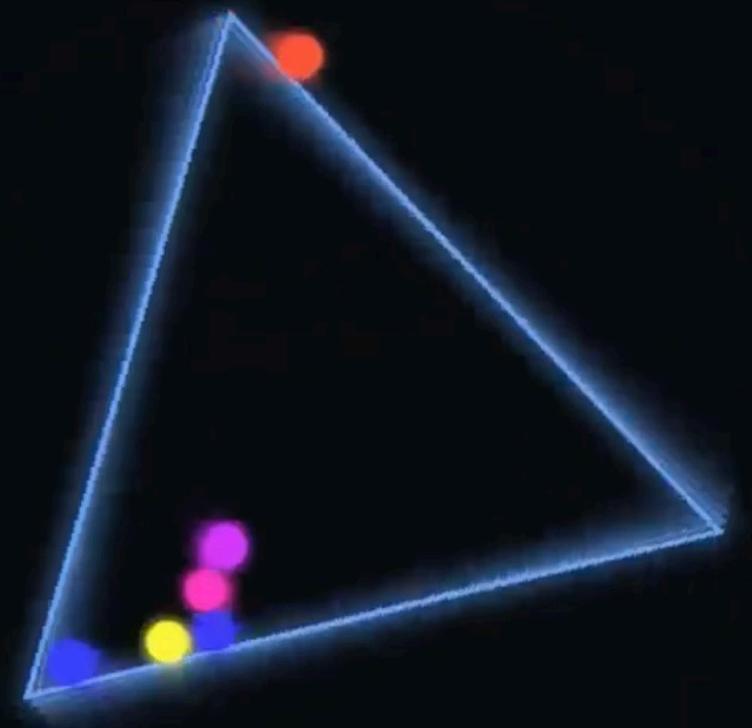
# bad practice #1 - trying to one-shot the whole app

You wouldn't ask freelancer to "build an app like Facebook" without any additional requirements.

**Don't ask LLMs to do that either.**

You won't like the result.

Work incrementally.



# bad practice #2 - building only in general-use assistant

Claude, Gemini have "Canvas" mechanism with interactive code preview. That's great for prototyping.

It's amazing for fast prototypers, but it's not sustainable as your codebase grows.

The screenshot shows the Gemini AI interface. On the left, there's a dark panel with text explaining a landing page with a form built using React. It highlights the use of Tailwind CSS for styling, Framer Motion for animations, functional components, and form handling with validation. A code editor at the bottom shows a snippet of React code for a contact form. On the right, the main interface displays a landing page titled "Launch Your Dreams" with a purple header and a "Key Features" section containing "Innovative Solutions" and "Dedicated Support". Below that is a "Community Driven" section and a "Get in Touch" form for name and email input.

**Gemini**  
2.0 Flash

**Explanation and Key Improvements:**

- **Modern UI with Tailwind CSS:** The code uses Tailwind CSS for styling, creating a clean, modern, and responsive design. It's integrated with shadcn/ui components for a polished look.
- **Framer Motion Animations:** Smooth animations are added using Framer Motion to enhance the user experience. Elements like the header and sections fade in and slide up, creating a dynamic feel.
- **Clear Component Structure:** The code is organized into functional components (`LandingPage`, `ContactForm`) for better maintainability.
- **Form Handling with Validation:** The `ContactForm` component manages form state using `useState` and includes a basic `handleSubmit` function. The form includes placeholders and is set to require input.
- **Loading State:** The `handleSubmit` function now includes a simulated loading state (using `setTimeout`) to demonstrate how you might handle asynchronous form submission (e.g., sending data to a server). The submit button is disabled and shows a loading icon.

Wpisz prompt dla Gemini

+ Canvas

Nie używamy Twoich czatów, które prowadzisz jako BM Solutions, do ulepszania naszych modeli. Gemini może popełniać błędy, także co do ludzi, dlatego sprawdzaj odpowiedzi twój prywatność! Gemini

Landing Page with Form (React)

Przejdz Kod Podgląd

Launch Your Dreams

We provide the tools and resources to help you achieve your goals. Join our community and take your first step towards success.

**Key Features**

**Innovative Solutions**  
We offer cutting-edge solutions tailored to your needs.

**Dedicated Support**  
Our team is here to support you every step of the way.

**Community Driven**  
Join a vibrant community of like-minded individuals.

**Get in Touch**

Name  
Your Name

Email

# bad practice #3 - delegating system understanding

When you don't know what you're actually, you can introduce vulnerabilities you're now aware of



leo

@leojr94\_

...

my saas was built with Cursor, zero hand written code

AI is no longer just an assistant, it's also the builder

Now, you can continue to whine about it or start building.

P.S. Yes, people pay for it

4:34 am · 15 Mar 2025 · 52.2K Views



leo

@leojr94\_

...

guys, i'm under attack

ever since I started to share how I built my SaaS using Cursor

random thing are happening, maxed out usage on api keys, people bypassing the subscription, creating random shit on db

as you know, I'm not technical so this is taking me longer than usual to figure out

for now, I will stop sharing what I do publicly on X

there are just some weird ppl out there

9:04 am · 17 Mar 2025 · 53.6K Views

## bad practice #4 - expecting the magic

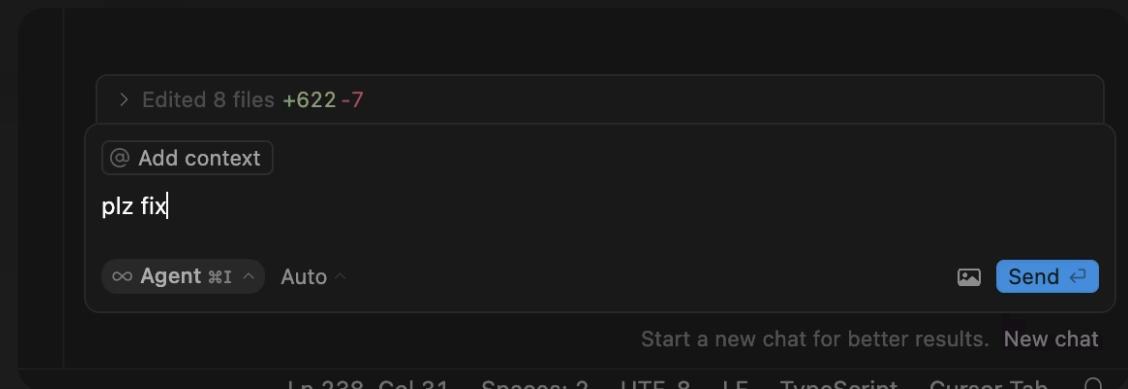
If you know EXACTLY what you want, with AI you'll get it faster.

If you don't know what you're doing, results are unpredictable.



# bad practice #5 - giving vague instructions

Sometimes it will work. Most often it won't.



**Bad practices come from expectation:**

**"AI will do it for me"**



# don't delegate your thinking to AI

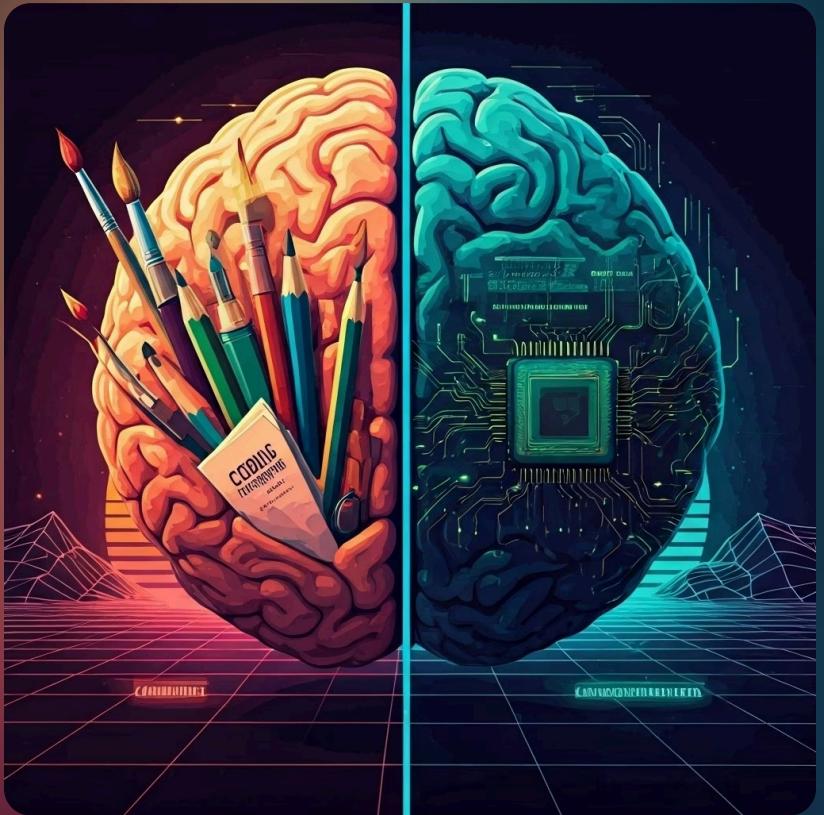
(even if model has "Thinking"/"Reasoning" mode

**core software craftsmanship practices +  
understanding how LLMs work → ✨**

# Good Practices - engineering craftsmanship

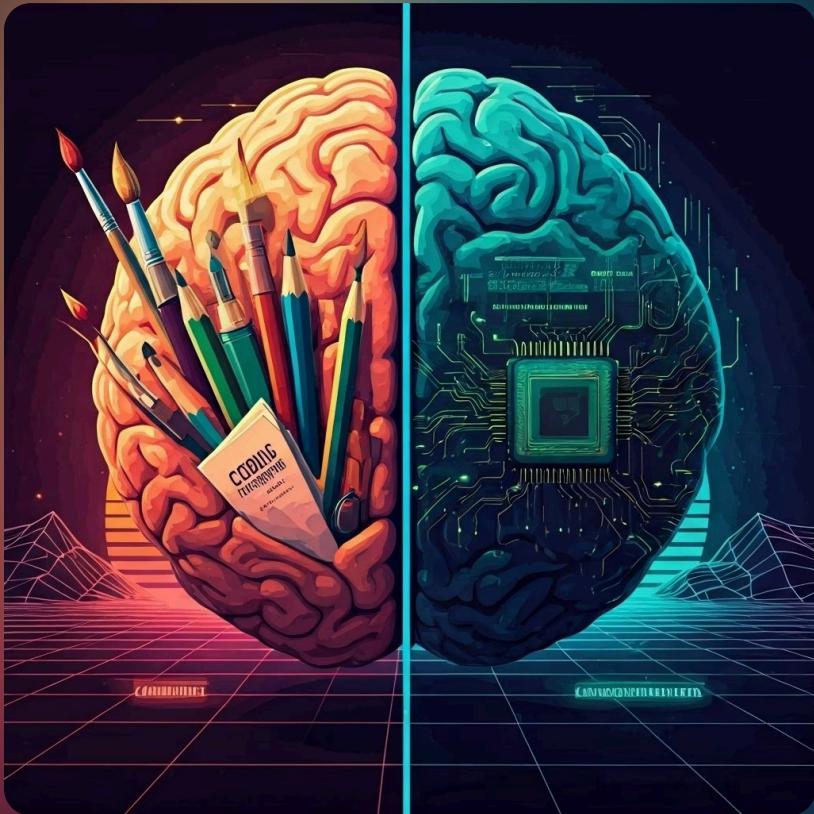
Strong engineering principles become even more vital in the age of AI-assisted development.





# Separate creative and coding process

- **Planning session:** what we will build
- **Implementation sessions:** let's build it.



# Separate creative and coding process

- **Dedicated mental space:** Each process gets full attention without context-switching penalties
- **Appropriate thinking modes:** Planning requires divergent thinking while implementation needs convergent thinking
- **Reduced cognitive load:** You don't have to simultaneously hold both the big picture and intricate details in mind

It's funny how "human" psychology applies to LLMs.



## Define requirements precisely.

If you won't provide requirements, model will assume that requirement exists.



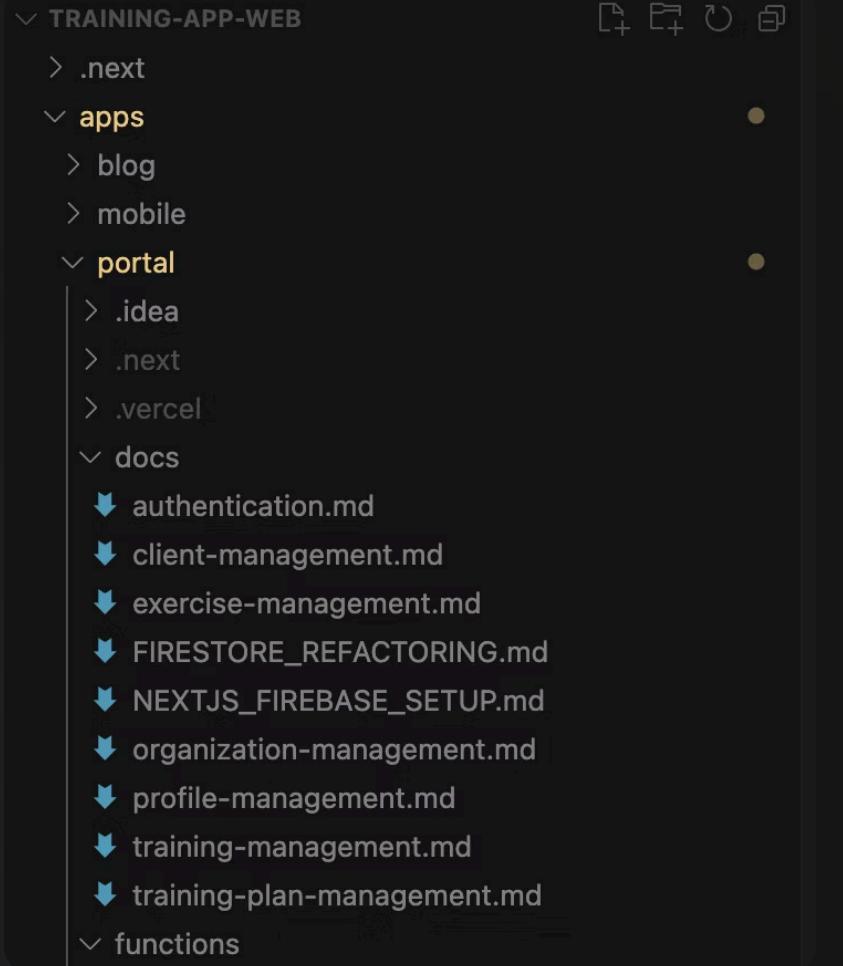
# Define requirements precisely.

"Build contact form"

vs

"Build customer contact form with name, phone, email and message and privacy policy checkbox. Email and privacy checkbox are required"

With 1st, you'll eventually get there. But why fix, if you can do it right from the beginning?



TRAINING-APP-WEB

- > .next
- ✓ apps
  - > blog
  - > mobile
- ✓ portal
  - > .idea
  - > .next
  - > .vercel
- ✓ docs
  - ⬇ authentication.md
  - ⬇ client-management.md
  - ⬇ exercise-management.md
  - ⬇ FIRESTORE\_REFACTORING.md
  - ⬇ NEXTJS\_FIREBASE\_SETUP.md
  - ⬇ organization-management.md
  - ⬇ profile-management.md
  - ⬇ training-management.md
  - ⬇ training-plan-management.md
- ✓ functions

# Build project docs.

- Document all functional requirements
- Define data structures
- Map out UI and user interactions
- Create a comprehensive schema of how everything works
- Create visual process maps if helpful

Keep it in repository. Cursor, Windsurf will be able to read those.

The screenshot shows a dark-themed code editor with multiple tabs at the top. The active tab is 'arnold-tools-testing-scenarios.md'. The content of the file is as follows:

```
1 # Arnold AI Tools Refactoring - Testing Scenarios
2
3 ## Overview
4 This document provides comprehensive testing scenarios to verify that
5 the Arnold AI tools refactoring was successful. The refactoring split
6 a 1938-line monolithic file into a modular, domain-driven
7 architecture.
8
9 ## Test Environment Setup
10
11 ### Prerequisites
12 - Arnold AI chat interface accessible
13 - Valid trainer authentication token
14 - Test clients and training programs in the database
15 - Gemini API key configured
16
17 ### Test Data Requirements
18 - At least 2 test clients with different statuses
19 - At least 1 tra
```

simple doc for coding agent

# What will be the output?

We have 3 people:

Jarek, he has blue eyes  
Natalie, she has green eyes  
Jarek, he has brown hair

What's the Jarek's hair color?



# What will be the output?

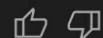
Works better for models with "Thinking" enabled

We have 3 people:

Jarek, he has blue eyes  
Natalie, she has green eyes  
Jarek, he has brown hair

What's the Jarek's hair color?

Jarek's hair color is brown.



is it possible that two people have name "Jarek"?

# Ultra-precise context: Product vs Product

```
// E-commerce context
package com.shop.domain
class Product(val id: String, var price: Double, var stock: Int) {
    fun decreaseStock(qty: Int) { stock -= qty }
}

// Manufacturing context
package com.factory.domain
class Product(val id: String, var blueprint: String, var status: Status) {
    enum class Status { DESIGN, PRODUCTION, DISCONTINUED }
    fun updateStatus(newStatus: Status) { status = newStatus }
}
```

# Ultra-precise context: Product vs Product

```
// Use Aggregate Roots with Context Prefixes
class CatalogProduct/* e-commerce attributes */
class ManufacturingProduct/* manufacturing attributes */
```

package name itself may not be enough for LLM

# Java Enterprise Edition

*Enterprisify your Java Class Names!*

*By [Hay Kranen](#)*

***RepositoryInstanceSchemaBroadcasterAuthenticationRemote-Callback***

[Enterprisify even more!](#)

[Again](#)

*[Fork this gist on Github to add your own words ;\)](#)*

<https://projects.haykranen.nl/java/>

# Domain-Driven Design with AI assistants

Domain-Driven Design principles become even more crucial when working with AI models. Clear bounded contexts decrease cognitive load for both humans and machines.

## Ubiquitous Language

Establish precise terminology that both development teams and AI models can understand consistently.

## Explicit Boundaries

Define clear domain separations to prevent context confusion in prompts and generated code.

## Strategic Modeling

Create domain models that guide AI toward appropriate solutions within specific contexts.

## Reduced Cognitive Load

Well-structured domains help both humans and AI focus on essential complexity rather than accidental complexity.

# Domain-Driven Design: Event Storming

Event Storming bridges the gap between domain experts and developers through collaborative modeling.



This lightweight, visual approach reveals hidden complexity and fosters shared understanding before coding begins.

# let's do this without the team.

Collaborative Process Mapping with Event Storming ▾

## Process Map

Based on the initial information, I'll create a basic payment flow process map that we can refine:

```
mermaid
graph TD
    User([User]) -->|Initiates payment| Frontend[Frontend App]
    Frontend -->|Submits payment request| Backend[Backend App]
    Backend -->|Sends payment to provider| PaymentProvider[Payment Provider]
    PaymentProvider -->|Payment status update| Webhook[Firebase Functions Webhook]
    Webhook -->|Stores payment data| Firestore[(Firestore)]
    Webhook -->|Notifies| Backend
    Backend -->|Updates status| Frontend
    Frontend -->|Shows confirmation| User
```

## Explanation

This initial process map shows a basic payment flow where:

1. The user initiates a payment through your frontend app
2. The frontend app sends the payment request to your backend
3. The backend communicates with the payment provider
4. The payment provider sends status updates via webhooks to your Firebase Functions
5. The webhook stores data in Firestore and notifies your backend
6. The backend updates the frontend with the payment status
7. The frontend shows a confirmation to the user

As we gather more specific information about your implementation, we can add more detail such as specific events (payment initiated, payment authorized, payment captured, etc.), commands (initiate payment, process webhook, update payment status), and the exact data flow between components. We'll also be able to identify potential error paths and how they're handled in your system.

Reply to Claude...

Claude 3.7 Sonnet ▾

The diagram illustrates the payment flow with the following components and interactions:

- User** initiates payment to the **Frontend App**.
- Frontend App** submits payment request to the **Backend App**.
- Backend App** sends payment to the **Payment Provider**.
- Payment Provider** sends payment status update to the **Firebase Functions Webhook**.
- Firebase Functions Webhook** stores payment data in the **Firestore** database.
- Firebase Functions Webhook** notifies the **Backend App**.
- Backend App** updates the **Frontend App** with payment status.
- Frontend App** shows confirmation to the **User**.

# let's do this without the team.

- prompt assistant to ask you questions
- ask it do create process diagram in mermaid
- work iteratively on this process

Put process diagram ifle in your docs

# Domain-Driven Design with AI assistants

## Naming Challenges

"Product" means different things in different contexts.

## Bounded Contexts

AI helps define clear boundaries between different domains.



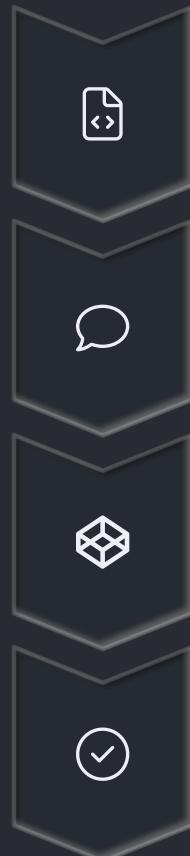
## Event Storming with AI

AI assistants can help identify domain events and aggregates.

## Ubiquitous Language

Consistent terminology between business and technical teams.

# Test-Driven Development in the AI Era



## Write Test

Define expected behavior first.

## Create Prompt

Describe implementation needs to AI.

## Generate Code

AI produces implementation based on prompt.

## Verify

Run tests to confirm correct behavior.

# Classic Test-Driven Development Cycle

While AI transforms software development, classic TDD principles remain foundational to quality code.

## Red: Write a Failing Test

Create a test that defines expected behavior before implementation exists.



## Refactor: Clean Up

Improve code design while ensuring tests still pass.



## Green: Write Minimal Code

Implement just enough code to make the failing test pass.



## Repeat: Iterate

Start the cycle again for new features or improvements.

# Manageable tasks:

## Do

- Create a Firestore collection for clients.
- Develop Next.js forms to add new clients.
- Implement functionality to read and display client data.
- Develop Next.js forms to edit existing client data.
- Implement functionality to delete clients.
- Implement client filtering and categorization logic using Firestore queries.

## Don't

- implement adding clients functionality

# Small Features and Incremental Development



## Break down complex tasks

Divide large problems into manageable pieces.



## Implement incrementally

Build one feature at a time.



## Validate continuously

Test each component before moving forward.

This approach works even better with AI assistance, as smaller prompts produce more accurate results.

# Split code into layers

AI assistants are eager to do lot's of changes. If you ask for UI change, you may end up with logic change.

If you're engineer - study Clean Architecture

If you're vibe coder - implement simple service layer

Separation creates maintainable, testable code.

- Easier AI prompting for specific changes
- Better testing isolation
- Simpler maintenance
- More reusable components

```
apps > portal > src > app > (auth) > training-plans > page.tsx > ...
26 export default function TrainingPlansPage() {
  // Apply filters when selection changes
  useEffect(() => {
    let filtered = [...programs];

    if (selectedClientId) {
      filtered = filtered.filter(program => program.client?.id === selectedClientId);
    }

    if (selectedTrainerId) {
      filtered = filtered.filter(program => program.trainerId === selectedTrainerId);
    }

    setFilteredPrograms(filtered);
  }, [selectedClientId, selectedTrainerId, programs]);
  const handleTypeChange = (type: string) => {
    const params = new URLSearchParams(searchParams.toString());
    params.set('type', type);
    router.push(`/training-plans?${params.toString()}`);
  };
  if (loading) {
```

# Split code into layers

## UI Layer

Handles only visual presentation and user interactions.

- Displays information to users
- Captures user input
- Manages visual state only
- No business rules or data processing

## Business Logic Layer

Contains all application rules and data processing.

- Implements core functionality
- Processes data independently
- Enforces business rules
- Remains UI-agnostic

# Use version control.

Commit small, meaningful changes.

"Added form validation", not "lots of changes".

Every you generate something that's working - make a commit.

The ability to go back a step is simply a matter of hygiene. Don't rely only on "Restore changes" functionality in IDE

Reddit post from r/cursor:

**Cursor f\*ck up my 4 months of works**

Question

Disclaimer, I'm a moron who worked on the same project without thinking about the risk that Cursor could break everything. Yesterday, Cursor (even though I only asked it to feed a view on my UI) destroyed months of development.

My question: How do you back up your projects/versions to ensure that the next action on cursor is reversible? Ops!

Also, I know that while I'm the concern, cursor isn't the only culprit, it's also Claude (while good overall) still has some flaws

Upvotes: 182 | Comments: 258 | Share

# Make use of multimodal input

- provide assistant with images for inspiration
- if your UI broke, share screenshot with what to fix

Again - human-like interactions. Text + image.



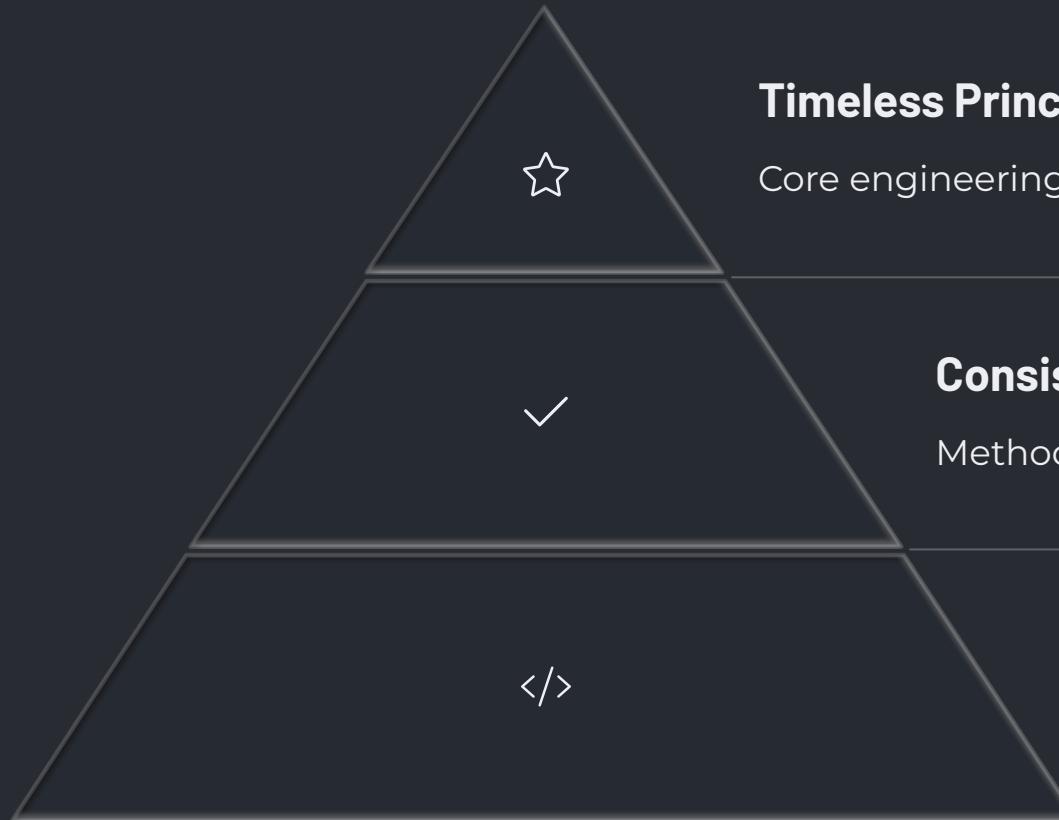
Here's the design I like. Build it with Tailwind:

❖ Okay, I can help you visualize how this Google search interface could be built using Tailwind CSS. Here's a breakdown of the structure and the Tailwind classes you might use:

HTML

```
<div class="bg-neutral-900 min-h-screen flex flex-col items-center justify-center">
  <div class="mb-8">
    <h1 class="text-white text-5xl font-light">Google</h1>
  </div>
  <div class="relative w-full max-w-xl">
    <div class="flex items-center rounded-full bg-neutral-800 py-2 pl-4 pr-2">
      <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="white" stroke-width="2">
        <path stroke-linecap="round" stroke-linejoin="round" d="M21 21l-5.197-5.197 M21 21l-5.197 5.197 M21 21l5.197-5.197 M21 21l5.197 5.197" />
      </svg>
      <input type="text" class="bg-transparent border-none outline-none text-white placeholder-transparent" />
    </div>
    <div class="flex items-center space-x-2">
      <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="currentColor" stroke="none" stroke-width="0">
        <path fill-rule="evenodd" d="M7 12a4 4 0 118 0 4 4 0 01-8 0zM14.229 13.29" />
      </svg>
      <button class="bg-transparent border-none outline-none text-white placeholder-transparent" type="button"><span>Search</span>
    </div>
  </div>
</div>
```

# Universal Software Engineering Principles



## Timeless Principles

Core engineering values transcend tools and technologies.

## Consistent Best Practices

Methodical application of standards ensures quality.

## Solid Foundations

Good design principles apply regardless of implementation method.

# Future of the Profession

## End of "Code Typists"

Pure implementation roles will diminish as AI handles basic coding tasks.

## Market Polarization

Industry will split between **specialists** with deep knowledge and **generalists** who orchestrate AI.

## New Key Competencies

Problem decomposition, prompt engineering, and system design become essential skills.





# Key Takeaways



## Universal Best Practices

Good engineering principles remain valuable regardless of tools used.



## AI as a Tool

AI enhances engineering but doesn't replace sound methodology.



## Responsible Vibe Coding

Blend responsible engineering practices with AI capabilities.

2025



DevFest Mediterranean 20...

2025

Pair Programming with AI

December 12, 2025

Event

Slides

[gdg.community.dev/events/det...](https://gdg.community.dev/events/details/B0D9E)



Beyond Code  
Anatomia Techno

2025

September 25, 2025

Bystra, Poland

Event

[schedule25.beyondcode.pl/](https://schedule25.beyondcode.pl/)



Future of European AI  
Responsible Vibe Coding

2025

May 6, 2025

Brussels

Event

[gdg.community.dev/events/det...](https://gdg.community.dev/events/details/B0D9E)



GDG on Campus WUT x Je...

2025

Supercharge your development with AI-  
driven coding tools

April 12, 2025

Warsaw

Event

[gdg.community.dev/events/det...](https://gdg.community.dev/events/details/B0D9E)



SFI IT Academic Festival

2025

Programmers Will Be Replaced by AI,  
Software Engineers Won't

March 27, 2025

Kraków

Event

[sfi.pl/pl](https://sfi.pl/pl)

get the slides at <https://michalik.tech/tech-talks>



# send me LinkedIn invite!

Jarosław Michalik [Add verification badge](#)  
Google Developer Expert | nowoczesny.dev | trainingpro.app | theimpl.com  
Cracow, Małopolskie, Poland · [Contact info](#)  
10,107 followers · 500+ connections

[Open to](#) [Add profile section](#) [Enhance profile](#) [Resources](#)

Tell non-profits you're interested in getting involved with your time and skills [X](#)  
[Get started](#)

[linkedin.com/in/jaroslawmichalik](https://linkedin.com/in/jaroslawmichalik)

<https://michalik.tech>

<https://theimpl.com>