# Pragmatic Programmer Tracer Bullets

**Section 12 Topic: Tracer Bullets**

**Why did you select this topic?**  
I picked this topic because I’ve worked on projects where myself and other co-workers have jumped into coding without seeing if their approach would even work. This approach is tempting because it allows you to quickly outpace or outperform your co-workers, but there’s a chance you can be misguided. Tracer bullets help guide the direction early on, which can save a lot of time and confusion later.

**Summary:**  
Tracer bullets are a way to build a thin working version of your application early in development, even if it's not complete. This lets you test the architecture, flow, and integration of parts like databases and UI. It’s like a prototype that works and keeps evolving, not a throwaway. The goal is to give feedback quickly and adjust direction as needed.

**References:**

* Page 50 through 55 in the programmer pragmatic book:
  + The two categories that I found the most appealing from the book are listed below. This differentiates a prototype from a tracer code approach.
  + I found this topic appealing because of the time saved and value add. As stated on page 52, For that initial build, all we could do was submit a query that listed all the rows in table, but it proved that the query UI could talk to the libraries, the libraries could serialize and unserialize the data.
  + Page 55, tracer code approach addresses a different problem. You need to know how the application hangs together. You want to show your users the interactions will work together in practice.
* Online Reference: <https://builtin.com/software-engineering-perspectives/what-are-tracer-bullets>
  + The article uses clear examples from development teams, showing how tracer bullets helped prevent wasted effort by proving architecture and catching flaws early.