■ Guardrails A to Z – Simple, Funny & Deep

■■■ *By: Hammad Bhai x GPT — Code Kings* ■

### ■ Chapter List

## 1■■ ■ Guardrails ka Buniyadi Taaruf

What?
Guardrails = “AI ke liye if■then rules: agar yeh hua, toh wapis mat aana, yeh mat bolna, yeh mat karna.”
Why?
• Bina guardrails: AI se sensitive data leak, hallucinations, hate speech nikal sakte.
• Guardrails se: AI pe “invisible fences” lag jaate—safe, compliant, predictable responses.

## 2■■ ■ Real■Life Analogy: Road, Stadium & Home Security

1. **Road Guardrail**
• Car highway pe slip na kare—side rail car ko dubara road par le aaye. ■
2. **Stadium Barrier**
• Fans pitch par na utaren—barrier crowd ko safe distance pe roke. ■
3. **Home Security Alarm**
• Window khula toh alarm baj jaaye—unauthorized entry block. ■
> AI Guardrail = software■level “security alarm + barrier” jo
• Input (galat prompt)
• Output (galat response)
• Action (risky tool call)
rokta hai.

## 3■■ ■ Teen Types of Guardrails – In■Depth

Type	Rokta Hai	Implementation Idea
Input	Unsafe / irrelevant prompts	Validate user query before passing to LLM
Output	Harmful / hallucinated LLM responses	Scan tokens or chunks for profanity / bias
Tool■Safe	Risky tool invocations (delete, transfer)	Require “Are you sure?” re■prompt or block

Bonus: Hybrid Guardrails
• Contextual: Adapt rules per conversation context.
• Learning: Update rules dynamically from logs.

## 4■■ ■ Python Examples – From Simple to Streaming

### A) Simple Profanity Filter (Beginner)

```
bad_words = {"idiot", "hate", "stupid"}

def filter_output(text):
    for w in bad_words:
        if w in text.lower():
            return "[Response blocked due to policy]"
    return text

print(filter_output("You are stupid!"))  # [Response blocked due to policy]
```

### B) Chunk■by■Chunk Streaming Guardrail

```
def stream_with_guardrail(stream):
    buffer = ""
    for token in stream:
        buffer += token
        if len(buffer) > 100:
            if violates_policy(buffer):
                yield "[■■ Content blocked]"
                return
            buffer = ""
        yield token
```

### C) Tool■Safe Execution Guardrail

```
def safe_tool_call(tool_func, *args, **kwargs):
    if is_risky_tool(tool_func.__name__):
        confirm = input("Are you sure? (yes/no): ")
        if confirm.lower() != "yes":
            print("■ Tool execution blocked.")
            return None
    return tool_func(*args, **kwargs)
```

## 5■■ ■ Pro Patterns: Chunking, Dynamic Rules & Tool■Safe

1. **Adaptive Chunking**: Adjust chunk size per rule complexity.
2. **Dynamic Rule Sets**: Use a policy engine to load JSON/YAML rules.
3. **Context■Aware**: Stricter checks for sensitive domains.
4. **Tool Plugins**: Wrap external calls with guardrail checks.

## 6■■ ■ Rapid■Fire FAQ & Edge Cases

• **Q:** AI phir bhi leak kar raha?
A: Increase strictness, log & retrain policies.
• **Q:** Latency high?
A: Pre■compile checks, async, batch.
• **Q:** False positives?
A: Whitelist, tune rules.
• **Edge Case:** Multi■lingual bad words → multilingual lists.

## 7■■ ■ Summary & Next Steps

- **Guardrails** = AI safety fences.
- **Types:** Input, Output, Tool■Safe, Hybrid.
- **Implementation:** Filters → chunk checks → policy engine → wrappers.
- **Pro Tip:** Log & refine from real data.

■ **Next:** Combine with Streaming & Context integration.
■ Command: `Hammad Bhai, integration guide do!`