# CHAPTER 1

# INTRODUCTION

## 1.1 The Problem: LLMs Sound Smart but Often Guess Wrong

Large Language Models are incredible at producing fluent, convincing text—but that fluency hides a dangerous flaw: they routinely generate statements that *sound true* but are actually wrong. These hallucinations are not rare, and in high-stakes settings like medical advice or legal reasoning, they can cause real harm.

## 1.2 Why Current Fixes Aren't Enough

Most existing systems try to catch hallucinations *after* the model has already generated an answer. Retrieval-augmented generation, consistency checks, or post-hoc verification help, but they waste compute and fail to prevent the model from producing false content in the first place. In other words, the model has already "spoken," and now you're scrambling to clean up the mess.

## 1.3 A Shift in Strategy: Stop the Hallucination Before It Starts

This project flips the usual pipeline. Instead of waiting for the model to hallucinate, we analyse the model's internal signals *before* generation begins. The goal is simple: determine whether the model is likely to answer reliably. If not, the system automatically reroutes the query to a safer option—retrieval, a larger model, or a human.

## 1.4 How We Estimate Confidence

The confidence estimator draws from three powerful internal clues:

#### 1.4.1. **Semantic Alignment**

We compare the model's internal representation of the query with embeddings from a trusted reference model. If alignment is weak, the model probably doesn't "understand" the query well.

#### 1.4.2. **Internal Convergence**

We look at how stable the hidden layers are as the model processes the input. Poorly converging layers usually signal uncertainty or confusion.

#### 1.4.3. **Learned Confidence**

A dedicated neural predictor is trained to read internal activations and estimate reliability directly. These three signals are combined into a single confidence score.

## 1.5 Smart Routing Based on Confidence

Once the score is computed, the system chooses one of four paths:

- **High confidence** → small local model responds
- **Medium confidence** → use retrieval to ground the answer
- **Low confidence** → escalate to a larger, stronger model
- **Very low confidence** → hand off to a human reviewer

This makes the system faster, cheaper, and safer than post-generation fixes.

## 1.6 What This Achieves

Across multiple QA benchmarks, this method sharply improves hallucination detection while cutting computational cost by around 40%. The model gets better at knowing *when it doesn't know*, and that self-awareness leads to fewer mistakes and tighter control over reliability.

| **Feature** | **Reactive Standard (Current Industry)** | **Proactive Approach (Proposed System)** |
|---|---|---|
| **Timing of Detection** | Checks reliability **after** the text is generated. | Estimates confidence **before** generation begins. |
| **Computation Cost** | High: Wastes compute generating bad answers that are later discarded. | Low: Routes simple queries to small models and prevents wastage. |
| **Methodology** | Relies on consistency checks (SelfCheckGPT) or post-hoc verification. | Uses internal signals (Alignment, Convergence) and external risk checks. |
| **Goal** | "Fix the mess" after the model has spoken. | "Stop the hallucination" before it starts. |

Table 1.1 Reactive vs Proactive

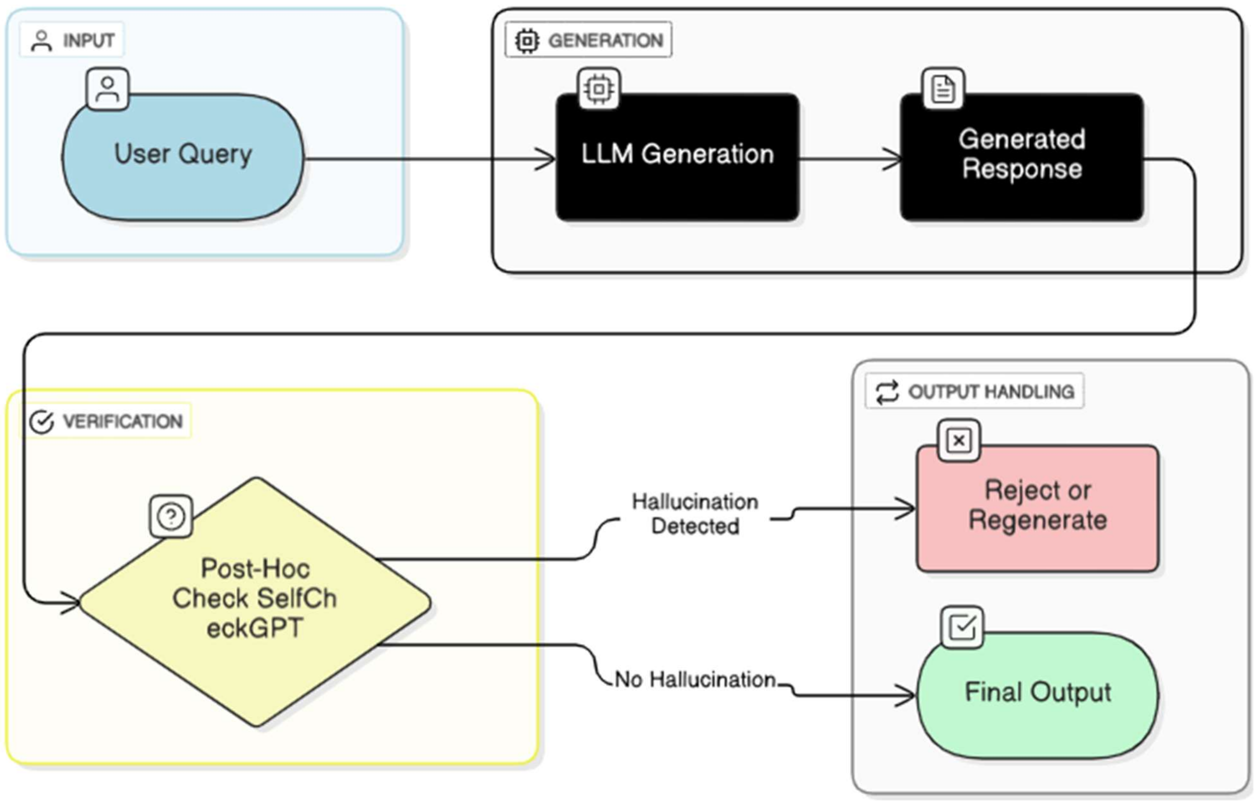

Figure 1.1 Traditional Reactive Hallucination Mitigation Pipeline