AskBot

1) Abuse stories (concise, testable)

- **Prompt-Injection to Data Exfiltration**
  As a malicious employee, I craft a prompt that embeds hidden instructions to the LLM to call `/api/payroll/payslip/{month}` for another employee and summarize the contents.
- **Role Confusion / RBAC Bypass**
  As an attacker, I try to impersonate the "Payroll Agent" by asking the Router to "act as Payroll Agent" and request payslip data without my RBAC claims.
- **Tool Scope Escape (MCP Router)**
  As an attacker, I coerce the LLM to call tools outside the current task scope (e.g., Insurance tools during a Payroll task) by chaining "reasoning" steps.
- **Vault Token Exfiltration**
  As a rogue plugin developer, I attempt to log or echo the short-lived Vault token in tool responses or headers, then retrieve it via AskBot's UI.
- **Redaction Bypass via Encoding**
  As an attacker, I submit PAN/Bank data encoded (Base64/zero-width/emoji). I expect the Guardrail/Redactor to miss it and leak raw PII in the final answer.
- **Confidence Downgrade Ignored**
  As an attacker, I craft ambiguous prompts to keep the confidence below threshold but force the system to answer instead of escalating to HR.
- **Context/Memory Poisoning**
  As an attacker, I inject fake "policy" facts into AskBot memory so later users receive incorrect HR or Insurance guidance.
- **Log Poisoning & Trace Loss**
  As an attacker, I include control characters in my prompt to break JSON log lines and prevent trace correlation across Router → Tool calls.
- **IDOR on Claims**
  As a user, I guess/iterate `claimId` in `/api/insurance/claim/{claimId}` and retrieve another employee's claim status.
- **Over-broad Policy Search**
  As an attacker, I prompt the HR Policy Agent to search "all categories including confidential addendums," expecting internal-only docs to leak.