EQUIVOCAL LEGAL · PROPOSAL DEFENSE

HZU

基于 LangGraph 的法律智能体系统设计与实现

自建 LangGraph + RAG · 开题答辩

姓名：沈佳豪

学号：

学院/专业：

指导教师：

湖州学院　计算机科学与技术学院

· 2026 年 1 月 12 日

目录

Agenda | AI-Native Minimal | Traceable MVP

Narrative

1. 背景与问题定义：为什么需要法律智能体
2. 目标与边界：面向可验收的 MVP
3. 系统方案：LangGraph + RAG 的工程化落地
4. 实验设计：验收指标与风险对策
5. 计划与预期成果：里程碑与交付物

Audience takeaway

- 这不是“聊天机器人”，而是**工程化流程**
- 关键输出可**追溯/回放/审计**
- 上游失败时可**降级**而非沉默

背景与意义：为什么需要法律智能体

Problem | Cost | Risk | Fragmented info

User pain

- **成本高、门槛高**：依赖专业人力；用户常常“问不到点子上”
- **风险点隐蔽**：条款风险埋在细节里，误解可能带来高代价
- **信息碎片化**：法条/案例/流程分散，检索与比对成本高

Why now

大模型带来自然语言交互能力，**RAG**把回答与依据绑定，让关键结论**可追溯**、**可复核** [1]。

目标不是“更会聊天”，而是“更可验收”。

问题定义：把“咨询”工程化到可验收

Definition | MVP | Traceable | Safe boundary

One-sentence mission

让用户在 3 分钟内得到**可执行的下一步**（需要补充哪些事实 / 风险点在哪里 / 建议怎么做），并保证**可追溯**、**可回放**、**可验收**。

Boundaries & principles

- **不替代律师**：输出建议与依据，提示适用范围与不确定性
- **可追溯**：关键输出绑定 traceId / 模型信息 / 引用来源
- **可降级**：上游不可用时返回结构化替代步骤（非“报错/沉默”）
- **隐私最小化**：最小化存储、可选脱敏，避免泄露敏感信息

系统方案：LangGraph + RAG 的工程化落地

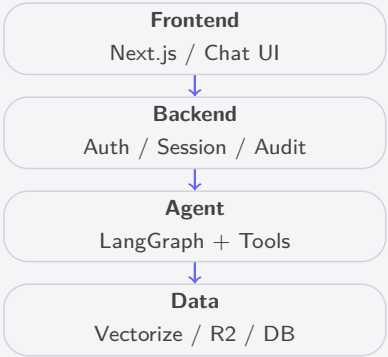
Architecture | Next.js | Java Gateway | LangGraph | RAG

Components

- **前端（Next.js）**：对话 UI、服务类型入口、SSE 流式渲染
- **后端（Java）**：JWT 鉴权、会话管理、审计留痕、限流/风控
- **智能体（FastAPI + LangGraph）**：多步推理 + 工具调用，必要时检索引用 [2]
- **数据层**：向量检索（Cloudflare Vectorize）+ 文件存储（R2）+ 结构化日志（DB）

目标：把“咨询”变成可追溯的工程流程。

Flow



关键流程：工具调用 + 引用 + 追溯（traceId）

Trace | Tools | Citations | Replayable

Request pipeline

- **Input**：问题/合同片段（可选脱敏）+ 用户身份
- **Guard**：鉴权、限流、风险提示模板
- **Agent**：决策（检索/生成/追问）
- **Tools**：RAG / OCR / 文书生成
- **Output**：建议 + 免责声明 +（可选）引用 + traceId

LangGraph (pseudo)

```
graph = StateGraph(State)
graph.add_node("decide", decide)
graph.add_node("rag", legal_rag_search)
graph.add_edge("decide", "rag")
graph.add_edge("rag", "decide")
graph.compile()
```

要点：把多步推理固定为图结构，便于测试与回放。

核心能力与服务类型（MVP）

MVP | Contract review | Drafting | Consulting

Service types

服务类型	典型输出
合同审查	风险点 + 修改建议 +（可选）引用依据
合同生成/保密协议/借条/租赁合同	结构化信息收集 + 文书草案 + 风险提示
法律咨询	追问补全事实 + 可执行下一步 + 边界声明

Why tools matter

工具让智能体能**检索引用**、**生成文书**、**做 OCR**，并把多步推理固定成可验证的流程（而不是一次性生成）[3]。

验收指标：可用、可追溯、可降级

Acceptance | Available | Traceable | Degradable

Availability

关键接口可用性 $\geq 99\%$；失败可定位（错误码 + traceId）。

Latency

流式输出首 token 体验优先；常见请求 $< 2s$（按环境调参）。

Traceability

会话/消息/关键输出可检索（provider / mode / latency / tokens）。

Fallback & boundary

检索失败/模型不可用时返回“下一步要补充的信息清单”，并提示不确定性与适用范围。

阶段性成果（当前进展）

Progress | MVP ready | Engineering-first

Product

多服务类型入口（合同审查 / 文书生成 / 法律咨询），围绕“**可执行下一步**”。

RAG infra

向量库（Cloudflare Vectorize）+ 文件存储（R2），支持引用与材料上传。

System

前端（Next.js）+ 后端（Java）+ 智能体（FastAPI + LangGraph）链路成型。

Acceptance baseline

追溯字段、审计日志、失败降级策略，支持**可验收与可复盘**。

计划与预期成果（W1 – W8）

Roadmap | W1 – W8 | Deliverables

Milestones

- W1 – W2：用例与数据结构定稿；会话/审计表结构；关键页面原型
- W3 – W4：Java 网关与鉴权完善；SSE 流式链路；端到端联调
- W5 – W6：智能体策略（追问/检索/生成）；降级与风控；引用输出规范
- W7：测试与验收脚本（合同审查/劳动争议/文书生成）
- W8：论文与答辩材料；演示脚本与彩排

Deliverables

- 可演示 MVP：登录 + 咨询 + 留痕 + 可追溯引用
- 关键场景 Demo：合同审查 / 文书生成 / 法律咨询
- 可验收口径：指标 + 日志 + traceId 回放 + 风险对策

Q & A

谢谢。欢迎提问与质疑我的边界与验收口径。

Questions Boundary Acceptance

参考文献 i

References LangGraph RAG Legal NLP

[1] Patrick Lewis et al. **"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks".** In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020. arXiv: 2005.11401 [cs.CL]. URL: https://arxiv.org/abs/2005.11401.

[2] LangChain AI. ***LangGraph: Build resilient language agents as graphs.*** 2024. URL: https://github.com/langchain-ai/langgraph (visited on 01/11/2026).

[3] Shunyu Yao et al. ***ReAct: Synergizing Reasoning and Acting in Language Models.*** 2022. arXiv: 2210.03629 [cs.CL]. URL: https://arxiv.org/abs/2210.03629.