孔维轩 (+86) 18796611232 | Jackkong29@Gmail.com

github.com/Jackela | jackela.github.io

个人简介

一名专注 AI 工程与系统架构的计算机科学硕士, 具备扎实的产品思维。我的核心开发范式是: 将 AI 辅助开发 与严格的工程纪律 (DDD, TDD, DbC) 相结合。我擅长利用 AI 工具高速生成 实现代码, 同时通过一个自动化的 “工程验证框架” 来确保代码质量、健主性与可维护性, 实现开发效率与 系统可靠性的平衡。

教育经历

悉尼大学 (The University of Sydney) *2024-02 – 2026-02 (预计)*

计算机科学硕士 *(Master of Computer Science)* 预计将于 *2025* 年 *12* 月完成所有课程

阿尔戈马大学 (Algoma University) *2020-09 – 2023-06*

计算机科学学士 *(Bachelor of Computer Science)*

核心能力与技术栈

架构与方法论: 领域驱动设计 (DDD), 事件驱动架构 (EDA), 测试驱动开发 (TDD), 契约式设计 (DbC), CQRS,

AI & 分布式: 多智能体系统 (Multi-Agent), LLM-as-Judge, Kafka, RabbitMQ, gRPC

语言与框架: Python (FastAPI), Java (Spring Boot), TypeScript/JavaScript (React, Node.js, Express)

数据库 & 云原生: PostgreSQL, MongoDB, Redis, AWS (Lambda, S3, EventBridge), Docker

实习经历

愿景明德 (北京) 控股集团有限公司 *2023-06 – 2023-08*

AI 工程师实习生

- AI 服务层解耦与架构: 协助设计并实现了一个基于 AWS 的 AI 服务层。利用微服务架构、API Gateway 和 gRPC 通信, 成功将核心 AI 推理模块 (Python) 与 3 个 上层业务应用 (如智能音箱、聊天机器人等) 彻底解耦, 提升了系统的可维护性与开发效率。
- 云原生异步管道: 引入事件驱动设计, 利用 AWS Lambda、S3 Events 和 EventBridge 构建了一个高性能的异步处理管道。此设计优化了 AI 服务的响应延迟, 并使用 Saga 模式确保了跨多个微服务的最终数据一致性。
- **DDD 实践与技术赋能:** 应用 DDD 思想参与了 AI 服务限界上下文 (Bounded Context) 的划分与定义，并主导了 LLM API 的技术文档编写与 3 场关于 LLM 应用与 Prompt Engineering 的技术培训。

核心项目

CATAMS (Capstone 毕业设计 临时教职工时间管理系统) *2024-08 – 2025-06 (预计)*

产品负责人 *github.com/Jackela/casual-academic-time-allocation-management-system*

- **AI 辅助开发范式:** 本项目是 “AI 辅助生成, 工程实践验证” 范式的核心案例研究。利用 AI 工具高速生成实现 代码, 同时设计并实施了一套严格的 “工程验证框架”。
- **“AI 安全网” (DDD, TDD & DbC):** 深入应用 DDD, 将《企业协议》中 20+条 复杂薪酬规则模型化为 3 个 限界上下文，并严格遵循 TDD 与 DbC 思想构建了自动化测试 “安全网” (**测试覆盖率 > 85%**)，自动化验证 AI 产出。
- 务实的架构权衡: 基于项目约束 (单人开发、一学期时限) 和 “康威定律” (Conway’s Law) 原则, 明确拒绝了微服务架构, 选型 “模块化单体” 架构。

Novel-Engine (多智能体互动小说生成器) *2024-10 – Present*

独立开发者 *github.com/Jackela/novel-engine*

- 项目概述: 一个探索性的 AI 项目, 旨在通过设计一个 “多智能体系统” (Multi-Agent System), 由 “导演” 智能体 来编排 多个 “角色” 智能体, 以在 “涌现行为” 与 “叙事控制” 之间取得平衡。
- 企业级架构 (DDD, EDA & CQRS): 采用领域驱动设计 , 将复杂业务 (如角色、叙事、世界状态) 划分为清晰的 “限界上下文” (Bounded Context)。使用 Kafka 构建事件驱动架构, 通过领域事件实现核心上下文的松耦合。应用 CQRS (命令查询职责分离) 模式, 为 “写” 操作 (命令) 和 “读” 操作 (查询) 分别进行优化。
- 健壮的状态管理架构: 识别出系统的核心复杂度在于“回合” (Turn) 的计算 (一个多步骤、可能失败的复杂事务)。为确保该过程的原子性和可恢复性, 主动应用了成熟的分布式系统设计模式: Saga 模式, 并实现了 Saga Coordinator (Saga 协调器)。