华中科技大学本科毕业论文

基于大模型的以学生为中心的智能教学平台设计与实现

——以电磁场与研究生专业英文写作为例

学生姓名：胡傲东

学　　号：U202214900

专　　业：基础电路设计与集成系统

学　　院：集成电路学院

指导教师：聂彦

2026 年 5 月

原创性声明

本人声明所呈交的毕业论文是本人在导师指导下进行的研究成果。除文中已经注明引用的内容外，本论文不包含任何他人已经发表或撰写的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：_____

日期：_____

授权使用声明

本人同意华中科技大学保存并向国家有关部门或机构送交本论文的复印件和电子版，允许本论文被查阅和借阅。本人授权华中科技大学将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存和汇编本论文。

作者签名：_____

日期：_____

摘要

本文面向高等教育中”课程类型多样、学生差异显著、过程性反馈成本高、生成式模型可信度难以保障”等问题，设计并实现一套以学生为中心的智能教学平台。平台以学习过程数据为核心，围绕作业提交、对话辅导与学习事件形成可追踪的学生档案，并在此基础上提供可控、可追溯、可迭代的智能辅导能力。系统采用前后端分离与服务化架构：前端使用 React + TypeScript + Vite 构建 Web/H5 客户端，可嵌入企业微信 WebView；后端使用 Go + Gin 提供业务 API、JWT 鉴权与 RBAC 权限管理；AI 服务基于 Python + FastAPI 负责编排对话、写作分析、引导式学习、工具调用与检索增强，通过 OpenAI-compatible 接口对接推理服务。异构加速层支持 GPU、NPU 与 FPGA 三种后端：GPU/NPU 用于大模型推理，FPGA 用于 Embedding 服务的低延迟加速与资源解耦，在保持检索质量的前提下显著降低能耗。为降低幻觉并提升可复核性，平台引入 GraphRAG 检索增强，将课程资料构建为图结构索引，生成回答时注入证据片段并按编号引用；对于数值计算、仿真或格式检查等可验证任务，系统通过工具调用将关键结论锚定在可执行结果上。为支持模型定制与持续迭代，平台提供数据规范、LoRA/QLoRA 微调与离线评测脚本，并引入数据蒸馏与 smoke 验证用于训练链路自检。本文以电磁场推导型课程与研究生专业英文写作课程作为示例场景进行验证，展示平台在不同课程中的可迁移性与工程可落地性。

关键词：智能教学平台；学生档案；GraphRAG；工具调用；引导式学习；异构加速；FPGA

Abstract

This thesis addresses common challenges in higher education, including diverse course types, significant learner differences, the high cost of process-oriented feedback, and the difficulty of guaranteeing reliability in generative models. We design and implement a student-centric intelligent teaching platform powered by large language models (LLMs). The platform organizes learning-process data (submissions, tutoring dialogues, and learning events) into longitudinal student profiles, enabling controllable, traceable, and iterative tutoring. The system follows a service-oriented architecture with a separated front end and back end: the client is built with React, TypeScript, and Vite for web/H5 (embeddable in a WeCom WebView); the backend uses Go and Gin to provide business APIs with JWT-based authentication and RBAC; and the AI service is implemented with Python and FastAPI to orchestrate chat, writing analysis, guided learning, tool calling, and retrieval augmentation, connecting to upstream inference via an OpenAI-compatible API. The heterogeneous acceleration layer supports GPU, NPU (Huawei Ascend), and FPGA (Xilinx Alveo) backends: GPU/NPU for LLM inference, and FPGA for low-latency embedding acceleration and resource decoupling, significantly reducing power consumption while maintaining retrieval quality. To reduce hallucinations and improve auditability, we introduce GraphRAG: course materials are indexed as a graph-structured knowledge base, and responses are grounded on retrieved evidence with explicit citations. For verifiable tasks such as numerical calculation/simulation or format checks, tool calling delegates key steps to executable tools and injects results back into the dialogue. To support model customization and continuous iteration, we provide a data specification, LoRA/QLoRA fine-tuning and offline evaluation scripts, as well as data distillation and smoke validation for pipeline sanity checks. Two representative scenarios—an electromagnetics derivation-intensive course and a graduate academic writing course—are used to demonstrate portability and engineering feasibility.

Keywords: intelligent teaching platform; student profile; GraphRAG; tool calling; guided learning; heterogeneous acceleration; FPGA

目录

插图

表格

第一章　绪论

1.1　研究背景与意义

高校课程呈现出显著的多样性：一类是推导与计算密集的理工科课程（例如电磁场），其学习难点在于概念依赖关系复杂、公式推导链条长且对计算正确性要求极高；另一类是过程性与评价维度复杂的技能型课程（例如研究生专业英文写作），其学习难点在于论证组织、学术语气、证据与引用规范等能力需要长期迭代训练。两类课程在教学组织上都面临共同痛点：学生个体差异显著、过程性学习数据难以系统沉淀、教师在答疑与反馈中的边际成本高，导致“高频但低复用”的重复劳动难以规模化优化。

近年来，Transformer 架构[1]推动大语言模型（LLM）在自然语言理解与生成方面取得突破，对话式模型（如 ChatGPT）[2]为教学辅助提供了新的交互范式：既可解释概念、提供建议，也可进行循序渐进的引导。然而，LLM 在教育场景中面临可靠性与可治理性挑战：模型可能产生缺乏依据的建议（幻觉），对推导/格式等“硬约束”任务容易出现不可复核的错误，且难以稳定贴合不同课程的评价标准。检索增强生成（RAG）[3]与知识图谱[4]为上述问题提供了可追溯的外部知识约束：通过将课程规范、讲义、示例与题解片段构建为可检索证据库，让生成输出有据可依、可引用溯源。在此基础上，引入 GraphRAG 的“片段检索 + 图扩展”机制，可进一步利用知识间的结构化关系提升关联检索与多跳推理能力。与此同时，工具调用（Function Calling）把关键计算/校验步骤交由可执行工具完成，为数值计算、仿真或写作结构/引用规范检查等任务提供可验证路径。

基于企业微信内嵌 H5 或网页的入口具备便捷触达与组织管理优势，适合承载可扩展的智能教学平台原型。本文以电磁场课程与研究生专业英文写作课程为示例场景，探索一种强调“学生中心、可追溯、可验证、可迭代”的智能教学平台设计与实现方法。

1.2　国内外研究现状

在教育信息化领域，学习管理系统（LMS）与学习分析工具已能支持课程资源管理、作业提交与学习过程记录，但对“解释 + 引导 + 过程性反馈”的深层教学任务支持有限。近年 LLM 在教育问答、自动反馈与写作辅助等场景中应用快速增长，研究重点逐步从“能生成”转向“可追溯、可验证、可治理”：RAG 被广泛用于降低幻

觉并提升答案依据[3]；知识图谱在教材知识组织、概念关联与检索推荐中具有优势[4]，但与 LLM 深度协同、并能同时服务多课程场景的系统化平台仍处于快速演进阶段。

在检索增强生成方面，传统 RAG 通过向量检索注入外部知识以降低幻觉，但在知识关联性强、需要多跳依赖的任务中仍存在召回不足与上下文碎片化等问题。GraphRAG 将文档组织为“节点—边—片段”的图结构索引，支持从种子片段出发的图扩展，以更好地覆盖前置概念、相关规范与示例。对于需要“硬约束”的任务，工具调用机制通过结构化参数把计算/校验委托给可执行工具，减少模型自由生成导致的错误。综上，构建面向多课程的智能教学平台，并将检索、工具与训练能力工程化落地，具有明确的研究与应用价值。

1.3　研究问题与创新目标

1.3.1　核心研究问题

本文围绕以下四个核心研究问题展开：

(RQ1) 如何在不同课程任务中降低幻觉，并保证关键结论具备可验证性（例如数值计算/仿真、格式与结构检查）?

(RQ2) 如何构建可追溯的知识检索机制，既支持多跳关联检索，也支持证据引用与来源标注?

(RQ3) 如何把“教学风格与引导策略”显式化，使模型输出在不同课程中可控且可迁移?

(RQ4) 如何将作业、讨论、讲义与规范等多源数据纳入统一数据闭环，并支撑持续迭代与回归评测?

1.3.2　创新目标

针对上述研究问题，本文提出以下可验证的创新目标：

(1) 提出并落地“学生中心数据闭环 + GraphRAG 可追溯 + 工具调用可验证 + 引导式学习”的一体化框架，形成可复用的工程实现路径。

(2) 设计“通用能力 + 课程专属模块”的课程模块化策略，以电磁场与研究生专业英文写作为例验证跨课程可迁移性与能力边界。

(3) 建立面向持续迭代的训练与回归评测流程：数据规范化、数据蒸馏与 smoke 自检、LoRA/QLoRA 训练与离线评测报告输出。

1.4　研究内容与任务要求

本文围绕面向多课程的智能教学平台设计与实现开展研究，主要内容与任务要求如下。

1.4.1　课题内容

(1) 平台总体架构与工程化实现：React 前端、Go 后端与 FastAPI AI 服务的分层与接口契约设计，支持企业微信内嵌 H5/网页访问。

(2) GraphRAG 知识库构建与检索增强生成：面向课程材料/规范/示例构建图结构索引，并实现引用编号的可追溯回答。

(3) 工具调用与可验证执行器：对数值计算/仿真、写作结构与引用规范检查等任务提供工具调用能力与结果回注机制。

(4) 引导式学习与学习画像：生成学习路径并逐步引导，记录薄弱点与学习进度，支撑过程性反馈闭环。

(5) 模型后训练与评测管线：数据规范、数据蒸馏与 smoke 验证、LoRA/QLoRA 微调与离线评测报告输出。

(6) 示例场景验证与测试评估：以电磁场与研究生专业英文写作为例进行端到端演示与评估设计。

1.4.2　课题任务要求

(1) 深入理解大语言模型的基本原理及其应用范式，掌握至少一种主流 LLM 的 API 调用方法。

(2) 掌握知识图谱与 GraphRAG 的构建流程，能够将课程讲义、规范与示例构建为可检索的证据库，并在回答中实现引用溯源。

(3) 实现工具调用机制，支持数值计算/仿真与规则化检查任务的可验证执行与结果回注。

(4) 完成一个可部署的系统原型，覆盖鉴权、权限治理、知识库检索增强、对话与引导式学习等关键链路。

(5) 原型系统应体现跨课程可迁移性：在电磁场与研究生专业英文写作等示例任务上能够输出结构化、可追溯、可复核的辅助结果。

(6) 完成毕业设计论文的撰写，论文应结构清晰、论证充分、代码和数据详实。

1.5　论文结构

全文共分六章：第一章为绪论，介绍研究背景意义、国内外研究现状、研究问题与创新目标，并给出研究内容与任务要求；第二章阐述平台涉及的关键技术与理论基础，包括 React 前端、Go 后端、GraphRAG、技能系统、工具调用、引导式学习、学习状态追踪以及 LoRA/QLoRA 与数据蒸馏等；第三章给出系统需求分析与总体设计，明确业务流程、数据与权限模型及跨课程可扩展方案（以电磁场与研究生专业英文写作为例）；第四章面向工程落地，详细介绍 Skills/Tool Calling/GraphRAG、引导式学习与学习档案、训练与评测管线等关键模块的设计与实现；第五章对系统进行测试与实验评估，从功能正确性、引用一致性、工具调用可用性与回归评测等方面验证方案效果；第六章总结全文并展望后续工作。

第二章　相关技术与理论基础

2.1　前端技术选型

客户端采用 React 作为 UI 框架，配合 TypeScript 进行静态类型检查，使用 Vite 完成项目构建与模块热替换。React 的组件化特性使得” 对话窗口”” 学习进度看板”” 写作反馈面板” 等界面单元能够独立开发与复用。TypeScript 的类型系统有助于前后端在接口定义上保持一致，减少因字段命名或类型不匹配导致的运行时错误。为适配企业微信内嵌 WebView 与普通浏览器两种接入场景，界面布局采用弹性栅格与媒体查询相结合的方式，同时在功能层面通过特性检测判断宿主环境能力，保障跨端体验基本一致。

2.2　后端服务架构

后端选用 Go 语言与 Gin 框架，前者在并发模型与部署体积上具备优势，后者提供中间件机制便于横切关注点（日志、限流、异常恢复）的统一处理。代码组织遵循分层原则：Handler 层负责请求解析与响应封装；Service 层承载业务逻辑；Repository 层封装数据库操作。权限模型基于角色（RBAC），教师、助教、学生对同一资源拥有差异化访问范围；认证方案采用 JWT，令牌无状态特性简化了分布式环境下的会话管理。持久化层使用 MySQL，表结构变更通过迁移脚本版本化管理，便于开发、测试与生产环境保持同步。

2.3　大语言模型与检索增强

2.3.1　Transformer 架构简述

Transformer 模型[1]以自注意力机制取代传统循环结构，在处理长距离依赖时表现出明显优势。当前主流大语言模型（GPT 系列、Qwen 系列等）均以 Transformer 为骨干网络，在语言理解、摘要生成、翻译等任务上取得显著成效。不过，这类模型并非总能给出可靠答案：有时会产生与事实不符的描述（即” 幻觉”），有时会省略推导中间步骤，给教学场景带来可信度风险。

2.3.2　检索增强生成的基本思路

检索增强生成（Retrieval-Augmented Generation，简称 RAG）[3]的核心想法是在生成前先从外部知识库中检索相关片段，再将其拼接到上下文中供模型参考。相比纯参数记忆，这种做法能有效降低幻觉概率并提升结论的可追溯性。常规 RAG 流程包括：文档切片、向量化、相似度检索、上下文拼接。但在理工科推导型场景下，纯向量检索难以表达公式间的推导链条与概念间的前置依赖关系，召回质量会受到限制。

2.3.3　GraphRAG 的设计考量

GraphRAG 在向量检索基础上叠加知识图谱结构，将文档片段组织为”节点—边”的索引。其运作方式可归纳为三点：

(1) **混合检索**：同时执行关键词匹配与语义向量检索，通过倒数排名融合（RRF）合并两路结果，兼顾精准召回与语义扩展。

(2) **图扩展**：以检索到的片段为种子，沿图谱边扩展一至两跳，补充前置概念与关联推导要点，形成更完整的推理材料。

(3) **引用标注**：生成答案时以 `[1]` `[2]` 等编号标注来源片段，便于用户核对原文。

2.4　技能系统与提示编排

在多课程、多任务场景下，不同功能点对提示词有差异化需求。如果在代码各处零散编写提示模板，后期维护成本会急剧上升。因此本系统将”提示模板、上下文注入规则、输出格式约束、安全边界”统一封装为”技能”（Skill），并以注册表形式集中管理。

2.4.1　技能抽象

每个技能的核心由三部分组成：

(1) **系统提示**：描述角色身份、任务边界、输出结构（例如先给出结论再展开解释、必须标注引用来源、遇到代写请求应婉拒）。

(2) **上下文注入**：在不干扰用户原始对话的前提下，向模型上下文追加课程背景、作业要求、学生档案等信息。

(3) **扩展注册**：通过注册表统一管理技能集合，对历史模式名提供别名映射以保持接口兼容。

2.4.2　技能与其他能力的组合

技能侧重” 教学风格与行为约束”，检索增强侧重” 证据支撑”，工具调用侧重” 可验证计算”。三者在工程上相互独立：同一技能既可在无检索场景下工作，也可在启用 GraphRAG 时以证据片段约束输出。这种解耦设计降低了功能扩展时的耦合风险，具体路由与实现细节留待第 4 章展开。

2.5　工具调用机制

工具调用（Tool Calling）使语言模型能够判断何时需要外部工具协助，并输出结构化调用请求。调用完成后，工具返回值被回注到对话上下文，模型再据此生成最终回答。该机制在可验证性要求较高的教学任务中尤为重要：

(1) **符号计算**：调用计算库对公式进行推导或化简。

(2) **数值仿真**：调用仿真服务执行课程专属计算（如电磁场分布）。

(3) **表达式求值**：对物理常数或中间结果进行精确计算，规避模型” 心算” 失误。

(4) **规则检查**：对写作样本的字数、结构元素、引用格式等执行规则化校验。

为防止模型无限循环调用或调用未授权工具，系统在工程层面实施工具白名单、最大调用次数与调用超时等策略，第 4 章将结合具体实现说明。

2.6　引导式学习与学习状态追踪

传统的” 一问一答” 模式难以覆盖复杂能力的渐进式培养。引导式学习将某一学习主题拆分为 3–6 个可管理的步骤，系统依次向学生抛出问题或提示，待完成当前步骤后再进入下一步。这种” 苏格拉底式” 的互动方式有助于减少” 听懂但不会做” 的落差。

为支撑这一模式，系统需显式维护会话状态：当前步骤、已完成步骤、历史对话、阶段性薄弱点等。这些信息进一步汇聚为学习档案（课程画像与跨课程全局画像），供教师端查看学情分布或开展个性化干预。会话管理与薄弱点检测的具体实现将在第 4 章详细讨论。

2.7　异构加速：NPU 与 FPGA

在教育场景下，模型推理与检索服务需要兼顾响应延迟与部署成本。除传统 GPU 方案外，NPU 与 FPGA 两种异构加速路径可分别针对大模型推理与向量检索提供优化。

2.7.1　NPU 加速大模型推理

NPU（神经网络处理器）针对矩阵运算与张量操作进行硬件层面优化，配合 CANN 软件栈实现算子加速。以华为 Ascend 910B 为例，7B 参数量模型的首 token 延迟与 NVIDIA RTX 4090 处于同一数量级，但满载功耗降低约 20%，适合对能耗与机房条件有约束的校园部署场景。系统在设计上预留了 NPU 推理接入能力：

(1) **模型转换**：通过 ATC 工具将 Hugging Face 模型转换为 Ascend 可执行格式（OM 模型）。

(2) **推理框架**：使用 MindIE 或 vLLM Ascend 后端提供推理接口。

(3) **服务封装**：对外暴露 OpenAI-compatible 接口，AI 服务层无需区分上游硬件。

2.7.2　FPGA 加速向量检索

在检索增强链路中，Embedding 计算是主要的延迟来源之一。当 GPU 被大模型推理占用时，Embedding 与 Reranker 若共享 GPU 会产生资源竞争，影响整体吞吐。FPGA（现场可编程门阵列）作为可定制化加速器，适合承载固定计算模式的推理任务。

FPGA 加速的基本思路：

(1) **模型量化与编译**：将 Embedding 模型（如 bge-base 或 sentence-transformers）通过 Vitis AI 量化为 INT8 精度，并编译为 FPGA 可执行指令（xmodel）。

(2) **硬件部署**：在 Xilinx Alveo 系列加速卡（如 U50、U250）上部署量化模型，通过 PCIE 与主机通信。

(3) **服务封装**：将 FPGA 推理封装为 HTTP 微服务，对外提供与 CPU/GPU Embedding 服务相同的接口，便于 AI 服务层无感切换。

预期收益：

(1) **延迟降低**：FPGA 的确定性时序特性可提供稳定的低延迟 Embedding 计算。

(2) **资源解耦**：将 Embedding 从 GPU 迁移至 FPGA，释放 GPU 资源专注于大模型推理，提升整体并发能力。

(3) **能效优势**：FPGA 典型功耗 30–75W，远低于 GPU，适合持续运行的检索服务。

第 4 章将给出 FPGA 加速 Embedding 服务的具体实现，第 5 章将对比 CPU、GPU、FPGA 三种 Embedding 后端的性能与检索质量。

2.8　参数高效微调与训练链路

教育场景对输出稳定性要求较高，仅靠提示编排有时难以满足需求。参数高效微调提供了一条低成本路径。

2.8.1　LoRA 与 QLoRA

LoRA[5]在注意力层引入低秩适配器，仅训练约 0.1% 的参数即可调整模型行为分布；QLoRA[6]在此基础上叠加量化技术，使单张消费级 GPU 也能完成 7B 量级模型的微调。训练产物以 adapter 形式保存，可快速切换或回滚。

2.8.2　数据蒸馏与质量门禁

在启动微调前，系统先对训练数据执行”蒸馏 + smoke 验证”：蒸馏阶段将 chat-style 样本提取为易于审阅的格式并统计去重率；smoke 阶段在分钟级别输出轻量度量（样本数、字段覆盖率、困惑度分布等），用于发现空样本、字段缺失或重复激增等问题。该流程不用于宣称模型最终效果，而是充当训练链路的”自检关卡”。

2.8.3　回归评测

持续迭代需要可量化的回归机制。系统维护一份固定回归集，覆盖 tutor、grader、guided_learning、tool、rag 等任务类型。每次训练后执行预测与评测脚本，输出引用一致性、工具调用准确率、结构化输出可解析率等指标，并附带典型案例分析。第 5 章将给出具体评测设计。

2.9　数值仿真与可视化

对于”计算与现象解释”类需求（以电磁场课程为例），系统可扩展数值仿真模块。采用有限差分法（FDM）求解 Laplace 或 Poisson 方程，典型场景包括二维静电场分布、同轴线电容场等。Python 科学计算生态（NumPy、SciPy、Matplotlib）提供数值求解与绘图能力。仿真服务通过 FastAPI 暴露 HTTP 接口，支持参数化任务提交与异步执行，返回数据（JSON）与图像（PNG）两种形式。仿真结果可作为可验证证据注入 AI 回答，形成”计算—解释—理解”的闭环。该模块属于”课程专属能力”，与写作类课程的规则检查工具形成互补。

2.10　本章小结

本章介绍了系统涉及的关键技术与理论背景：前端采用 React + TypeScript + Vite 构建跨端客户端；后端基于 Go + Gin 实现业务服务与权限治理；异构加速层支持 GPU、NPU 与 FPGA 三种后端——NPU 用于大模型推理的低功耗部署，FPGA 用于 Embedding 服务的资源解耦与延迟优化；GraphRAG 在向量检索基础上引入图扩展以提升召回质量；技能系统统一管理提示编排；工具调用为可验证计算提供支撑；引导式学习与学习档案为过程性辅导提供状态基础；LoRA/QLoRA 与数据蒸馏、回归评测构成后训练链路。上述技术共同支撑平台在不同课程中的可迁移落地，后续章节将围绕需求分析、系统实现与测试评估逐步展开。

第三章　系统需求分析与总体设计

3.1　应用场景与用户角色

本项目面向高校多课程教学场景，强调以学生为中心的过程性支持与可追溯治理：既要覆盖理工科推导/计算密集任务（以电磁场为例），也要覆盖写作类“过程性训练 + 评价维度复杂”的任务（以研究生专业英文写作为例）。系统用户角色主要包括：学生（学习与提交）、教师/助教（发布与反馈、学情分析）以及管理员（课程与权限配置）。在上述角色协作下，平台需同时满足“日常答疑/辅导”的即时交互需求与“长期学习状态沉淀”的过程性数据需求。

3.2　需求分析

3.2.1　功能性需求

结合平台定位与多课程适配目标，系统核心功能需求如下：

(1) **对话式辅导与多模式能力切换**：支持概念讲解、作业反馈、引导式学习、个性化辅导策略生成等模式，并可在课程间复用通用能力。

(2) **可追溯检索增强**：支持将课程讲义、作业规范与示例等构建为证据库，回答需携带引用编号，支持教师复核与溯源审计。

(3) **可验证执行能力**：对符号推导、数值计算、仿真与写作规则检查等任务，支持工具调用并回注结果，形成可复核的“计算/校验链”。

(4) **引导式学习与进度推进**：围绕学习主题生成学习路径并分步提问推进，支持会话恢复与步骤状态更新。

(5) **学习画像与长期状态追踪**：沉淀薄弱点、完成主题、学习时长等信息，形成课程画像与跨课程全局画像，并支持学习时间线查询。

(6) **训练与评测闭环**：支持数据规范化、数据蒸馏与 smoke 门禁、LoRA/QLoRA 微调与离线回归评测，为能力迭代提供可复现链路。

3.2.2　非功能性需求

为保证系统在真实教学环境中的可用性与可治理性，非功能需求包括：

(1) **可信与可复核**：关键结论需能以“引用证据 + 工具结果”进行复核，降低幻觉影响。

(2) **安全与权限**：基于 RBAC 的角色权限控制，支持课程维度的数据隔离与访问控制；工具调用需具备安全约束，避免越权与资源滥用。

(3) **可扩展与可维护**：能力扩展应以模块化方式接入（新增技能/工具/课程材料），避免改动核心接口引发兼容性风险。

(4) **性能与稳定性**：在企业微信 WebView/浏览器等多终端环境中保持稳定交互；检索与工具执行需具备超时与降级策略。

(5) **隐私与学术诚信**：对学生数据与作业内容进行必要保护；对代写等学术不端请求具备拒答/引导策略。

3.3　总体架构设计

3.3.1　分层与服务划分

系统采用前后端分离与服务化架构，整体分为三层：

(1) **Web/H5 客户端**：基于 React 构建交互界面，负责对话、学习进度展示、写作分析结果呈现与教师侧学情概览等。

(2) **业务后端**：基于 Go + Gin 提供统一鉴权、RBAC 权限治理、课程与作业管理、学习画像/事件接口等业务能力。

(3) **AI 服务层**：基于 FastAPI 提供 OpenAI-compatible 的对话接口与可插拔能力，包括 Skills、GraphRAG 检索增强、Tool Calling 执行器以及引导式学习会话管理等。

该分层使“教学业务逻辑”与“模型推理与能力编排”解耦：后端负责权限、数据与流程治理，AI 服务负责生成与可验证能力落地，前端负责交互与可解释呈现。

3.3.2　关键业务流程

以一次“学生提问/辅导”为例，核心链路可概括为：前端提交问题与上下文 → 后端鉴权与权限校验 → AI 服务选择技能（`mode`）并可选启用 GraphRAG（`_rag` 后缀）→ 若需要可验证结果则触发工具调用 → 回注工具结果与引用证据 → 生成最终回答并返回 → 后端记录学习事件与画像更新。该流程将生成式回答纳入“可追溯（证据）+ 可验证（工具）+ 可沉淀（画像/事件）”的数据闭环。

3.4　模块划分与接口约定

为便于后续章节的实现描述与复现，本节给出平台关键能力与模块化落地的对应关系：

表 3.1: 关键能力与实现模块对应关系

能力模块	实现要点（模块化落地）
Skills（提示编排）	技能注册表统一管理系统提示与约束，`mode` 选择技能并支持别名映射；技能可注入课程/作业/画像等上下文。
GraphRAG（可追溯检索）	基于图结构索引进行“种子检索 + 图扩展”，回答强制引用编号；支持混合检索与索引更新与 ACL 过滤。
Tool Calling（可验证执行）	统一工具参数 Schema 与白名单，限制最大调用次数；工具执行结果回注对话上下文用于最终解释。
Guided Learning（过程性辅导）	学习路径生成 + 分步提问推进；会话状态（TTL、用户绑定）维护进度与薄弱点。
训练与评测	chat-style 数据规范、LoRA/QLoRA 微调、离线评测报告；训练前增加蒸馏与 smoke 自检门禁。
学习状态追踪	学习事件流 + 课程画像/全局画像；将薄弱点、完成主题、学习时长等沉淀为可追踪档案。

接口层面，平台对外保持稳定的对话协议：以 `messages` 表示多轮对话，以 `mode` 表示技能选择；当模式名以 `_rag` 结尾时，表示在保持技能风格不变的同时启用检索增强。对于需要工具调用的场景，采用结构化工具列表与 `tool` 回注消息组织对话，使“调用—执行—解释”链路可复现。

3.5　数据与权限模型概述

系统采用 JWT + RBAC 进行鉴权与授权：学生、教师、助教在课程、作业、学习画像等资源上具备不同权限边界。为支持长期学习状态追踪，后端提供课程画像、全局画像与学习事件接口：课程画像用于呈现某门课内的薄弱点与学习进度分布，全局画像用于跨课程归纳学生能力结构，学习事件用于形成可回放的时间线。对于知识库内容，系统在索引与检索环节引入课程维度的访问控制（ACL），避免跨课程材料泄露与越权引用。

3.6　本章小结

本章从应用场景出发给出系统需求与总体设计，明确平台需同时满足多模式辅导、可追溯检索、可验证执行、引导式学习、长期学习追踪以及训练评测闭环等能力，并给出分层架构与模块划分。下一章将进一步围绕 Skills、GraphRAG、工具调用、学习状态追踪与训练评测管线等关键模块展开实现细节。

第四章　系统关键模块设计与实现

4.1　技能系统与对话编排

平台将”提示模板、上下文注入、输出约束、安全规则”封装为可复用的技能单元，并通过注册表集中管理。运行时以 mode 参数选择技能，解决了”提示词散落、行为不可控”的工程问题，使系统具备可维护的提示治理能力。

4.1.1　技能注册与路由

AI 服务层将技能定义为一组可调用的能力模块，并为历史模式名提供别名映射以保持接口兼容。平台同时支持 _rag 后缀约定：当 mode 以 _rag 结尾时，在保持技能教学风格不变的前提下启用 GraphRAG 检索增强，实现”技能（风格）—检索（证据）”的解耦组合。

4.1.2　上下文注入与输出治理

为使回答贴合课程语境且便于前端结构化展示，技能系统在系统提示中约定输出组织方式（例如先给出结论或评价，再展开解释与建议），并在不干扰用户对话的前提下注入必要上下文：课程信息、作业要求、学生档案与检索证据等。对于存在学术不端风险的请求（如要求代写整篇），系统提示引导模型采取拒答或提供提纲与修改建议等稳健策略，将学术诚信约束纳入可治理范围。

4.1.3　已实现技能与能力边界

从工程落地角度，平台将技能分为”通用能力”与”课程专属能力”两类：

(1) **通用能力**：概念讲解、作业反馈、引导式学习与个性化辅导策略等，强调结构化输出与过程性引导，可迁移到不同课程。

(2) **课程专属能力**：以电磁场为例的公式推导与仿真解读；以写作为例的写作类型感知评估与学术规范反馈。课程专属能力通过权限与开关边界控制，避免跨课程误用。

该划分使平台在扩展课程时优先复用通用能力，仅在必要处以课程模块补齐差异化需求。

4.2　GraphRAG 检索增强实现

平台在 AI 服务层实现轻量化 GraphRAG：离线阶段将课程材料切片并构建图结构索引；在线阶段通过混合检索与图扩展获取相关证据片段，并将其注入系统消息，要求回答按引用编号输出。相较传统 RAG，该方案在概念依赖关系较强的任务中更容易覆盖前置概念与相关规范；同时，引用编号为教师复核与错误定位提供了直接抓手。检索阶段支持按课程维度的访问控制，保证不同课程材料隔离。

4.3　工具调用执行器与安全治理

4.3.1　调用流程与消息组织

平台在对话接口中提供工具调用能力：当模型判断需要可验证结果时，生成结构化 `tool_calls`；系统解析后执行工具，并以 `tool` 消息将执行结果回注到对话上下文；随后模型基于工具结果生成最终解释与建议。该”调用—执行—回注—解释”链路使关键数值可复现、可审计。

4.3.2　安全约束与可控性

为避免模型进入无限调用或越权调用状态，平台在工程上引入以下约束：

(1) **工具白名单**：仅允许调用预先注册的工具，未注册名称直接拒绝执行。

(2) **最大调用次数**：每次对话限制最大工具调用轮数。

(3) **超时与降级**：工具执行设置超时；不可用时返回失败信息并引导模型给出替代方案。

(4) **结果回注**：工具输出以结构化文本或 JSON 形式回注，保证后续回答引用的是可复现的外部结果。

4.4　引导式学习与学习状态追踪

平台将引导式学习作为”学生中心”的核心交互方式：系统围绕学习主题生成学习路径，并在多轮对话中维护进度、薄弱点与阶段性目标，实现”诊断—引导—巩固”的过程性辅导。

4.4.1　学习路径生成与会话管理

引导式学习首先生成结构化学习路径（3–6 步），并为每一步附带目标描述、前置概念与是否需要工具验证等标记。系统以 `session_id` 绑定会话与用户身份，并引入有效期与会话数量上限，避免无界增长与越权访问。会话推进过程中，系统维护学习目标、当前步骤、历史对话与步骤完成状态。

4.4.2　薄弱点检测与学习画像

平台对辅导过程中的”纠错/提示”语句进行轻量解析，在负面信号上下文中提取概念薄弱点（如写作中的”逻辑连接”或理工科中的”边界条件”），并在会话内累积统计。后端提供学习画像与学习事件接口：将薄弱点、完成主题与学习时长沉淀为课程画像与跨课程全局画像，并以学习事件流形成可回放时间线，为教师侧学情分析提供数据支撑。

4.4.3　个性化辅导策略生成

在具备学习档案后，系统可基于历史薄弱点与学习进度生成结构化辅导策略，例如 1–2 周学习计划、针对薄弱点的专项练习与推荐主题。该能力同样通过技能系统实现，输出结构化数据便于前端展示与教师复核。

4.5　异构加速服务部署

AI 服务通过 OpenAI-compatible 接口与上游推理服务对接，部署侧支持 GPU、NPU 与 FPGA 三种异构加速后端，分别针对大模型推理与向量检索进行优化。

4.5.1　GPU 推理部署

GPU 方案采用 vLLM 作为推理引擎，加载 Hugging Face 格式模型，支持连续批处理与 PagedAttention 以提升吞吐。部署时通过 Docker 容器化，配置 CUDA 驱动与显存限制即可启动。该方案在 NVIDIA RTX 4090 上可获得较低延迟与较高吞吐。

4.5.2　Ascend NPU 推理部署

为满足国产化与节能需求，平台同时支持华为 Ascend NPU 后端。部署流程如下：

(1) **模型转换**：使用 ATC 将 Hugging Face 模型转换为 Ascend 可执行格式（OM 模型），或使用 MindIE 直接加载权重。

(2) **推理引擎**：采用 MindIE 提供 Python/C++ 推理接口，或通过 vLLM 的 Ascend 后端实现兼容。

(3) **服务封装**：推理服务对外暴露 OpenAI-compatible 接口，AI 服务层无需区分上游是 GPU 还是 NPU 后端。

实测表明，Ascend 910B 在首 token 延迟上与 RTX 4090 处于同一数量级，后续 token 吞吐略低约 15%，但满载功耗降低约 20%。

4.5.3 FPGA 加速 Embedding 服务

在检索增强链路中，Embedding 计算是主要延迟来源之一。当 GPU 被大模型推理占用时，Embedding 若共享 GPU 会产生资源竞争。平台引入 FPGA 加速方案，将 Embedding 计算迁移至 Xilinx Alveo 加速卡，实现资源解耦与延迟优化。

实现流程：

(1) **模型量化**：将 Embedding 模型（如 bge-base-zh、sentence-transformers）通过 Vitis AI 量化工具转换为 INT8 精度。量化过程使用课程语料库的代表性样本进行校准，保证量化后的向量表示与 FP32 版本保持较高余弦相似度（> 0.98）。

(2) **模型编译**：使用 Vitis AI Compiler 将量化模型编译为 FPGA 可执行指令(xmodel)，指定目标设备（如 U50、U250）的 DPU 配置。

(3) **硬件部署**：在 Xilinx Alveo 加速卡上部署 xmodel，通过 DPU（Deep Learning Processing Unit）IP 核执行推理。FPGA 通过 PCIE 与主机通信，使用 VART（Vitis AI Runtime）API 进行调用。

(4) **服务封装**：将 FPGA Embedding 推理封装为 FastAPI 微服务，对外提供与 CPU/GPU Embedding 服务相同的 REST 接口（`/v1/embeddings`），便于 AI 服务层无感切换。

接口设计：

```
POST /v1/embeddings
{
  "model": "bge-base-zh-fpga",
  "input": ["query text 1", "query text 2"]
}
Response:
{
  "data": [
    {"embedding": [0.1, 0.2, ...], "index": 0},
    {"embedding": [0.3, 0.4, ...], "index": 1}
  ]
}
```

配置切换：AI 服务通过环境变量 `EMBEDDING_BACKEND` 选择后端(`cpu`、`gpu`、`fpga`)，无需修改业务代码即可切换。

4.6 训练、蒸馏与评测管线

为提升模型在教学风格、引用规范、工具调用与引导式学习等能力上的稳定性，平台提供可复现的训练与评测工具链。

4.6.1　数据规范与样本类型

训练数据采用 chat-style JSONL 表示：每条样本包含 `mode` 与多轮 `messages`，与运行时接口保持一致。数据按任务属性分桶组织，典型包括：

(1) **教学风格与结构化输出**：约束”先结论—再解释—再建议”的组织方式，保证格式可解析。

(2) **RAG 引用约束**：注入证据片段并要求仅基于片段作答，强制标注引用编号，证据不足时追问或拒答。

(3) **工具调用样本**：包含工具调用请求与结果回注，使模型学会在需要可验证结果时触发工具并正确解释。

(4) **拒答与追问**：对缺参、超范围或学术不端请求给出稳健行为。

数据规范化的直接收益是训练与上线协议对齐，减少格式漂移导致的线上不稳定。

4.6.2　LoRA/QLoRA 微调与产物管理

平台采用 LoRA/QLoRA[5-6] 对基座模型进行参数高效微调，并将产物以 adapter 形式输出。训练侧同时输出 adapter 权重、训练配置与日志；支持训练完成后自动生成预测与评测报告，形成可对比的版本迭代依据。

4.6.3　数据蒸馏与质量门禁

在启动微调前，平台引入”蒸馏 + smoke”作为前置门禁：将 chat-style 数据蒸馏为易于审阅的格式并统计去重率；随后以分钟级轻量指标验证数据链路可用，用于发现空样本、重复激增或字段缺失等问题。smoke 指标仅用于链路检查，不用于宣称最终效果。

4.7　课程示例模块

4.7.1　电磁场：数值仿真与推导验证

对电磁场类推导与计算任务，平台可扩展数值仿真与可视化能力，并通过工具调用将关键计算交由可执行组件完成。例如，采用有限差分法求解二维静电场（Laplace/Poisson 方程）或同轴线电容场分布，输出字段数据与可视化图像，再由模型结合引用证据与仿真结果进行解释与引导，形成”计算—解释—理解”的教学闭环。

4.7.2　研究生专业英文写作：结构化评价与规范反馈

对写作类课程，系统以结构化评价维度（rubrics）组织反馈，强调问题定位、修改顺序与可执行建议；对引用规范、段落结构与格式等硬约束项，可交由工具进行规则化检查，减少凭空建议的风险。技能提示中显式加入学术诚信约束，默认不代写整篇内容，而以示范片段、段落框架与改写建议辅助学生完成迭代。

4.8　本章小结

本章围绕平台工程落地给出关键模块的设计与实现：以技能系统进行能力编排与治理；以 GraphRAG 提供可追溯证据链；以工具调用提供可验证执行链；以引导式学习与学习画像实现过程性辅导与长期追踪；异构加速层支持 GPU、NPU 与 FPGA 三种后端——GPU/NPU 用于大模型推理，FPGA 用于 Embedding 服务的低延迟加速与资源解耦；训练、蒸馏与评测管线支撑能力迭代。下一章将从测试与实验评估角度验证系统方案与实现效果。

第五章　测试与实验评估

5.1　评估维度与指标设计

本章围绕四个维度对平台进行评估：功能正确性验证系统各模块能否按预期工作；可追溯性验证答案能否溯源至检索证据；可验证性验证工具调用结果是否被正确引用；可迭代性验证回归评测能否有效约束迭代风险。具体评估指标包括：

(1) **功能正确性**：鉴权与权限边界是否正确、核心业务流程能否正常运行。

(2) **引用一致性**：答案中的引用编号是否与检索片段一一对应；证据不足时系统是否触发追问或拒答。

(3) **工具调用可用性**：模型是否在合适时机触发调用、参数能否正确解析、执行结果是否被准确解释。

(4) **结构化输出稳定性**：JSON 或分点列表是否可解析，便于前端渲染与教师复核。

(5) **学习追踪完整性**：会话状态能否正确恢复、薄弱点与学习时长能否持续沉淀。

5.2　测试环境配置

5.2.1　硬件环境

表 5.1 列出了测试所用硬件。为对比 GPU、NPU 与 FPGA 三种异构加速后端，分别部署对应推理服务。

表 5.1: 测试硬件配置

配置项	GPU 环境	NPU 环境	FPGA 环境
服务器 CPU	Intel Xeon 8 核	鲲鹏 920 64 核	Intel Xeon 8 核
内存	32 GB	64 GB	32 GB
加速卡	RTX 4090 24GB	Ascend 910B 64GB	Xilinx Alveo U50
加速卡功耗	350 W（满载）	280 W（满载）	75 W（满载）
存储	NVMe SSD 512 GB	NVMe SSD 1 TB	NVMe SSD 512 GB

5.2.2　软件环境

表 5.2: 测试软件配置

组件	GPU 环境	NPU 环境	FPGA 环境
操作系统	Ubuntu 22.04	openEuler 22.03	Ubuntu 22.04
LLM 推理框架	vLLM 0.4.x	MindIE 1.0	—
Embedding 框架	sentence-transformers	sentence-transformers	Vitis AI 3.5
Python	3.11	3.9	3.10
基座模型	Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct	—
Embedding 模型	bge-base-zh (FP32)	bge-base-zh (FP32)	bge-base-zh (INT8)

5.3　功能测试

5.3.1　核心功能用例

表 5.3 汇总了各模块的功能测试结果。测试覆盖用户鉴权、权限控制、对话服务、检索增强、工具调用、引导式学习、学习画像与写作分析等关键链路。

表 5.3: 核心功能测试结果

模块	测试内容	预期	结果
用户鉴权	持有效 JWT 访问受保护接口	200 OK	通过
权限控制	学生角色访问教师接口	403 Forbidden	通过
对话服务	tutor 模式概念解释	结构化回答	通过
检索增强	tutor_rag 模式带引用回答	引用编号正确	通过
工具调用	数值计算触发工具	返回可验证结果	通过
引导式学习	创建会话并推进步骤	状态更新	通过
学习画像	查询课程画像接口	返回薄弱点列表	通过
写作分析	提交摘要获取反馈	多维度评分	通过

5.4　异构加速性能对比

本节对比 GPU、NPU 与 FPGA 三种后端在大模型推理与 Embedding 计算两个环节的性能表现。

5.4.1 大模型推理：GPU vs NPU

测试任务为单轮对话，输入 prompt 约 500 token，输出约 200 token。

表 5.4: GPU 与 NPU 大模型推理性能对比

指标	RTX 4090	Ascend 910B
首 token 延迟（TTFT）	180 ms	210 ms
后续 token 吞吐	42 token/s	36 token/s
单请求端到端延迟	4.9 s	5.6 s
满载功耗	350 W	280 W

Ascend 910B 的首 token 延迟与 RTX 4090 处于同一数量级，吞吐略低约 15%，但功耗降低约 20%，适合国产化与绿色部署场景。

5.4.2 Embedding 计算：CPU vs GPU vs FPGA

为评估将 Embedding 计算迁移至 FPGA 后的性能影响，设计以下对照组：

表 5.5: Embedding 后端对照组设计

组别	Embedding 执行位置	精度	备注
G0（基线）	CPU（sentence-transformers）	FP32	最小可用方案
G1	GPU（sentence-transformers）	FP32	观察 GPU 竞争
G2（目标）	FPGA（Vitis AI）	INT8	仅替换 Embedding

测试任务为批量 Embedding 计算，batch size = 8，文本长度约 128 token。

表 5.6: Embedding 后端性能对比

组别	P50 延迟 (ms)	P95 延迟 (ms)	吞吐 (QPS)	功耗 (W)
G0（CPU）	85	120	12	65
G1（GPU）	15	25	65	180
G2（FPGA）	18	28	55	45

分析：

(1) FPGA 方案（G2）的延迟与 GPU（G1）处于同一数量级，P95 延迟仅高约 12%。

(2) FPGA 吞吐略低于 GPU（约 85%），但功耗仅为 GPU 的 25%，能效比显著更优。

(3) 将 Embedding 从 GPU 迁移至 FPGA 后，GPU 可专注于大模型推理，整体并发能力提升。

5.4.3　端到端 RAG 链路对比

将 Embedding 后端切换后，评估完整 RAG 链路的端到端延迟变化。测试任务为带检索增强的写作规范问答，top_k = 5。

表 5.7: RAG 端到端延迟对比

组别	T_embed(ms)	T_rag_total(ms)	T_e2e(s)
G0（CPU Embed）	85	180	5.2
G1（GPU Embed）	15	110	4.9
G2（FPGA Embed）	18	115	4.95

FPGA 方案的端到端延迟与 GPU 方案基本持平，瓶颈已从 Embedding 转移到 LLM 生成阶段。

5.5　检索质量评估

为验证 FPGA 量化（INT8）是否影响检索质量，使用固定回归集对三种 Embedding 后端进行离线评测。

5.5.1　回归集构成

回归集按写作类型分层采样，共 200 条查询：

表 5.8: 检索质量评测回归集

类型	样本数	说明
文献综述（literature_review）	50	需引用规范片段
课程论文（course_paper）	50	需引用范例
学位论文章节（thesis）	50	需引用结构规范
摘要（abstract）	50	含证据不足样本
合计	200	—

5.5.2　质量指标结果

表 5.9: 检索质量指标对比

组别	Recall@5	引用一致性	拒答准确率	向量余弦相似度
G0（CPU FP32）	88.5%	92.5%	85.0%	—
G1（GPU FP32）	88.5%	92.5%	85.0%	—
G2（FPGA INT8）	87.0%	91.0%	84.0%	0.985

分析：

(1) FPGA INT8 量化后，Recall@5 下降约 1.5 个百分点，引用一致性下降约 1.5 个百分点，在可接受范围内。

(2) INT8 向量与 FP32 向量的平均余弦相似度为 0.985，表明量化对语义表示影响较小。

(3) 拒答准确率基本持平，证据不足时的稳健行为未受量化影响。

5.6　离线评测与回归机制

为支撑模型与提示策略的持续迭代，系统采用” 固定回归集 + 指标报告 + 案例分析” 的评测范式。

5.6.1　回归集构成

回归集按任务类型组织，当前规模 120 条，覆盖平台主要能力：

表 5.10: 模型能力回归集构成

任务类型	样本数	评测重点
tutor（概念讲解）	30	结构化输出、教学风格
grader（作业反馈）	25	评价维度、可执行建议
guided_learning	20	路径合理性、步骤推进
tool_calling	15	调用时机、参数正确性
rag_citation	20	引用一致性、证据相关性
refusal/followup	10	拒答与追问稳健性
合计	120	—

5.6.2　评测指标

表 5.11 给出当前版本在回归集上的各项指标。

表 5.11: 离线评测指标

指标	数值	说明
引用一致性	92.5%	引用编号与检索片段对应
工具调用准确率	88.0%	调用时机正确且参数可解析
结构化输出可解析率	95.0%	JSON/分点格式正确
拒答/追问正确率	85.0%	证据不足时采取稳健策略
教学风格符合度	90.0%	人工评分

5.7　典型场景案例

5.7.1　电磁场场景

以静电场边界条件为例，展示系统的” 证据链 + 计算链” 输出方式。

学生提问：请解释平面电容器两介质分界面上的电场边界条件。

系统检索：检索到 3 个相关片段——[1] 边界条件定义、[2] 法向分量连续性、[3] 切向分量连续性。

系统回答（节选）：

在两种介质接触面上，电场需满足以下条件：

1. 法向分量：$D_{1n} = D_{2n}$（无自由面电荷时）[1][2]

2. 切向分量：$E_{1t} = E_{2t}$ [3]

这表明……（后文省略）

上述回答中，引用编号 [1][2][3] 与检索片段一一对应，结论基于证据生成，未出现凭空编造现象。

5.7.2　写作场景

以论文摘要评估为例，展示结构化反馈能力。

学生提交：一段约 200 词的英文摘要草稿。

系统反馈（节选）：

写作类型：摘要（Abstract）

综合评分：72/100

分维度评估：

- 学术语气：7/10 ——建议将 “really important” 改为 “significant”
- 段落结构：8/10 ——背景-方法-结果-结论框架完整
- 逻辑连接：6/10 ——第 2–3 句之间缺少过渡
- 引用规范：N/A（摘要通常不含引用）

修改优先级：1. 逻辑连接 → 2. 学术语气

该反馈按评估框架输出，建议具有可操作性，便于学生按优先级逐项修改。

5.8　本章小结

本章从功能测试、性能测试与离线评测三个层面对平台进行验证。功能测试覆盖鉴权、权限、对话、检索增强、工具调用、引导式学习、学习画像与写作分析等关键

链路，全部通过。异构加速性能测试对比了 GPU（RTX 4090）、NPU（Ascend 910B）与 FPGA（Alveo U50）三种后端：NPU 在大模型推理中首 token 延迟与 GPU 接近，功耗降低 20%；FPGA 在 Embedding 计算中延迟与 GPU 持平，功耗仅为 GPU 的 25%，且将 Embedding 从 GPU 解耦后可提升整体并发能力。检索质量评估表明，FPGA INT8 量化对 Recall@5 的影响约 1.5 个百分点，处于可接受范围。离线评测在回归集上验证了引用一致性 92.5%、工具调用准确率 88.0% 等指标，表明系统在可追溯与可验证方面达到预期目标。

第六章　总结与展望

6.1　工作回顾

本文针对多课程教学场景中” 反馈成本较高、学习过程难以追踪、生成式模型可信度不足” 等问题，设计并实现了一套以学生为中心的智能教学平台。下面从四个方面对所做工作进行回顾。

6.1.1　系统架构与工程实现

平台采用前后端分离与服务化架构：前端以 React + TypeScript + Vite 构建 Web/H5 客户端，可嵌入企业微信 WebView 并保持跨端一致体验；后端基于 Go + Gin 提供业务 API，使用 JWT 进行无状态认证，配合 RBAC 模型实现多角色权限管理；AI 服务以 Python + FastAPI 实现，通过 OpenAI-compatible 接口与上游推理服务对接。代码采用 Monorepo 组织，前后端共享类型定义，降低接口不一致风险。在异构加速方面，系统支持 GPU、NPU 与 FPGA 三种后端——GPU/NPU 用于大模型推理，FPGA 用于 Embedding 服务的低延迟加速与资源解耦。

6.1.2　可追溯与可验证的智能辅导

为应对大语言模型的幻觉与可信度问题，平台构建了” 证据链 + 计算链” 的双重可追溯机制：

(1) **GraphRAG 检索增强**：将课程讲义与规范构建为图结构索引，采用混合检索与图扩展获取相关证据，回答中以编号形式标注来源，使结论可追溯。

(2) **工具调用**：对数值计算、仿真与格式检查等任务，通过工具调用获取可执行结果，降低模型” 心算失误” 与” 凭空建议” 的风险。

6.1.3　过程性辅导与学习追踪

平台以引导式学习为核心交互方式，将复杂学习主题拆解为可管理的步骤，逐步提问推进，实现” 诊断—引导—巩固” 的过程性辅导。同时，通过学习事件流与多级学习画像（课程画像/全局画像）将薄弱点、学习时长与完成主题等信息沉淀为可查阅档案，为教师侧学情分析与个性化干预提供数据支撑。

6.1.4　训练与评测链路

为支持模型定制与持续迭代，平台提供完整的后训练工具链：数据规范化确保训练与上线协议一致；数据蒸馏与 smoke 验证作为质量门禁；LoRA/QLoRA 参数高效微调输出可版本化的 adapter；离线回归评测通过固定评测集量化引用一致性、工具调用准确率与结构化输出稳定性等指标。

6.2　主要贡献

本文的主要贡献可归纳为以下五点：

(1) 提出并落地”学生中心数据闭环 + GraphRAG 可追溯 + 工具调用可验证 + 引导式学习”的一体化框架，形成可复用的工程路径。

(2) 设计”通用能力 + 课程专属模块”的模块化策略，以电磁场与研究生专业英文写作为例验证跨课程可迁移性。

(3) 对比 GPU 与 NPU 两种大模型推理后端的性能与能耗，为校园场景下的国产化与绿色部署提供参考。

(4) 引入 FPGA 加速 Embedding 服务，实现检索链路与大模型推理的资源解耦，在保持检索质量的前提下显著降低能耗。

(5) 建立面向持续迭代的训练与回归评测流程，使模型迭代从主观判断转向可量化回归。

6.3　不足与改进方向

尽管本文完成了平台原型实现，仍存在以下局限：

6.3.1　当前不足

(1) **知识图谱粒度**：当前 GraphRAG 以文档片段为节点，对概念、公式与物理量之间细粒度关系的建模仍有欠缺。

(2) **评测集规模**：回归集约百条量级，难以覆盖所有边界情况；需进一步扩展并引入人工评测。

(3) **真实场景验证**：当前验证主要基于模拟数据，尚未在真实课程中大规模部署并收集师生反馈。

(4) **企业微信深度集成**：OAuth 与消息推送能力处于预留状态，未完成完整的内嵌体验。

(5) **FPGA 量化损失**：INT8 量化导致 Recall@5 下降约 1.5 个百分点，对高精度场景需进一步优化量化策略。

6.3.2 后续工作

针对上述不足，可从以下方向继续改进：

(1) **知识抽取与融合**：引入实体关系抽取与图数据库（如 Neo4j），提升知识图谱的细粒度与可维护性。

(2) **评测体系扩展**：构建千条量级的课程问答评测集，量化准确率与教学效果；引入 A/B 测试框架。

(3) **检索与上下文优化**：优化检索排序与上下文压缩策略，降低延迟并提升长上下文任务的稳定性。

(4) **企业微信完整集成**：完成 OAuth 认证与消息推送能力，提升触达体验。

(5) **多模态能力**：引入图像理解，支持手写作业识别与公式图片解析。

(6) **学习分析可视化**：为教师提供班级学情看板，展示薄弱点分布与能力发展趋势。

(7) **异构加速生态完善**：跟进 Ascend 与 Xilinx 软件栈更新，探索 Reranker 的 FPGA 加速与混合精度量化。

6.4 结语

本文完成了一套以学生为中心的智能教学平台原型，在” 可追溯、可验证、可迭代” 的设计目标下，为将大语言模型可控地引入教育场景提供了工程实践参考。通过引入 GPU、NPU 与 FPGA 三种异构加速后端，平台在性能、能耗与资源利用之间取得了更好的平衡。希望本研究能够为高校智能教学系统的建设提供借鉴，并在后续工作中不断完善与推广。

附录 A　系统接口说明

A.1　AI 服务核心接口

表 A.1 列出了 AI 服务层的核心 API 接口。

表 A.1: AI 服务核心接口说明

接口路径	方法	功能说明
/v1/chat/completions	POST	OpenAI-compatible 对话接口，支持 mode 选择技能，支持 _rag 后缀启用检索增强
/v1/writing/analyze	POST	写作样本分析，返回多维度评估结果
/v1/guided/start	POST	创建引导式学习会话
/v1/guided/step	POST	推进学习步骤
/v1/graphrag/index	POST	添加或更新知识库索引
/v1/graphrag/index	DELETE	删除指定索引条目
/health	GET	服务健康检查

A.2　后端业务接口

表 A.2 列出了后端业务层的核心 API 接口。

表 A.2: 后端业务核心接口说明

接口路径	方法	功能说明
/api/auth/login	POST	用户登录，返回 JWT Token
/api/auth/register	POST	用户注册
/api/courses	GET/POST	课程列表获取与创建
/api/courses/:id	GET/PUT/DELETE	单个课程的查询、更新与删除
/api/assignments	GET/POST	作业列表与创建
/api/submissions	GET/POST	作业提交列表与创建
/api/ai/chat	POST	AI 对话代理（转发至 AI 服务）
/api/profile/course/:id	GET	课程学习画像查询
/api/profile/global	GET	全局学习画像查询
/api/events	GET/POST	学习事件查询与记录

附录 B　关键代码片段

B.1　技能注册与路由

以下代码展示了技能系统的注册与路由机制（Python）：

```
# skills/registry.py

SKILLS = {
    "tutor": TutorSkill(),
    "grader": GraderSkill(),
    "guided_learning": GuidedLearningSkill(),
    "writing_feedback": WritingFeedbackSkill(),
    "personalized_strategy": PersonalizedStrategySkill(),
}

# 别名映射，保证向后兼容
ALIASES = {
    "answer": "tutor",
    "explain": "tutor",
    "grade": "grader",
}

def get_skill(mode: str) -> BaseSkill:
    """根据 mode 获取技能实例，支持 _rag 后缀"""
    base_mode = mode.replace("_rag", "")
    resolved = ALIASES.get(base_mode, base_mode)
    return SKILLS.get(resolved, SKILLS["tutor"])
```

B.2　GraphRAG 检索流程

以下代码展示了 GraphRAG 的混合检索与图扩展逻辑（Python）：

```
# graphrag/retrieve.py
```

```
def hybrid_retrieve(query: str, top_k: int = 5) -> List[Chunk]:
    # 1. 关键词检索
    keyword_results = keyword_search(query, top_k=top_k*2)
    # 2. 语义向量检索
    semantic_results = vector_search(query, top_k=top_k*2)
    # 3. RRF 融合排序
    fused = reciprocal_rank_fusion(
        [keyword_results, semantic_results], k=60
    )
    return fused[:top_k]

def graph_expand(chunks: List[Chunk], hops: int = 1) -> List[Chunk]:
    """从种子片段出发，沿知识图谱边扩展"""
    expanded = set(chunks)
    for chunk in chunks:
        neighbors = get_neighbors(chunk.id, max_hops=hops)
        expanded.update(neighbors)
    return list(expanded)
```

B.3　工具调用执行器

以下代码展示了工具调用的执行与结果回注逻辑（Python）：

```
# tools/executor.py

TOOL_WHITELIST = {"calculate", "simulate", "check_format"}
MAX_TOOL_CALLS = 3

async def execute_tool_calls(
    tool_calls: List[ToolCall],
    context: ChatContext
) -> List[ToolResult]:
    results = []
    for i, call in enumerate(tool_calls):
        if i >= MAX_TOOL_CALLS:
            break
        if call.name not in TOOL_WHITELIST:
            results.append(ToolResult(
                call_id=call.id,
```

```
                error=f"Tool {call.name} not allowed"
            ))
            continue
        try:
            result = await TOOLS[call.name].execute(call.arguments)
            results.append(ToolResult(
                call_id=call.id,
                content=result
            ))
        except TimeoutError:
            results.append(ToolResult(
                call_id=call.id,
                error="Tool execution timed out"
            ))
    return results
```

附录 C　部署指南

C.1　环境要求

- Docker 24.0 及以上
- Docker Compose 2.x 及以上
- 推理服务需 NVIDIA GPU（推荐 RTX 4090 或同级）
- 内存建议 32GB 以上

C.2　快速部署步骤

1. 克隆代码仓库并进入 `code/` 目录
2. 复制 `.env.example` 为 `.env` 并配置环境变量
3. 执行 `docker-compose up -d` 启动所有服务
4. 执行 `docker-compose logs -f` 查看日志确认启动成功
5. 访问 `http://localhost:3000` 进入前端应用

C.3　服务端口说明

表 C.1: 服务端口分配

服务	说明	端口
frontend	React 前端	3000
backend	Go 后端	8080
ai_service	AI 服务	8000
simulation	仿真服务	8001
mysql	数据库	3306

参考文献

[1] VASWANI A, SHAZEER N, PARMAR N, et al. Attention Is All You Need[C/OL]//Advances in Neural Information Processing Systems: vol. 30. 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[2] OpenAI. ChatGPT: Optimizing Language Models for Dialogue[EB/OL]. 2022. https://openai.com/blog/chatgpt.

[3] LEWIS P, PEREZ E, PIKTUS A, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks[C/OL]//Advances in Neural Information Processing Systems: vol. 33. 2020. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[4] HOGAN A, BLOMQVIST E, COCHEZ M, et al. Knowledge Graphs[M/OL]. Morgan & Claypool Publishers, 2020. https://www.morganclaypool.com/doi/abs/10.2200/S01045ED1V01Y202009DSK016.

[5] HU E J, SHEN Y, WALLIS P, et al. LoRA: Low-Rank Adaptation of Large Language Models[J/OL]. arXiv preprint arXiv:2106.09685, 2021. https://arxiv.org/abs/2106.09685.

[6] DETTMERS T, PAGNONI A, HOLTZMAN A, et al. QLoRA: Efficient Finetuning of Quantized LLMs[J/OL]. arXiv preprint arXiv:2305.14314, 2023. https://arxiv.org/abs/2305.14314.