选课系统程序设计报告

组长：杨天琢 组员：徐健峰 郑晓霏

一、程序功能介绍

本项目设计并实现了一个基于 PyQt5 的图形化“智能选课系统”，专为北京大学计算机方向学生设计。系统集成了 DeepSeek 智能评估功能，旨在提高学生选课效率，通过智能推荐、课程评价、自动排课、冲突检测等功能，帮助学生做出合理的选课决策。

系统主要功能包括：

1. 智能选课流程管理

系统实现了完整的 9 步选课流程，从欢迎界面开始，依次进行年级选择、专业选择、必修课选择、选择性必修课选择、通识课选择、教师推荐、AI 智能评估，最终生成完整的课表。系统支持电子信息与技术、信息与计算科学、智能科学与技术、应用物理学、计算机科学与技术、通班等 6 个专业方向（如图一），覆盖大一到大四各年级的个性化选课需求。

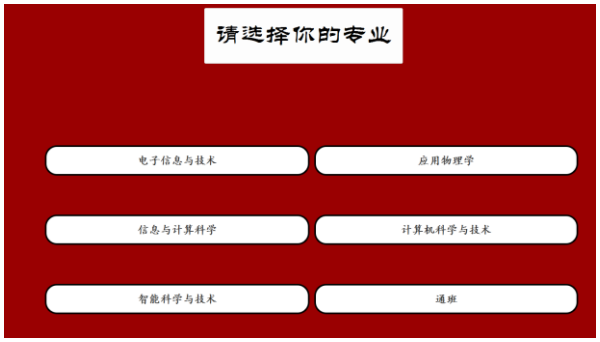


图 1　专业选择

2. AI 智能评估系统

系统集成了 DeepSeek 大语言模型 API，能够提供专业的选课建议。AI 系统会智能分析课程结构合理性、学分分配、专业匹配度等多个维度，基于学生的年级和专业背景提供个性化的学习建议。系统还提供短期、中期、长期的学习规划指导，帮助学生制定合理的学习计划（如图二）。

图二 AI 智能评估

3. 课程信息管理与筛选

系统支持从 Excel 文件中动态加载课程数据，展示课程名称、代码、教师、时间安排、学分等基础信息。系统能够根据培养方案智能筛选必修课、选择性必修课与通识课，并提供实时的学分统计和课程数量统计功能（如图三）。我们集成了课程评分显示功能，帮助学生了解课程的历史评价，避免选择质量较差的课程。

图三 通识课选择

4. 智能排课与冲突检测

系统采用时间段重叠检测算法来判断课程时间冲突，能够自动识别并提示时间冲突的课程组合。排课系统会优先推荐时间无冲突的课程组合，并提供课程组合建议以提高选课效率。我们设计了可视化课程表生成功能，将选课结果以清晰的表格形式展示。

5. 教师推荐系统

系统基于课程类型和专业特点提供智能教师推荐功能。推荐算法会结合 AI 分析结果，考虑教师的教学风格、课程评价、专业匹配度等因素，为学生提供个性化的教师建议（如图四）。

图四 教师推荐

6. 现代化用户界面

我们整体采用了 PyQt5 框架构建，采用模块化布局设计，使用.ui 文件与 Python 脚本分离的开发模式，提高了界面开发效率。界面设计采用 Material Design 风格，配备专业的配色方案，支持淡入淡出动画效果和按钮悬停交互。

7. 系统工具与调试

在不能显示的部分，我们确保了系统的完整性。我们设计了系统状态检查工具，能够全面检测系统运行状态；AI 服务连接测试工具确保 API 服务的可用性；还包含课程重名检查、时间段重叠检查、界面按钮错位检测等专业工具。

二、项目各模块与类设计细节

1. 核心架构模块

1.1 主程序模块（main_enhanced.py）

主程序模块实现了完整的选课流程控制逻辑，采用状态机模式管理 9 个选课步骤，确保流程的连贯性和稳定性。模块包含全局用户数据管理功能，负责维护学生的年级、专业、已选课程等关键信息。系统设计了动画效果基类 AnimatedDialog，为所有界面提供统一的动画支持，各界面类继承自该基类，实现了统一的用户体验。主程序还负责协调各个模块之间的数据传递和状态同步。

1.2 配置管理模块（config.py）

配置管理模块集中管理 API 密钥和系统配置，支持 DeepSeek、OpenAI、千问、智谱等

多种 AI 服务配置。模块提供了统一的 UI 样式配置，包括颜色主题、字体设置、动画参数等，确保整个系统的视觉一致性。系统还定义了组件样式模板，包括按钮、表格、标签等组件的统一样式，支持高 DPI 显示和跨平台兼容性。

1.3 AI 集成模块（llm_integration.py）

AI 集成模块实现了多 API 服务支持，包括 DeepSeek、OpenAI、千问、智谱等主流大语言模型。模块采用智能 API 调用策略，当主要服务不可用时能够自动切换到备用服务，确保系统的稳定性。模块设计了专业的教育领域提示词，能够生成结构化的评估报告。系统还实现了完善的错误处理和重试机制，提高了 API 调用的成功率。

```
self.api_configs = {
    'deepseek': {
        'base_url': 'https://api.deepseek.com/v1/chat/completions',
        'headers': {
            'Authorization': f'Bearer {get_api_key("deepseek")}',
            'Content-Type': 'application/json'
        }
    },
    'openai': {
        'base_url': 'https://api.openai.com/v1/chat/completions',
        'headers': {
            'Authorization': f'Bearer {get_api_key("openai")}',
            'Content-Type': 'application/json'
        }
    },
    'qwen': {
        'base_url': 'https://dashscope.aliyuncs.com/api/v1/services/aigc/text-generation/generation',
        'headers': {
            'Authorization': f'Bearer {get_api_key("qwen")}',
            'Content-Type': 'application/json'
```

部分 API

2. 智能推荐与调度模块

2.1 AI 推荐系统（ai_recommender.py）

AI 推荐系统基于学生画像实现个性化推荐算法，通过分析学生的年级、专业、兴趣偏好等信息，构建个性化的推荐模型。系统能够提取课程的多维特征，包括工作量、内容质量、考核方式、时间安排等，并计算课程与学生画像的匹配分数。推荐算法会考虑工作量匹配度、内容评分、时间分布合理性、兴趣标签匹配等多个维度，为每门推荐课程生成详细的推荐理由。

```
# 时间安排
available_slots = self.scheduler.get_available_slots()
course_slots = {(slot.day, s)
                for slot in course.time_slots
                for s in range(slot.start_slot, slot.end_slot + 1)}
slot_usage = len(course_slots & available_slots) / len(course_slots)
if slot_usage > 0.8:
    reasons.append("课程时间安排合理，不会与其他课程冲突")

# 兴趣匹配
if course.course_type == '通识课':
    for interest in self.student_profile.interests:
        if interest.lower() in course.name.lower():
            reasons.append(f"课程内容与你的{interest}兴趣相关")
            break
```

AI 逻辑部分代码

2.2 课程调度器（course_scheduler.py）

课程调度器实现了时间槽解析和管理功能，通过 TimeSlot 类表示课程的时间安排，支持复杂的时间段解析。系统采用高效的时间冲突检测算法，能够快速识别课程之间的时间冲突。调度器还实现了可用时间槽计算功能，能够分析当前已选课程的时间占用情况，为后续选课提供时间参考。系统还提供了课程组合推荐策略，能够根据学分要求、时间偏好等因素推荐最优的课程组合。

```
def recommend_courses(self,
                      min_credits: float = 0,
                      max_courses: int = None,
                      preferred_days: List[int] = None) -> List[Course]:
    """推荐不冲突的课程

    Args:
        min_credits: 最小学分要求
        max_courses: 最多推荐课程数
        preferred_days: 优先推荐的上课日期（1-7表示周一到周日）

    Returns:
        推荐的课程列表
    """
```

课程推荐函数接口

2.3 课程评分模块（course_rating.py）

课程评分模块负责管理课程评价数据，支持多维度评分计算，包括工作量评分、内容质量评分、考核方式评分等。模块实现了评分统计和可视化功能，能够生成课程评分的统计报告。系统还考虑了评价数量的权重，评价数量越多的课程其评分可信度越高。

```
    # 获取可用时间槽
    available_slots = self.get_available_slots()

    # 对可选课程进行评分
    scored_courses = []
    for course in self.available_courses:
        # 检查是否有时间冲突
        if self.check_conflicts(course):
            continue

        # 计算课程评分
        score = 0

        # 1. 基础分：课程学分
        score += course.credit * 10

        # 2. 时间槽利用率
        course_slots = {(slot.day, s)
                        for slot in course.time_slots
                        for s in range(slot.start_slot, slot.end_slot + 1)}
        available_course_slots = course_slots & available_slots
        slot_usage = len(available_course_slots) / len(course_slots)
        score += slot_usage * 20

        # 3. 优先日期加分
```

课程评分片段代码

3. 用户界面模块

3.1 界面基类设计

系统设计了 AnimatedDialog 基类，为所有界面提供统一的动画效果和样式支持。基类实现了淡入淡出动画、按钮悬停效果等交互功能，提升了用户体验。系统采用响应式布局设

计，能够适配不同屏幕分辨率和 DPI 设置，确保在各种显示设备上都有良好的显示效果。

```python
class AnimatedDialog(QtWidgets.QDialog):
    """带有动画效果的对话框基类"""
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        self.setAttribute(QtCore.Qt.WA_TranslucentBackground)
        self.setWindowTitle(APP_CONFIG['name'])

    def showEvent(self, event):
        super().showEvent(event)
        self.fade_in()

    def fade_in(self):
        """淡入动画"""
        self.setWindowOpacity(0)
        self.animation = QPropertyAnimation(self, b"windowOpacity")
        self.animation.setDuration(UI_CONFIG['animation']['fade_duration'])
        self.animation.setStartValue(0)
        self.animation.setEndValue(1)
        self.animation.setEasingCurve(QEasingCurve.OutCubic)
        self.animation.start()
```

部分动画效果

3.2 各功能界面

系统包含 9 个主要功能界面，每个界面都有明确的职责分工。WelcomeDialog 负责系统介绍和启动引导，AgeDialog 和 MajorDialog 分别处理年级和专业选择，CompulsoryChooseUi 负责必修课的选择和管理，OptimalCompulsoryUi 和 OptimalDialog 处理选择性必修课和通识课的选择，TeacherDialog 提供教师推荐功能，EvaluationDialog 展示 AI 智能评估结果，FinalDialog 生成最终的课程安排。

```python
class FinalDialog(FinalDialogClass):
    def __init__(self):
        super().__init__()
        self.setup_final_ui()

    def setup_final_ui(self):
        """设置最终课表界面"""
        # 计算并显示正确的总学分
        compulsory_credits = sum(course['credit'] for course in user_data.get("compulsory_courses", []))
        optional_credits = sum(course['credit'] for course in user_data.get("optional_compulsory_courses", []))
        general_credits = sum(course['credit'] for course in user_data.get("general_courses", []))
        total_credits = compulsory_credits + optional_credits + general_credits
        user_data["total_credits"] = total_credits

        print(f"最终课表学分统计: 必修{compulsory_credits} + 选择性必修{optional_credits} + 通识{general_credits} =

        # 重新连接完成按钮，确保程序正确退出
        self.finish_button.clicked.disconnect()  # 断开原有连接
        self.finish_button.clicked.connect(self.finish_and_exit)
```

部分 FinalDialog 代码

4. 数据处理模块

4.1 课程数据提取（extract_courses.py）

课程数据提取模块负责从 Excel 文件中读取和解析课程数据，实现了课程信息的标准化处理。模块能够处理不同格式的 Excel 文件，自动识别和映射课程信息的各个字段。系统还实现了专业课程映射功能，能够根据学生的专业自动筛选相关的课程。模块包含完善的

数据验证和错误处理机制，确保数据的准确性和完整性。

4.2 数据管理

系统支持多种数据源格式，包括 Excel、CSV 等常见格式，具备良好的数据兼容性。系统实现了数据缓存机制，提高了数据访问的性能。系统还支持数据的更新和版本管理，能够处理课程信息的变更。

5. 系统工具模块

5.1 测试与检查工具

系统配备了完整的测试与检查工具集。system_check.py 提供全面的系统状态检查功能，能够检测依赖包、文件完整性、模块导入、UI 类、数据加载等各个方面。test_config.py 专门用于配置测试和验证，确保系统配置的正确性。check_ai_service.py 提供 AI 服务连接测试功能，能够验证 API 服务的可用性。check_ui_overlaps.py 用于检查界面布局问题，check_tables.py 验证数据表结构的正确性。

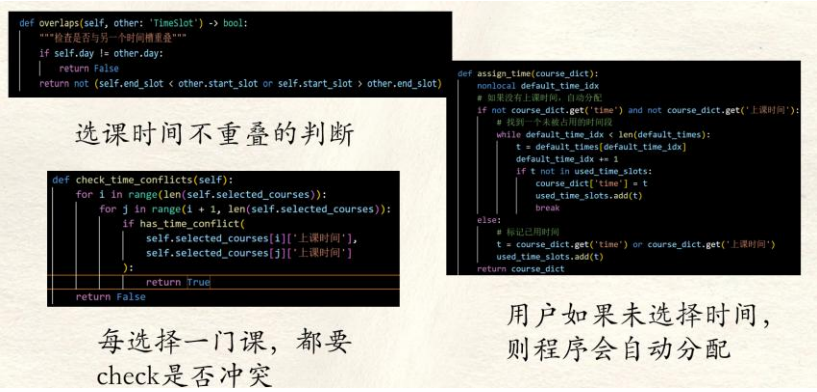
5.2 调试工具

系统提供了丰富的调试工具，包括 debug_main.py 用于主程序调试，debug_ui.py 用于界面调试，以及各种专门的测试脚本用于功能验证。这些工具大大提高了开发效率和系统稳定性。

6. 技术实现细节

6.1 核心算法

系统的冲突检测算法通过遍历所有课程的时间槽，检查是否存在时间重叠来判断冲突。算法的时间复杂度为 $O(n^2)$，其中 n 为课程数量，在实际应用中能够满足性能要求。课程推荐算法采用多维度评分机制，综合考虑工作量匹配度、内容评分、时间分布、兴趣匹配等因素，通过加权计算得出最终的推荐分数。



部分选课逻辑代码

6.2 架构设计模式

系统采用模块化设计模式，每个功能模块都有明确的职责分工，便于维护和扩展。界面组件采用组合模式而非继承模式，提高了代码的灵活性和可维护性。AI 服务调用采用策略模式，支持多种 API 服务的动态切换。系统还实现了观察者模式，当用户数据发生变化时能够自动通知相关界面进行更新。

6.3 错误处理机制

系统实现了完善的错误处理机制，包括异常捕获、用户友好的错误提示、系统自动恢复等功能。系统还提供了详细的日志记录功能，便于问题定位和调试。当关键服务不可用时，系统能够自动切换到备用方案，确保功能的连续性。

三、小组成员分工情况

组长：杨天琢

- 负责项目整体架构设计和程序框架搭建
- 实现核心算法和 AI 集成功能
- 开发主程序流程控制和数据管理模块
- 负责项目视频剪辑和最终展示

组员：徐健峰

- 负责程序精调和性能优化
- 实现界面美化和用户体验优化
- 开发动画效果和交互反馈功能
- 负责系统测试和错误修复

组员：郑晓霏

- 负责数据收集与处理
- 实现课程数据提取和管理模块
- 开发 PPT 制作和报告撰写
- 负责用户需求分析和功能验证

四、项目总结与反思

4.1 项目成果

本项目成功实现了一个功能完整的智能选课系统。我们认为我们的技术创新性主要在于成功集成了 DeepSeek AI 大语言模型，实现了智能化的选课建议功能；我们采用现代化的 UI 的设计理念，界面设计美观且功能实用。在设计中我们还开发了完善的检查和调试工具，大大提高了系统的稳定性和可维护性。

在功能完整性方面，我们实现了完整的 9 步选课流程，覆盖了从课程选择到最终课表生成的全过程；支持 6 个专业方向和 4 个年级的个性化选课需求，能够满足不同学生的选课需求。我们设计的系统采用模块化设计理念，代码结构清晰，各个模块职责明确，便于后续的维护和功能扩展。同时，我们学习并实现了配置化管理，支持灵活的部署和定制，能够适应不同的使用环境。

4.2 技术亮点

AI 技术应用是本项目的核心亮点之一。我们用 prompt 的形式成功将大语言模型技术应用于教育领域，实现了智能化的课程分析和建议功能。我们的系统系统能够分析课程结构的合理性，评估学分分配的合理性，判断专业匹配度等多个维度。

我们使用了 pyqt5 作为开发和精调软件，界面美观且功能实用——在这个基础上，我们实现了丰富的动画效果，包括淡入淡出动画、按钮悬停效果等，提升了用户的交互体验。在开发过程中，我们注意到不同显示器不同分辨率和相对位置的问题，因此我们采用响应式布局设计，能够适配不同分辨率的显示设备。

在初始设计时，我们采用了清晰的模块划分，各个模块职责明确，降低了模块间的耦合度，在智能系统的帮助下，我们实现了完善的配置管理机制，支持灵活的部署——从而

实现了强大的扩展性，能够方便地添加新的功能和模块。

4.3 项目反思与改进方向

在技术改进方面，我们承认现在的选课系统还有较大的提升空间。AI 推荐算法的准确性可以进一步优化，通过引入更多的数据源和更复杂的算法模型，提高推荐的精准度。未来可以增加更多的数据源，包括更多的课程信息、学生评价数据等，为推荐算法提供更丰富的数据支持和微调。

我们也有考虑有关记忆性的调整策略，比如增加用户偏好学习功能，通过分析用户的历史选课行为，学习用户的偏好模式，提供更加个性化的推荐。在不同用户的协同方面，我们考虑开发课程评价和反馈系统，让学生能够分享课程体验，为其他学生提供参考。

同时，我们也反思了有关我们团队协作的问题：项目在初期需求分析阶段存在一些不足，导致后期功能调整较多。但所幸在项目开发过程中我们设计了完善的测试和调试工具，大大提高了系统的稳定性和可靠性。

4.4 项目价值与意义

我们的初衷是为北京大学的学生提供智能化的选课工具，能够帮助学生做出更合理的选课决策。我们认为，这次的 pj 展示了我们对 AI 技术在教育领域的应用的信心。

在个人层面上，我们收获了很多宝贵的实战经验。在开发过程中，我们不仅学会了如何将 AI 技术真正落地应用，还掌握了现代界面开发的各种技巧。从最初的需求讨论，到系统架构设计，再到代码编写和调试测试，每个环节都让我们对软件工程有了更深刻的理解。

4.5 总结

虽然受限于工作量和工作时长，我们的项目尚且稚嫩，但客观来说相对成功地实现了一个功能完整的智能选课系统，满足了基本的课程要求和学生诉求。