都队_工程报告

一、项目基本信息

队名： 都队

团队成员：

- 穆佳月
- 宋国庆
- 钱伟

Python代码行数： 10,100行（核心代码，精确到百行）

说明： 开发过程中编写了约13,000行代码（包含3,000行辅助脚本用于数据初始化、bug修复、功能集成等），最终交付版本保留10,100行核心代码，体现了工程化的代码管理思想。

主要应用框架及版本：

框架/库	版本号	用途说明
PyQt5	**5.15.10**	桌面应用GUI框架，构建所有界面组件
SQLite	**3.x**	轻量级关系型数据库，本地数据持久化
Matplotlib	**3.10.0**	数据可视化库，生成统计图表
NumPy	**2.1.3**	数值计算库，数据处理和分析
Requests	**2.31.0**	HTTP请求库，AI助手API调用
Pillow	**11.1.0**	图像处理库，界面资源处理

开发环境：

- **Python版本**：Python 3.13.5
- **操作系统**：Windows 10/11
- **开发工具**：VS Code / PyCharm
- **版本控制**：Git + GitHub

二、团队分工

钱伟

负责模块：

1. **项目架构设计**：MVC架构搭建、模块化设计规划、数据库表结构设计
2. **核心功能开发**：数据库管理器（db_manager.py）、数据加载工具（data_loader.py）、主窗口框架（main_window.py）
3. **数据库功能**：学习记录重置功能、数据备份与恢复机制

4. **Git版本管理**：分支管理（main、exam、review分支策略）、代码审查与合并、版本发布管理

代码贡献： 约4,500行

穆佳月

负责模块：

1. **错题本重构**：知识点分组卡片设计（mistakes_widget.py +1100行）、卡片复习模式实现、重测功能开发（支持编程题代码执行）、复习设置与进度跟踪
2. **界面优化**：错题本UI重构（左右分栏布局）、卡片UI渐变背景和阴影效果、交互动画优化
3. **AI助手集成**：DeepSeek API对接（ai_assistant_widget.py）、悬浮式助手设计、对话管理与历史记录
4. **多模块优化**：考试模块优化（exam_widget.py）、练习模块优化（practice_widget.py）、统计页面优化（statistics_widget.py）

代码贡献： 约4,800行

宋国庆

负责模块：

1. **知识点学习模块**：知识点展示界面（knowledge_widget.py）、学习进度跟踪、代码示例展示
2. **题库练习系统**：四种题型支持（practice_widget.py）、题目筛选与加载、答题评判逻辑
3. **代码编辑器**：Python代码编辑器（editor_widget.py）、代码执行器（code_executor.py）、超时保护与安全沙箱
4. **数据可视化**：成绩统计图表（statistics_widget.py）、学习进度展示（progress_widget.py）、Matplotlib图表集成
5. **数据初始化脚本**：初始数据脚本（scripts/init_data.py）、考试配置脚本（clean_duplicates_and_reconfigure.py）

代码贡献： 约3,800行

三、项目亮点

1. 智能错题本重构：知识点分组卡片设计，支持卡片式复习模式，编程题重测功能带代码运行验证。

2. PTA风格多测试点判题：3场模拟考试，38个测试点，编程题部分分机制，真实模拟二级考试评分规则。

3. AI学习助手：集成DeepSeek API，领域限定提示词，悬浮式设计，平均响应2-3秒。

四、得意的Python语言特性应用

4.1 装饰器（Decorator）应用

应用场景： 数据库连接管理

```python
def with_database_connection(func):
    """装饰器：自动管理数据库连接"""
    def wrapper(*args, **kwargs):
        db_manager.connect()
        try:
            result = func(*args, **kwargs)
            return result
        finally:
            db_manager.disconnect()
    return wrapper

@with_database_connection
def load_user_statistics(user_id):
    """加载用户统计数据"""
    sql = "SELECT * FROM study_statistics WHERE user_id = ?"
    return db_manager.execute_query(sql, (user_id,))
```

优势： 代码简洁，防止连接泄漏，易于维护。

4.2 上下文管理器（with语句）

应用场景： 文件操作和代码执行

```python
# 代码保存功能
def save_code(self):
    file_path, _ = QFileDialog.getSaveFileName(self, '保存代码', '', 'Python Files (*.py)')
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as f:
            f.write(self.editor.toPlainText())
```

优势： 自动资源管理，异常安全，代码简洁。

4.3 列表推导式与生成器表达式

应用场景： 数据处理和UI生成

```python
# 筛选未掌握的错题
unmastered_mistakes = [
    mistake for mistake in wrong_questions
    if not mistake['mastered']
]

# 按知识点分组
grouped_mistakes = {
    category: [m for m in mistakes if m['category'] == category]
    for category in set(m['category'] for m in mistakes)
}
```

优势： 代码简洁，性能优秀，可读性强。

4.4 字典的高级用法

应用场景： 配置管理和数据映射

```python
# 主题配置管理
THEME_COLORS = {
    'primary': '#0078D7',
    'success': '#28A745',
    'danger': '#DC3545',
}

# 题型对应的判题函数
JUDGE_FUNCTIONS = {
    'choice': judge_choice_question,
    'judge': judge_judge_question,
    'fill': judge_fill_question,
    'code': judge_code_question,
}

# 动态调用判题函数
judge_func = JUDGE_FUNCTIONS.get(question_type)
result = judge_func(user_answer, correct_answer)
```

优势： 配置集中管理，代码灵活，性能优秀（O(1)查找）。

4.5 异常处理与日志记录

应用场景： 数据库操作和代码执行

```python
def execute_query(self, sql, params=()):
    """执行查询，返回结果"""
    try:
        cursor = self.conn.cursor()
        cursor.execute(sql, params)
        return cursor.fetchall()
    except sqlite3.Error as e:
        print(f"❌ 数据库查询错误: {e}")
        return []
    finally:
        if cursor:
            cursor.close()
```

优势： 错误定位准确，程序健壮性强，用户体验好。

4.6 多线程（QThread）应用

应用场景： AI助手API调用

```python
class DeepSeekThread(QThread):
    """DeepSeek API调用线程"""
    response_received = pyqtSignal(str)
    error_occurred = pyqtSignal(str)

    def run(self):
        try:
            response = requests.post(
                "https://api.deepseek.com/v1/chat/completions",
                headers={"Authorization": f"Bearer {self.api_key}"},
                json={"model": "deepseek-chat", "messages": self.messages},
                timeout=30
            )
            content = response.json()["choices"][0]["message"]["content"]
            self.response_received.emit(content)
        except Exception as e:
            self.error_occurred.emit(str(e))
```

优势： 界面不卡顿，实时反馈，安全退出。

五、软件运行中可能遇到的问题

5.1 环境依赖问题

问题1：缺少依赖包

现象： `ModuleNotFoundError: No module named 'PyQt5'`

解决方案：

```
pip install -r requirements.txt
```

问题2：PyQt5安装失败（Windows）

解决方案：

```
# 先升级pip
python -m pip install --upgrade pip

# 使用国内镜像源
pip install PyQt5 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

5.2 数据库相关问题

问题3：数据库文件损坏

现象： `sqlite3.DatabaseError: database disk image is malformed`

解决方案：

```
# 使用备份恢复
cd database
copy python_learning.db.backup python_learning.db
```

问题4：数据库锁定

现象： `sqlite3.OperationalError: database is locked`

解决方案： 关闭其他正在运行的程序实例，重启程序。

5.3 代码执行相关问题

问题5：编程题执行超时

现象： 编程题运行时提示"代码执行超时（5秒）"

解决方案： 检查代码是否有死循环，优化算法复杂度。

5.4 AI助手相关问题

问题6：AI助手无响应

解决方案： 检查网络连接，确认API密钥是否有效。

5.5 界面显示问题

问题7：界面字体过小/过大

解决方案： 进入"个人主页"标签，在"外观设置"中调整字体大小。

六、工具使用情况

VS Code（主要IDE）：轻量级IDE，启动快速，插件丰富（Python扩展、Git集成、Markdown预览），智能代码补全和调试功能强大。

Git + GitHub：三分支策略（main、exam、review），提交规范（feat、fix、docs前缀），总提交50+次，分支合并15+次。

DeepSeek AI API：性价比高，响应速度快（2-3秒），中文优化，稳定性好（99.5%+可用性），月调用约500次。

SQLite Database Browser：可视化查看数据库表结构，手动执行SQL查询测试，轻量级免费开源工具。

Matplotlib 3.10.0：生成准确率饼图和题型分布柱状图，中文字体支持，与PyQt5集成方便，性能优秀（<0.5秒生成图表）。

手动测试流程：功能测试、边界测试、异常测试、兼容性测试。发现并修复12个bug，遗留bug 0个。

七、其它补充

7.1 主要收获

技术能力提升：

1. PyQt5框架掌握：从零到熟练，深刻理解信号槽机制和事件驱动模型，掌握自定义控件开发。错题本卡片复习模式综合运用了QSplitter、QScrollArea和自定义Widget。

2. 数据库设计能力：掌握规范化设计（第三范式）、外键约束和索引优化。设计12张表实现复杂的多对多关系，错题本查询优化后速度提升300%。

3. Git版本控制：实践main/exam/review三分支策略，成功管理50+次提交和15+次分支合并，使用git reset保持历史清晰。

4. API集成经验：理解RESTful API调用机制，使用QThread实现异步编程，优雅处理网络超时和API失效。

项目管理能力：

1. 需求分析与迭代开发：理解学生学习Python的痛点，区分核心功能和扩展功能，采用迭代开发（v1.0基础功能 → v1.1错题本重构 → v1.2 AI助手）。

2. 团队协作：按模块分配任务，使用review分支审查组员代码，成功协调3人团队按时完成项目。

3. 问题解决能力：掌握断点调试、日志分析、异常堆栈追踪等调试技巧，善于查找官方文档和GitHub Issues，举一反三总结通用解决方案。

报告编写：都队
日期：2025年12月26日
Python代码行数：10,100行（核心代码，开发过程约13,000行）
项目地址：https://github.com/mjiayue/111-master