智学未来教育平台

详细说明书

1、 引言

1.1 编写目的

编写本文档的目的是为了使用户了解我们智学未来教育平台详细设计，包括总体设计、程序功能设计和用户界面设计。

1.2 背景

因为与需求开发文档中的背景相同，所以在此文档中进行简述，详情可以在需求开发文档中查看。

为了将智能教育纳入人工智能的应用场景和为教育注入新能量，我们小组基于讯飞人工智能平台提供的技术进行智学未来教育平台的开发。充分的将讯飞人工智能平台提供的技术融入我们平台来减轻老师的教学负担和帮助学生进行高效地学习。

1.3 定义

下文中星火认知大模型 Spark 4.0 Ultra 和 Spark3.5 Max 统称为星火大模型。

1.4 参考资料

[1]召南.《科大讯飞发布星火认知大模型，深度赋能教育等四大领域》.2023.05.07

2、总体设计

2.1 需求概述

因为与概要说明文档中的需求概述相同，详情可以在概要说明文档中查看。

2.2 软件结构

智学未来教育平台的结构分为用户管理、学生功能、老师功能和游客功能，如图 1 所示。

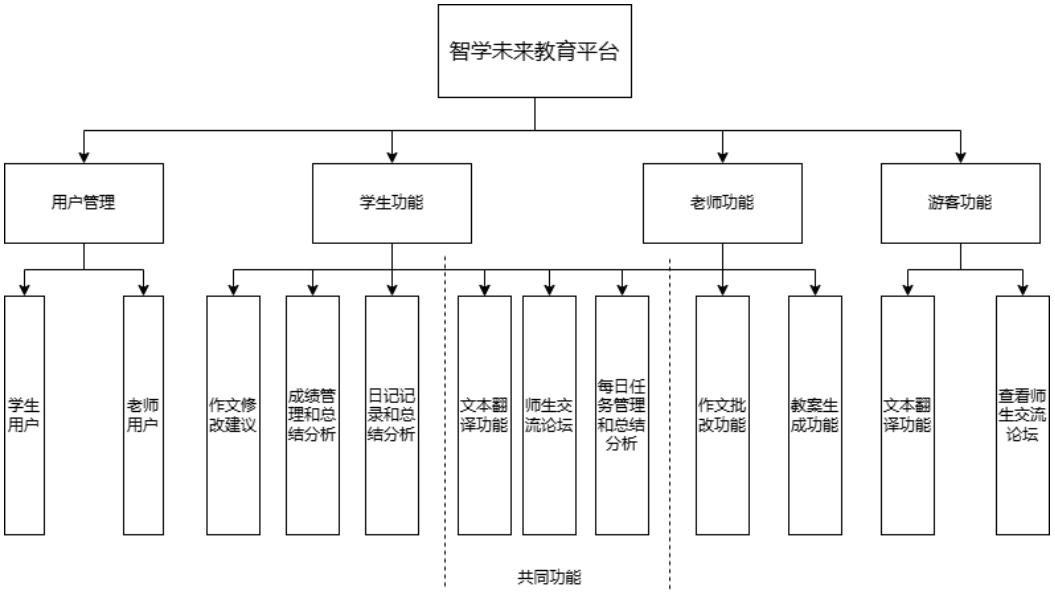

图 1. 智学未来教育平台的总体结构图

3、程序功能设计

3.1 功能实现描述

3.1.1 成绩管理和总结分析功能

(1)、功能设计

前端设计：

本应用使用 React 和 Carbon 组件库开发，旨在提供学生成绩分析功能。

前端主要组件包括输入框、选择框、复选框和按钮，使用 useState 管理应用状态，如用户信息、考试信息、总分和科目选择等。

用户可以通过前端界面输入并保存基础信息和考试信息，触发成绩分析操作并接收 AI 建议。

前端通过 fetch 方法调用后端 API，将用户输入的数据发送到服务器保存和分析。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的请求。

实现三个主要的 API 接口：

保存基础信息（SaveInfoView）

保存考试信息（SaveExamInfoView）

分析考试信息（ExamSummaryView）

后端通过数据库存储和读取用户的基础信息和考试信息，并调用 Spark API 进行数据分析，返回分析结果。

（2）、使用流程

前端使用流程：

编辑科目信息：

学生用户点击“编辑科目信息”按钮进入编辑状态，首先选择教育阶段（小学、初中、高中），然后勾选额外科目（物理、化学、生物、地理、历史、政治），再输入各科满分总分（语文、数学、英语以及选中的额外科目），点击“保存基础信息”按钮保存输入的信息。

查看科目信息：

学生用户点击“查看科目信息”按钮查看已保存的基础信息，包括教育阶段、额外科目和各科总分，然后输入考试信息，在“添加考试成绩”部分输入考试名称、各科成绩（语文、数学、英语以及选中的额外科目）和自我评价，然后点击“添加考试成绩”按钮保存考试信息。

分析成绩：

输入多次考试信息后，用户点击“分析成绩”按钮，触发分析操作。前端调用后端 API 进行分析，并接收 AI 建议显示在页面上。

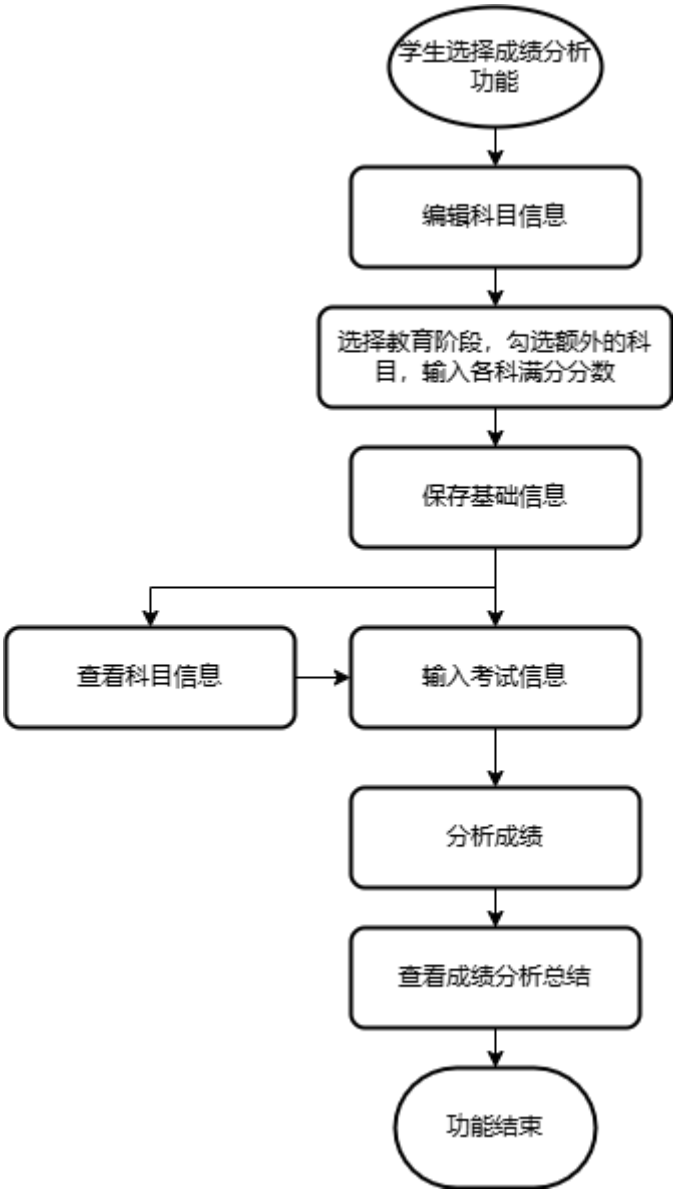

图 2.成绩分析流程图

后端处理流程：

保存基础信息：

后端接收前端发送的基础信息请求，包括学生 ID、教育阶段、科目总分等，然后验证学生 ID 是否存在，若不存在则返回错误响应，存在则将基础信息保存到数据库中，并返回保存成功的响应。

保存考试信息：

后端接收前端发送的考试信息请求，包括学生 ID、考试名称、考试类型、各科成绩和自我评价等，然后验证学生 ID 是否存在，若不存在则返回错误响应。存在则将考试信息保存到数据库中，并返回保存成功的响应。

分析考试信息：

后端获取当前学生的所有考试信息，并组织数据构造查询字符串，调用 Spark API 进行数据分析，并返回分析结果。

（3）、错误提示：

前端错误提示：

1、基础信息保存失败：若保存基础信息失败，控制台会输出错误信息 “保存基础信息失败”。

2、考试信息保存失败：若保存考试信息失败，控制台会输出错误信息 “保存考试信息失败”。

3、分析失败：若分析操作失败，控制台会输出错误信息 “分析失败”。

4、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、缺少学生 ID：若请求中缺少学生 ID，返回错误响应 “缺少学生 ID”。

2、无效的学生 ID：若学生 ID 无效，返回错误响应 “无效的 student_id”。

3、数据库保存失败：若保存基础信息或考试信息失败，返回错误响应及错误信息。

4、分析失败：若调用 Spark API 进行分析失败，返回服务器内部错误响应。

3.1.2 学习计划（每日任务管理）

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，旨在提供用户管理学习任务的功能。

通过 useState 管理应用状态，包括任务列表、当前任务、当前日期、AI 建议等。用户可以添加每天的学习任务，查看任务列表，更新任务状态，并生成学习总结。

前端通过 fetch 方法调用后端 API，将用户输入的数据发送到服务器保存和分析。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的请求。实现三个主要的 API 接口：

获取和添加任务天信息（DayListCreateAPIView）

分析任务信息（AnalyzeTasksAPIView）

更新任务状态（TaskUpdateAPIView）

后端通过数据库存储和读取用户的任务信息，并调用 AI 服务进行数据分析，返回分析结果。

（2）、使用流程

前端使用流程：

获取任务当天信息：

用户打开页面时，会自动调用 fetchDays 方法，从服务器获取任务天信息，并显示在页面上。

添加任务：

用户在页面中输入任务名称和选择任务状态（未开始、进行中、已完成）。然后点击“添加任务”按钮，将任务添加到当前日期的任务列表中。如果任务名称为空，页面会显示错误提示信息。

添加当天任务：

用户输入日期并添加任务后，点击“添加当天任务”按钮，将当天任务保存到服务器。如果日期为空或任务列表为空，页面会显示错误提示信息。

查看和更新任务：用户可以在页面中查看任务列表，并更新任务状态。

任务状态更新会发送 PATCH 请求到服务器，保存更新后的任务状态。

生成学习总结：

用户点击“当天任务总结”按钮，触发分析操作。前端调用后端 API 进行任务分析，并接收 AI 建议显示在页面上。如果没有任务，页面会显示错误提示信息。

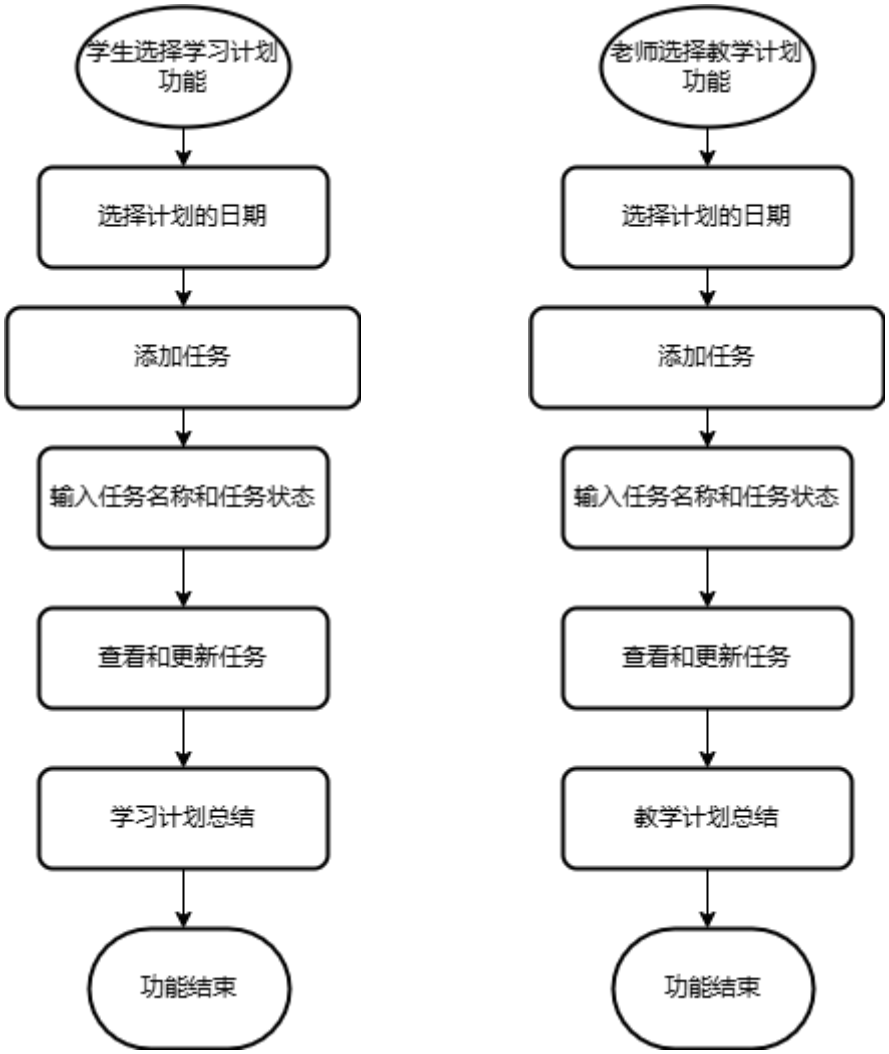

图 3.学生和老师的每日任务管理流程图

后端处理流程：

获取和添加任务天信息：

后端接收前端发送的 GET 请求，返回当前用户的任务天信息。后端接收前端发送的 POST 请求，保存新的任务天信息到数据库中。

分析任务信息：

后端接收前端发送的 POST 请求，获取指定日期范围内的任务天信息。调用 AI 服务进行任务分析，生成学习总结并返回给前端。

更新任务状态：

后端接收前端发送的 PATCH 请求，更新指定任务的状态。

（3）、错误提示

前端错误提示：

1、获取任务天信息失败：如果获取任务天信息失败，控制台会输出错误信息 “获取天数据失败”。

2、添加任务名称为空：如果任务名称为空，页面会显示错误提示 “请输入任务名称”。

3、添加当天任务失败：如果日期为空或任务列表为空，页面会显示错误提示 “请输入日期” 或 “任务不能为空”。

4、分析任务失败：如果分析任务失败，控制台会输出错误信息 “分析失败”。

请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、缺少任务天信息：如果请求中缺少任务天信息，返回错误响应 “No tasks found for the specified date range.”。

2、无效的任务 ID：如果任务 ID 无效或用户无权访问，返回错误响应 “Task not found or not authorized”。

3、数据库保存失败：如果保存任务天信息或更新任务状态失败，返回错误响应及错误信息。

4、分析失败：如果调用 AI 服务进行分析失败，返回服务器内部错误响应。

3.1.3 日记本

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，旨在提供用户管理日记的功能。

通过 useState 管理应用状态，包括日记列表、当前日记、查看模式、排序选项和 AI 分析结果等。用户可以创建新日记、查看日记、保存日记并生成日记总结。

使用 ReactQuill 组件提供富文本编辑功能，用户可以对日记内容进行格式化。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的请求。

实现两个主要的 API 接口：

获取和保存日记信息（DiaryListView）

分析日记信息（DiarySummaryView）

后端通过数据库存储和读取用户的日记信息，并调用 Spark API 进行数据分析，返回分析结果。

（2）、使用流程

前端使用流程：

获取日记列表：

用户打开页面时，会自动调用 fetchDiaries 方法，从服务器获取日记列表，并显示在页面上。

新建日记：

用户点击“新建日记”按钮，进入编辑模式。用户输入日记标题、选择日期、选择心情，并通过 ReactQuill 输入日记内容。点击“保存”按钮，将新日记保存到服务器。

查看日记：

用户点击某篇日记，进入查看模式。然后用户可以查看日记标题、日期、心情和内容。点击“返回”按钮，回到主页。

保存日记：

用户在编辑模式下，点击“保存”按钮，将新日记或更新的日记保存到服务器。如果保存成功，页面返回到主页，显示更新后的日记列表。如果保存失败，页面会显示错误提示信息。

排序日记：

用户可以选择按日期或心情对日记进行排序。排序选项变更时，日记列表会按选定的排序方式重新显示。

生成日记总结：

用户点击“每周日记总结”按钮，触发分析操作。前端调用后端 API 进行日

记分析，并接收 AI 建议显示在页面上。

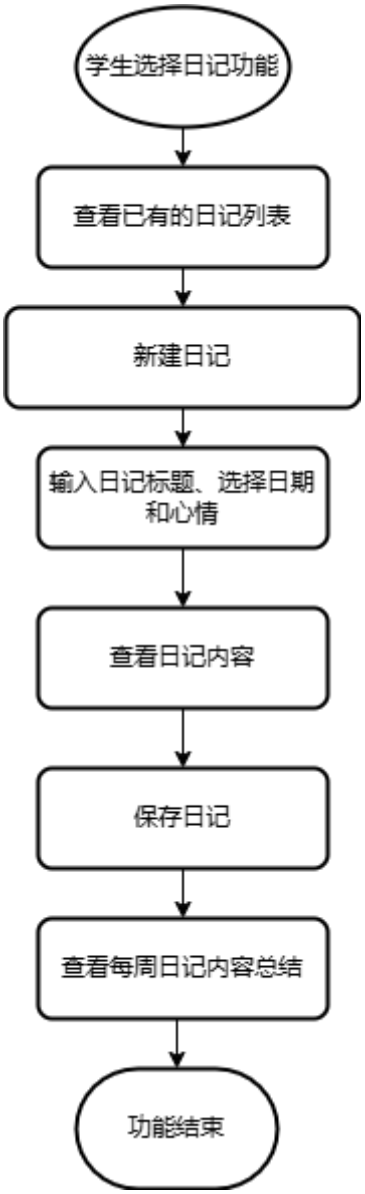

图 4.日记管理流程图

后端处理流程：

获取和保存日记信息:

后端接收前端发送的 GET 请求，返回当前用户的日记信息。后端接收前端发送的 POST 请求，保存新的日记信息到数据库中。

分析日记信息:

后端接收前端发送的 POST 请求，获取当前用户的所有日记信息。组织数据构造查询字符串，并调用 Spark API 进行日记分析。然后返回分析结果给前端。

（3）、错误提示

前端错误提示：

1、获取日记列表失败：如果获取日记列表失败，控制台会输出错误信息 “获取日记失败”。

2、输入日记标题为空：如果日记标题为空，页面会显示错误提示 “请输入日记标题”。

3、选择日期为空：如果日期为空，页面会显示错误提示 “请选择日期”。

4、选择心情为空：如果心情为空，页面会显示错误提示 “请选择心情”。

5、保存日记失败：如果保存日记失败，控制台会输出错误信息 “保存失败，请检查输入”。

6、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

7、分析失败：如果分析操作失败，控制台会输出错误信息 “分析失败”。

后端错误提示：

1、缺少用户 ID：如果请求中缺少用户 ID，返回错误响应 “缺少用户 ID”。

2、数据库保存失败：如果保存日记信息失败，返回错误响应及错误信息。

3、分析失败：如果调用 Spark API 进行分析失败，返回服务器内部错误响应。

3.1.4 文本翻译

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，提供文本翻译功能。通过 useState 管理文本输入和翻译后的文本状态。使用 fetch 方法调用后端 API 进行文本翻译。在用户输入文本时，设置一个 1 秒的延迟，通过 useRef 实现防抖机制，减少 API 调用次数。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的文本翻译请求。接收到请求后，调用讯飞翻译 API 进行翻译，并将翻译结果返回给前端。

（2）、使用流程

前端使用流程：

输入文本：

用户在文本输入框中输入需要翻译的文本。每次输入变化时，通过 handleTextChange 方法更新文本状态。如果用户停止输入超过 1 秒，将调用 handleTranslate 方法进行文本翻译。

调用翻译 API：、

在 handleTranslate 方法中，使用 fetch 方法发送 POST 请求到后端 API。请求体中包含需要翻译的文本。

接收翻译结果：

后端 API 返回翻译结果后，更新翻译后的文本状态 translatedText。翻译结果显示在只读的文本区域中。

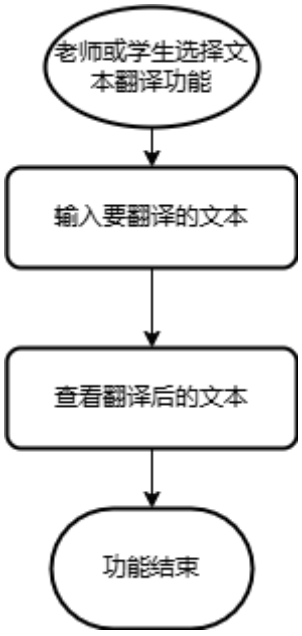

图 5.文本翻译流程图

后端使用流程：

接收翻译请求：

后端接收前端发送的 POST 请求，获取请求体中的文本。

调用讯飞翻译 API：

使用讯飞翻译 API 进行文本翻译，设置翻译的语言对（从英文到中文）。

返回翻译结果：

获取翻译结果后，返回给前端，包含翻译后的文本。

（3）、错误提示

前端错误提示：

1、翻译失败：如果翻译失败，控制台会输出错误信息 “翻译失败”。

2、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、缺少文本：如果请求体中缺少文本，返回错误响应 “缺少文本”。

2、翻译失败：如果调用讯飞翻译 API 失败，返回服务器内部错误响应。

3.1.5 师生论坛

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，提供师生论坛功能。通过 useState 管理问题列表、当前问题、用户输入等状态。用户可以浏览问题、创建新问题、查看问题详情并添加跟进回复。不同用户类型（学生和教师）可以显示不同的图标。使用 fetch 方法调用后端 API，进行数据的获取和保存。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的论坛相关请求。

实现三个主要的 API 接口：

获取和创建问题（QuestionListView）

获取问题详情（QuestionDetailView）

创建跟进回复（FollowUpView）

后端通过数据库存储和读取论坛的相关信息，并进行权限控制，确保只有特定类型的用户可以进行相关操作。

（2）、使用流程

前端使用流程：

获取问题列表：用户打开页面时，会自动调用 fetchQuestions 方法，从服务器获取问题列表，并显示在页面上。

创建新问题：用户输入问题内容，点击“新建问题”按钮，提交新问题。

如果问题内容为空，页面会显示错误提示信息。

查看问题详情：用户点击某个问题，打开问题详情模态框，显示问题的详细信息和所有跟进回复。

添加跟进回复：用户在问题详情模态框中输入跟进回复内容，并点击“发送”按钮。跟进回复会显示在问题详情模态框中。

分页浏览问题：用户可以通过分页控件，浏览不同页的问题列表。

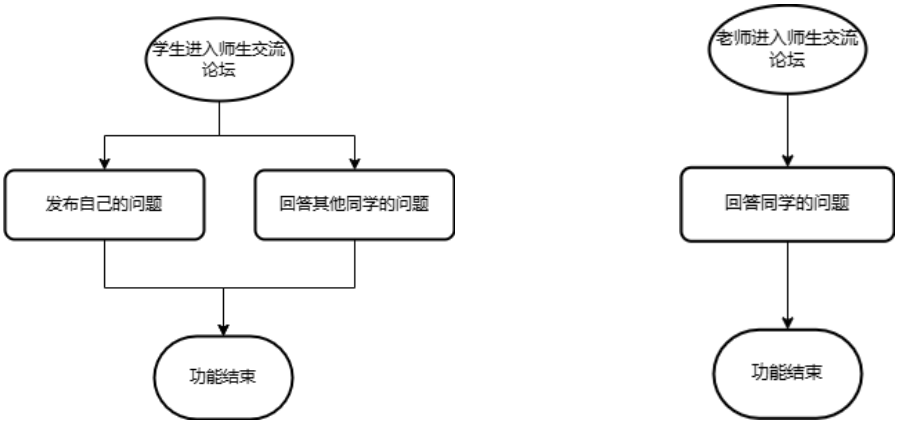

图 6. 学生和老师师生论坛流程图

后端处理流程：

获取问题列表：后端接收前端发送的 GET 请求，返回所有问题的列表。

创建新问题后端接收前端发送的 POST 请求，创建新问题。只有学生用户可以创建问题，教师和其他用户类型无权限创建问题。

获取问题详情：后端接收前端发送的 GET 请求，根据问题 ID 返回问题的详细信息。

创建跟进回复：后端接收前端发送的 POST 请求，创建新的跟进回复。只有教师和学生用户可以创建跟进回复。

（3）、错误提示

前端错误提示：

1、获取问题列表失败：如果获取问题列表失败，控制台会输出错误信息 “获取问题列表失败”。

2、输入问题内容为空：如果问题内容为空，页面会显示错误提示 “问题内容不能为空”。

3、创建问题失败：如果创建问题失败，控制台会输出错误信息 “创建问题失败”。

4、获取问题详情失败：如果获取问题详情失败，控制台会输出错误信息 “获取问题详情失败”。

5、创建跟进回复失败：如果创建跟进回复失败，控制台会输出错误信息 “创建跟进回复失败”。

6、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、缺少问题内容：如果请求体中缺少问题内容，返回错误响应 “问题内容不能为空”。

2、无权限创建问题：如果非学生用户尝试创建问题，返回错误响应 “只有学生可以提出问题”。

3、问题不存在：如果请求的问题 ID 无效，返回错误响应 “问题不存在”。

4、无效的用户类型：如果非教师或学生用户尝试创建跟进回复，返回错误响应“无效的用户类型”。

5、内容不能为空：如果请求体中缺少跟进回复内容，返回错误响应 “内容不能为空”。

3.1.6 作文润色

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，提供英语作文润色功能。通过 useState 管理文本输入、润色结果和加载状态。用户可以在文本框中输入需要润色的作文，并点击按钮获取润色建议。使用 fetch 方法调用后端 API，提交作文文本并获取润色建议。

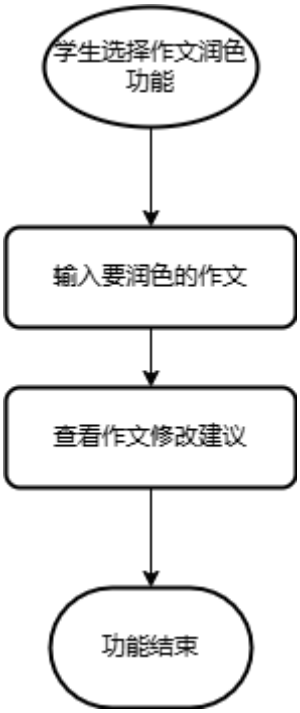

图 7. 作文润色流程图

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的作文润色请求。

实现一个主要的 API 接口：

作文润色（EssayCorrectionView）

后端调用 AI 服务进行作文润色，并返回润色建议给前端。

（2）、使用流程

前端使用流程：

输入作文文本：用户在文本输入框中输入需要润色的英语作文。文本框使用 TextArea 组件，允许用户输入最多 1000 字的文本。

提交润色请求：用户点击“立即润色”按钮，触发 handleCorrection 方法。

该方法使用 fetch 方法发送 POST 请求到后端 API，提交作文文本。

获取润色建议：后端 API 返回润色建议后，更新润色结果状态 translatedText。

润色建议显示在只读的文本区域中。

后端处理流程：

接收润色请求：后端接收前端发送的 POST 请求，获取请求体中的作文文本。

调用 AI 服务：使用 AI 服务进行作文润色，生成润色建议。AI 服务通过设置适当的参数，模拟英语老师的角色，提供详细的修改建议。

返回润色建议：获取润色建议后，返回给前端，包含润色后的建议内容。

（3）、错误提示

前端错误提示：

1、润色失败：如果润色失败，控制台会输出错误信息 “批改失败”。

2、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、作文文本为空：如果请求体中缺少作文文本，返回错误响应 “作文文本不能为空”。

2、超出请求限制：如果超出 AI 服务的请求限制，返回错误响应 “请求过多，请稍后再试”。

3、其他错误：如果发生其他错误，返回服务器内部错误响应。

3.1.7 教案生成

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，提供教案生成和下载功能。通过 useState 管理教案主题、生成的教案模板、加载状态等。用户可以输入教案主题，点击按钮生成教案模板，或下载生成的教案文档。使用 fetch 方法调用后端 API，提交教案主题并获取生成的教案模板或下载教案文档。

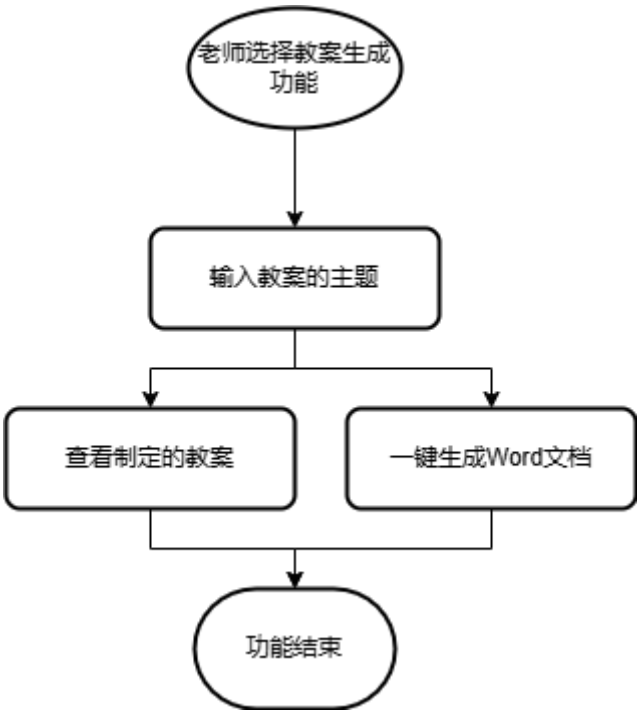

图 8. 教案生成流程图

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的教案生成请求。实现两个主要的 API 接口：

教案生成（TeachingPlanView）

教案文档生成和下载（GenerateLessonPlanView）

后端调用 AI 服务生成教案模板，并将生成的教案保存为 Word 文档，提供下载。

（2）、使用流程

前端使用流程：

输入教案主题：用户在输入框中输入教案的主题。输入框使用 TextInput 组件，允许用户输入教案主题。

生成教案模板：用户点击“生成教案模板”按钮，触发 handleGenerateLessonPlan 方法。该方法使用 fetch 方法发送 POST 请求到后端 API，提交教案主题并获取生成的教案模板。获取到的教案模板显示在只读的文本区域中。

下载教案文档：用户点击“下载教案 docx”按钮，触发 handleDownloadLessonPlan 方法。该方法使用 fetch 方法发送 POST 请求到后端 API，生成并下载教案文档。

生成的教案文档保存为 Word 文件并自动下载到本地。

后端处理流程：

接收教案生成请求：后端接收前端发送的 POST 请求，获取请求体中的教案主题。使用 AI 服务生成教案模板，返回给前端。

生成教案文档：后端接收前端发送的 POST 请求，获取请求体中的教案主题。使用 AI 服务生成教案模板，并将生成的教案保存为 Word 文档。返回生成的 Word 文档给前端，提供下载。

（3）、错误提示

前端错误提示：

1、教案生成失败：如果生成教案模板失败，控制台会输出错误信息 “生成教案模板失败”。

2、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、教案提示词为空：如果请求体中缺少教案提示词，返回错误响应 “教案提示词不能为空”。

2、教案生成失败：如果 AI 服务生成教案失败，返回服务器内部错误响应。

3、其他错误：如果发生其他错误，返回相应的错误响应。

3.1.8 作文批改

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，提供英语作文批改功能。通过 useState 管理作文标题、作文内容、批改结果和加载状态。用户可以在文本框中输入作文标题和内容，并点击按钮获取批改结果。使用 fetch 方法调用后端 API，提交作文标题和内容，并获取批改结果。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理前端发送的作文批改请求。

实现一个主要的 API 接口：

作文批改（WritingCorrectView）

后端调用 AI 服务进行作文批改，并返回批改结果给前端。

（2）、使用流程

前端使用流程：

输入作文标题和内容：用户在文本输入框中输入作文标题和需要批改的作文内容。标题和内容的输入框使用 TextArea 组件，分别允许用户输入最多 100 字和 1000 字的文本。

提交批改请求：用户点击“立即批改”按钮，触发 handleCorrection 方法。该方法使用 fetch 方法发送 POST 请求到后端 API，提交作文标题和内容并获取批改结果。获取到的批改结果显示在只读的文本区域中。

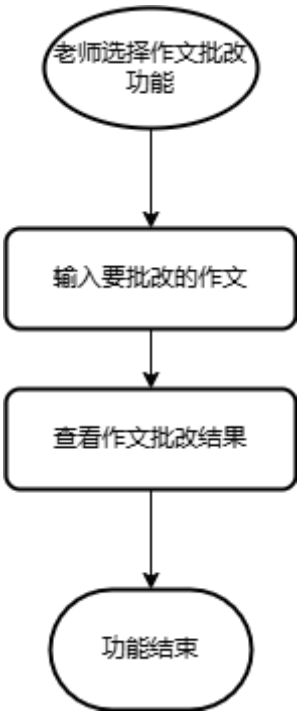

图 9. 作文批改流程图

后端处理流程：

接收批改请求：后端接收前端发送的 POST 请求，获取请求体中的作文标题和内容。

调用 AI 服务：使用 AI 服务进行作文批改，生成批改结果。AI 服务通过设置适当的参数，模拟英语老师的角色，提供详细的批改建议。

返回批改结果：获取批改结果后，返回给前端，包含批改后的建议内容。

（3）、错误提示

前端错误提示：

1、批改失败：如果批改失败，控制台会输出错误信息 “批改失败”。

2、请求出错：在任何请求出错的情况下，控制台会输出 “请求出错” 及错误信息。

后端错误提示：

1、作文标题或内容为空：如果请求体中缺少作文标题或内容，返回错误响应 “作文标题和内容不能为空”。

2、批改失败：如果 AI 服务批改作文失败，返回服务器内部错误响应。

3、其他错误：如果发生其他错误，返回相应的错误响应。

3.1.9 用户登录

（1）、功能设计

前端设计：

本前端应用使用 React 和 Carbon 组件库开发，提供用户登录、登出和游客登录功能。通过 useState 管理用户名、密码、记住我选项、错误消息等状态。使用 axios 进行 API 请求，处理登录、登出和游客登录。提供输入框供用户输入用户名和密码，并使用按钮提交登录请求。实现记住我功能，利用 localStorage 保存用户信息。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理用户登录和登出请求。

实现三个主要的 API 接口：

用户登录（LoginUserView）

用户登出（LogoutUserView）

显示登录状态（ShowLoginStatus）

（2）、使用流程

前端使用流程：

输入用户名和密码：用户在用户名和密码输入框中输入对应的信息。输入框使用 TextInput 组件，分别用于输入用户名和密码。

提交登录请求：用户点击“继续”按钮，触发 onLogin 方法。该方法使用 axios 发送 POST 请求到后端 API，提交用户名和密码并获取登录结果。如果登录成功，保存用户信息并跳转到主页；如果登录失败，显示错误消息。

记住我功能：用户可以勾选“记住我”选项，触发 handleRememberMeChange 方法。该方法使用 localStorage 保存或移除用户名和密码信息。

用户登出：用户点击“退出登录”按钮，触发 onLogout 方法。该方法使用 axios 发送 GET 请求到后端 API，处理用户登出。

游客登录：用户点击“游客登录”按钮，触发 tourist 方法。该方法检查用户是否已登录，未登录则直接跳转到主页。

后端处理流程：

接收登录请求：后端接收前端发送的 POST 请求，获取请求体中的用户名和密码。使用 Django 的 authenticate 方法验证用户名和密码。如果验证成功，使用 login 方法登录用户，并返回用户信息；如果验证失败，返回错误信息。

接收登出请求：后端接收前端发送的 GET 请求，检查用户是否已登录。如果用户已登录，使用 logout 方法登出用户，并返回成功信息。

显示登录状态：后端接收前端发送的 GET 请求，检查用户是否已登录。如果用户已登录，返回当前用户信息；如果未登录，返回未登录信息。

（3）、错误提示

前端错误提示：

1、用户名或密码为空：如果用户名或密码为空，前端将显示错误消息 “用户名或密码为空”。

2、登录失败：如果登录失败，前端将显示错误消息 “用户名或密码错误”。

3、请求出错：在任何请求出错的情况下，前端将显示错误消息 “请求失败，请稍后再试”。

后端错误提示：

1、用户名或密码为空：如果请求体中缺少用户名或密码，返回错误响应 “用户名为空” 或 “密码为空”。

2、登录失败：如果用户名或密码验证失败，返回错误响应 “登录失败”。

3、未登录：如果用户未登录，返回错误响应 “未登录”。

3.1.10 用户注册

（1）、功能设计

前端设计：

使用 React 和 Carbon 组件库开发用户注册页面。通过 useState 管理用户名、密码、ID、用户类型和错误消息等状态。提供输入框供用户输入用户名、密码和 ID，并使用单选按钮选择用户类型（学生或老师）。使用 axios 进行 API 请求，处理用户注册请求。提供按钮提交注册请求，注册成功后自动跳转到登录页面。

后端设计：

后端使用 Django REST framework 提供 API 接口，处理用户注册请求。

实现用户注册的 API 接口 CreateUserView：检查请求体中的用户名、密码和用户类型。验证用户名是否已存在，用户类型是否正确。

根据用户类型分别创建学生或老师用户。使用事务确保用户和学生/老师记录的原子性创建。

（2）、使用流程

前端使用流程：

输入注册信息：用户在用户名、密码和 ID 输入框中输入对应的信息。

输入框使用 TextInput 组件。

选择用户类型：用户通过单选按钮选择用户类型（学生或老师）。单选按钮使用 RadioButton 组件。

提交注册请求：用户点击“立即注册”按钮，触发 handleRegister 方法。

该方法使用 axios 发送 POST 请求到后端 API，提交用户名、密码、用户类型和 ID。如果注册成功，显示成功消息并跳转到登录页面；如果注册失败，显示错误消息。

后端处理流程：

接收注册请求：后端接收前端发送的 POST 请求，获取请求体中的用户名、密码、用户类型和 ID。验证请求数据的完整性和合法性。检查用户名是否已存在，用户类型是否正确。

创建用户：根据用户类型分别创建学生或老师用户。使用事务确保用户和学生/老师记录的原子性创建。返回注册成功或失败的响应。

（3）、错误提示

前端错误提示：

1、必填项为空：如果用户名、密码、ID 或用户类型为空，前端将显示错误消息 “请填写所有必填项”。

2、请求失败：在任何请求出错的情况下，前端将显示错误消息 “请求失败，请稍后再试”。

3、注册失败：如果后端返回注册失败，前端将显示错误消息，内容为后端返回的详细信息。

后端错误提示：

1、用户名或密码为空：如果请求体中缺少用户名或密码，返回错误响应 “用户名为空” 或 “密码为空”。

2、用户名已存在：如果用户名已被使用，返回错误响应 “用户名已被使用”。

3、用户类型错误：如果用户类型不正确，返回错误响应 “用户类型错误”。

4、内部错误：在任何其他错误情况下，返回错误响应 “内部错误”。

4、用户界面设计

4.1 平台主页实现

4.1.1 概述

4.1.1.1 用户界面目标

平台主页设计旨在为用户提供直观和友好的操作体验，便于用户快速了解和访问平台的主要功能和服务。

4.1.1.2 用户群体

主要用户群体包括学生和老师。

学生用户通过平台获取学习资源和智能辅助工具。

老师用户通过平台获取教学辅助工具和管理学生学习情况。

4.1.2 界面布局

4.1.2.1 总体布局

平台主页采用一页式滚动布局，包含导航栏、头部介绍、产品特色、服务介绍、演示图片等部分。

通过 SmoothScroll 实现平滑滚动效果，提升用户体验。

4.1.2.2 主要页面设计

1、导航栏

固定在页面顶部，包含平台各个部分的链接，便于用户快速导航。

点击导航链接可平滑滚动至对应部分。

2、头部介绍

包含平台标题和简介，简明扼要地介绍平台的核心价值。

提供“进入平台”按钮，引导用户登录或注册。

```
export const Header = (props) => {
  return (
    <header id="header">
      <div className="intro">
        <div className="overlay">
          <div className="container">
            <div className="row">
              <div className="col-md-8 col-md-offset-2 intro-text">
                <h1>{props.data ? props.data.title : "Loading"}<span></span></h1>
                <p>{props.data ? props.data.paragraph : "Loading"}</p>
                <a href="/login" className="btn btn-custom btn-lg page-scroll">进入平台</a>
              </div>
            </div>
          </div>
        </div>
```

```
        </div>
      </header>
    );
};
```

3、产品特色

以网格布局展示平台的主要特色功能，每个功能包括图标、标题和简要描述。

图标和标题使用不同颜色区分，增强视觉层次感。

```
export const Features = (props) => {
  return (
    <div id="features" className="text-center">
      <div className="container">
        <div className="col-md-10 col-md-offset-1 section-title">
          <h2>产品特色</h2>
        </div>
        <div className="row">
          {props.data ? props.data.map((d, i) => (
            <div key={`${d.title}-${i}`} className="col-xs-6 col-md-3">
              <i className={d.icon}></i>
              <h3>{d.title}</h3>
              <p>{d.text}</p>
            </div>
          )) : "Loading..."}
        </div>
      </div>
    </div>
  );
};
```

4、服务介绍

详细介绍平台提供的主要服务，包括成绩分析、作文修改等。

每个服务模块包含图标、标题和描述，布局为三列设计。

```
    export const Services = (props) => {
  return (
    <div id="services" className="text-center">
      <div className="container">
        <div className="section-title">
          <h2>服务介绍</h2>
          <p>我们的平台结合讯飞星火大模型，为学生用户提供全面的、智能的学习解决方案，涵盖从成绩分析到作文修改的多项功能，助力学生高效学习。为老师用户提供智能的教学助手工具，涵盖从教学计划方案生成到作文批改等多项功能，帮助老师减轻教学负担。</p>
        </div>
```

```
          <div className="row">
            {props.data ? props.data.map((d, i) => (
              <div key={`${d.name}-${i}`} className="col-md-4">
                <i className={d.icon}></i>
                <div className="service-desc">
                  <h3>{d.name}</h3>
                  <p>{d.text}</p>
                </div>
              </div>
            )) : "loading"}
          </div>
        </div>
      </div>
    );
};
```

5、演示图片

通过图片展示平台的界面和功能，增强用户的直观理解。

图片按网格布局排列，提供大图和小图切换展示功能。

```
      export const Gallery = (props) => {
  return (
    <div id="portfolio" className="text-center">
      <div className="container">
        <div className="section-title">
          <h2>演示图片</h2>
          <p></p>
        </div>
        <div className="row">
          <div className="portfolio-items">
            {props.data ? props.data.map((d, i) => (
              <div key={`${d.title}-${i}`} className="col-sm-6 col-md-4 col-lg-4">
                <Image title={d.title} largeImage={d.largeImage} smallImage={d.smallImage}
/>
                </div>
              )) : "Loading..."}
          </div>
        </div>
      </div>
    );
};
```

4.1.2.3 交互设计

点击导航栏的链接，页面平滑滚动到对应的部分，用户可以流畅地浏览不同的内容。

头部介绍部分的“进入平台”按钮，引导用户登录或注册，便于用户快速开始使用平台。

服务介绍部分的每个服务模块点击后可以展开更多详细信息（如果实现）。

4.1.3 视觉设计

4.1.3.1 色彩方案

主色调为蓝色，辅以白色和灰色，体现科技感和专业性。

不同模块使用不同色块区分，提高视觉层次感。

4.1.3.2 字体

使用现代简洁的无衬线字体，如 Arial 或 Helvetica。

不同层级的标题和正文字体大小和样式有所区分，增强层次感。

4.1.3.3 图表和图像

产品特色和服务介绍部分使用简洁明了的图标，增强信息传达。

演示图片部分使用高质量的图片，展示平台的界面和功能。

4.1.4 用户体验设计

1、导航设计

固定导航栏，用户可以随时访问平台的不同部分。

使用 SmoothScroll 实现平滑滚动，提升用户浏览体验。

2、可以性

平台主页布局简洁，信息展示清晰，用户可以快速找到需要的内容。

各模块之间的过渡平滑自然，用户体验流畅。

3、可访问性

界面元素间的对比度足够，便于视力不佳的用户阅读。

图片和图标提供替代文本，便于使用屏幕阅读器的用户理解内容。

4.1.5 界面设计规范

1、设计指南

平台主页设计应遵循简洁、直观的原则，尽量减少用户的学习成本。

设计过程中应注意保持视觉一致性，提升用户的整体体验。

2、组件库

平台主页设计中使用了 Bootstrap 组件库，确保界面的一致性和响应式设计。

使用 Font Awesome 提供的图标，增强界面的视觉效果。

4.1.6 效果展示


localhost:3000
localhost:3000
智学未来教育平台
智学未来教育平台
AI驱动的一站式教育学习平台
进入平台
产品特色
AI赋能教育
借助星火大模型，实现教育的AI赋能，使学习更高效、更有趣。
智能成绩分析
利用讯飞星火大模型，实时分析学生的学习成绩，提供详尽的分析报告，帮助学生发现不足并给出学习建议规划。
智能作文批改与修改
AI自动批改作文，帮助学生提升写作能力和帮助老师快速批改作文。
多功能交流平台
提供师生论坛和文本翻译功能，方便师生交流学习经验，促进互动与合作。
服务介绍
我们的平台结合讯飞星火大模型，为学生用户提供全面的、智能的学习解决方案，涵盖从成绩分析到作文修改的多项功能，助力学生高效学习。
为老师用户提供智能的教学助手工具，涵盖从教学计划方案生成到作文批改等多项功能，帮助老师减轻教学负担。
AI成绩分析
精准分析学习成绩，生成详细报告，帮助学生了解自身学习情况。
个性化教学计划编写
根据教师需求，定制教学计划，确保高效、有针对性的教学。
日记本
提供日记本功能，鼓励学生记录学习过程和心得，培养良好的学习习惯。

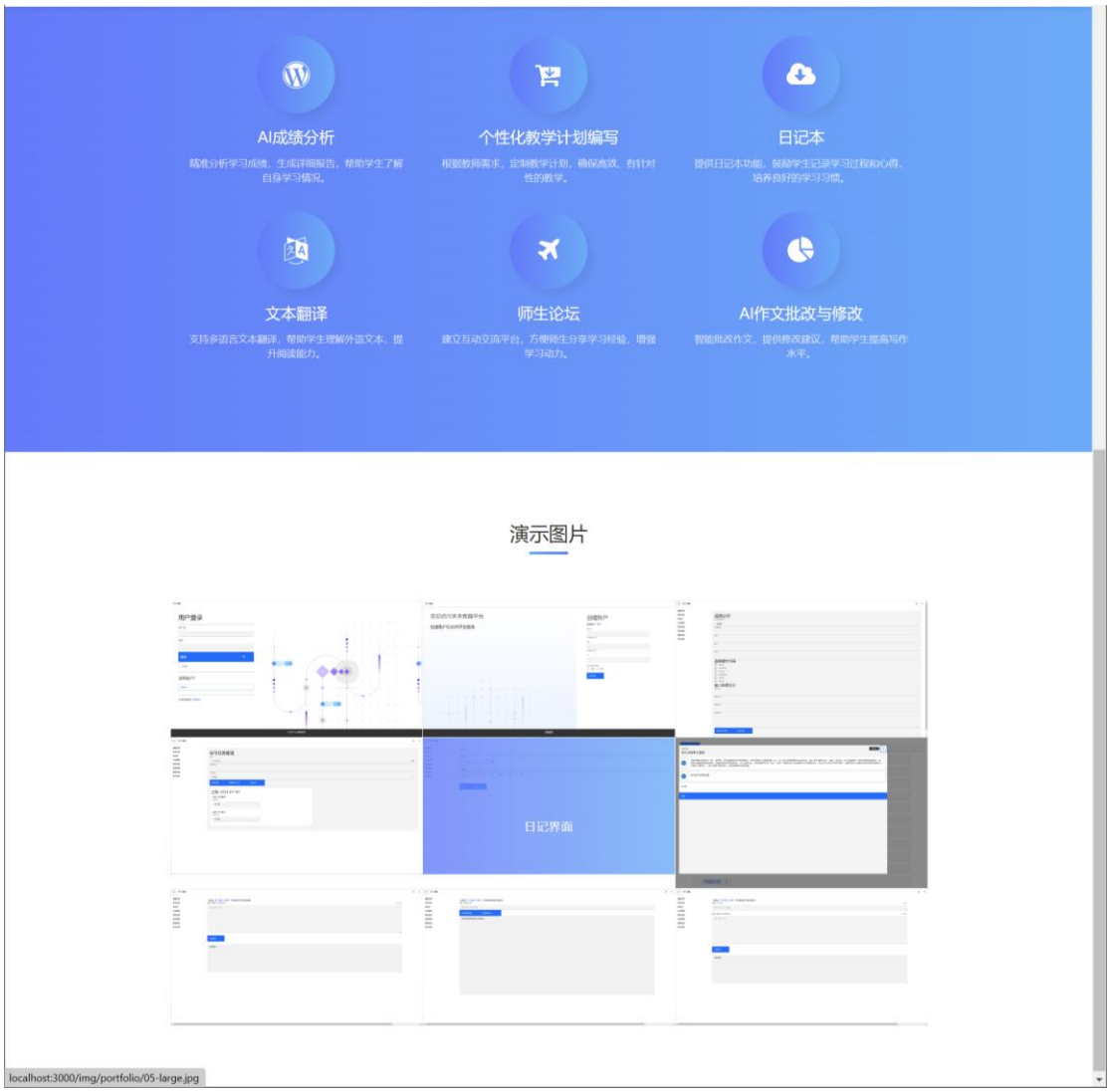

图 10.主页效果展示

4.2 功能界面实现

4.2.1.1 概述

4.2.1.1.1、用户界面目标

功能界面的设计目标是提供一个便捷、直观的操作平台，帮助学生和教师用户高效地管理和使用各种学习辅助工具，学生功能界面展示如图 4 所示，老师功能界面展示如图 5 所示。

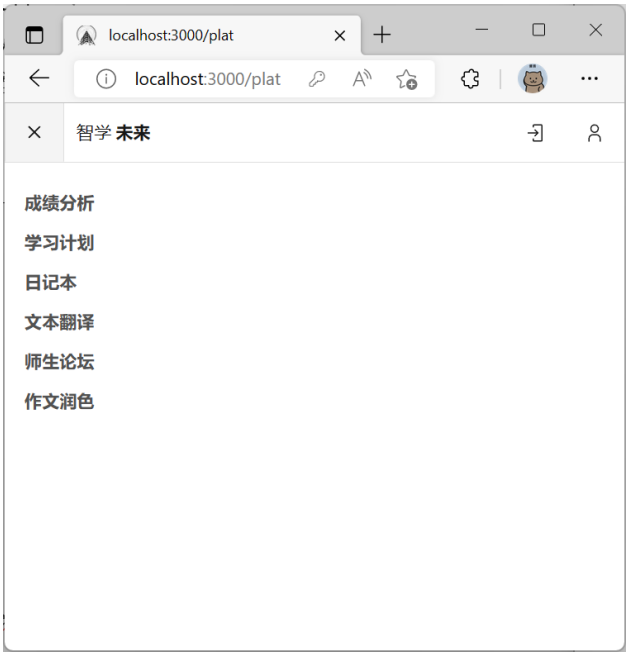

图 11.学生功能界面

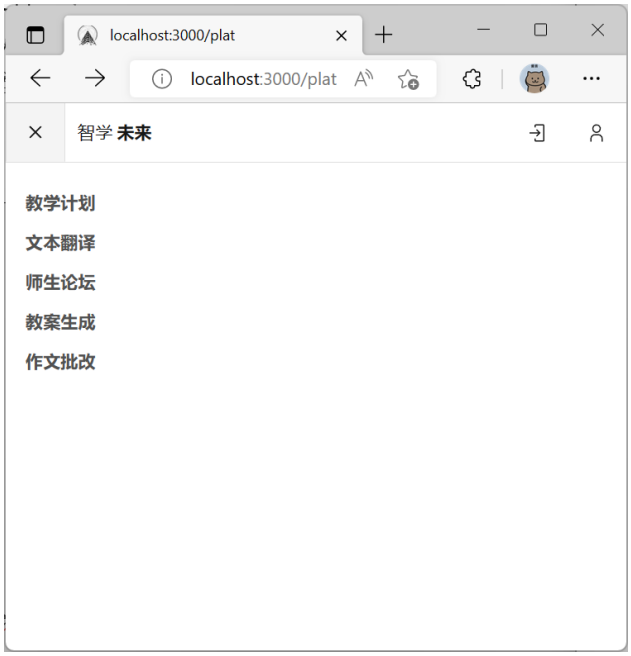

图 12.老师功能界面

4.2.1.2、界面布局

4.2.1.2.1 总体布局

界面由头部导航栏、侧边导航栏和内容区三部分组成。

头部导航栏包含平台名称和全局操作按钮。

侧边导航栏包含学生用户可访问的主要功能链接。

内容区根据选择的功能动态展示对应的内容。

4.2.1.2.2 主页页面设计

1、头部导航栏

平台的头部导航栏固定在页面顶部，包含平台名称和全局操作按钮。

平台名称为“智学未来”，点击可返回主页。

全局操作按钮包括注册和登录按钮，使用 Carbon Design System 提供的图标。

```
<Header>
  <HeaderMenuButton aria-label="打开侧边菜单" isCollapsible onClick={toggleSideNav}
isActive={isSideNavExpanded} />
  <HeaderName href="/" prefix="智学">未来</HeaderName>
  <HeaderGlobalBar>
    <Button renderIcon={Login} onClick={ ()=>navigate('/register')} size="small" kind="ghost"
hasIconOnly />
    <Button renderIcon={User} onClick={ ()=>navigate('/login')} size="small" kind="ghost"
hasIconOnly />
  </HeaderGlobalBar>
</Header>
```

2、侧边导航栏

侧边导航栏固定在页面左侧，提供学生用户可访问的主要功能链接。

根据用户身份（学生或教师）动态显示不同的功能菜单。

主要功能包括成绩分析、学习计划、日记本、文本翻译、师生论坛、作文润色等。

```
<SideNav isFixedNav={true} expanded={isSideNavExpanded} isChildOfHeader={false} aria-
label="Side navigation">
  <SideNavItems>
    {student_id && (<SideNavLink href='/plat/analyse'>成绩分析</SideNavLink>)}
    {student_id && (<SideNavLink href='/plat/to_do_list'>学习计划</SideNavLink>)}
    {student_id && (<SideNavLink href='/plat/diary'>日记本</SideNavLink>)}
    {teacher_id && (<SideNavLink href='/plat/to_do_list'>教学计划</SideNavLink>)}
    <SideNavLink href='/plat/text'>文本翻译</SideNavLink>
    <SideNavLink href='/plat/bbs'>师生论坛</SideNavLink>
    {student_id && (<SideNavLink href='/plat/composition'>作文润色</SideNavLink>)}
    {teacher_id && (<SideNavLink href='/plat/teaching_plan'>教案生成</SideNavLink>)}
    {teacher_id && (<SideNavLink href='/plat/correct'>作文批改</SideNavLink>)}
  </SideNavItems>
</SideNav>
```

3、内容区

内容区根据用户选择的功能动态展示对应的内容。

通过 <Outlet /> 组件实现内容的动态渲染。

```
<UserProvider>
  <Outlet />
</UserProvider>
```

4.2.1.3 视觉设计

1、导航栏设计

固定导航栏和侧边栏，用户可以随时访问平台的不同部分。

侧边导航栏根据用户身份动态显示不同的功能菜单，提升个性化体验。

2、可用性

界面布局简洁，信息展示清晰，用户可以快速找到需要的内容。

各模块之间的过渡平滑自然，用户体验流畅。

3、可访问性

界面元素间的对比度足够，便于视力不佳的用户阅读。

图片和图标提供替代文本，便于使用屏幕阅读器的用户理解内容。

4.2.1.5 界面设计规范

1、设计指南

学生功能界面设计应遵循简洁、直观的原则，尽量减少用户的学习成本。

设计过程中应注意保持视觉一致性，提升用户的整体体验。

2、组件库

使用 Carbon Design System 提供的组件，确保界面的一致性和响应式设计。

使用 Font Awesome 提供的图标，增强界面的视觉效果。