辽宁工程技术大学软件学院

课程设计报告

基于多设计模式融合的智能决策系统

Intelligent Decision System Based on Multiple Design Patterns

班　　级： 软件工程23-15班

组　　长： 王佳兴

组　　员： 潘嘉鑫

邢纹源

2025 年 12 月 14 日

目录

1 系统概述

本系统是一个基于多设计模式融合的智能决策平台，采用前后端分离的现代化Web应用架构进行设计与实现。系统在前端层面使用Vue3框架构建用户交互界面，在后端层面采用Flask框架提供RESTful API服务，通过将策略模式、工厂模式、模板方法模式、观察者模式以及单例模式这五种经典设计模式进行深度融合，实现了一个功能完整、架构清晰、易于扩展的智能系统平台。系统的核心目标是展示设计模式在实际软件工程项目中的综合应用，通过具体的AI决策场景来验证各种设计模式的实用价值和协同效果。

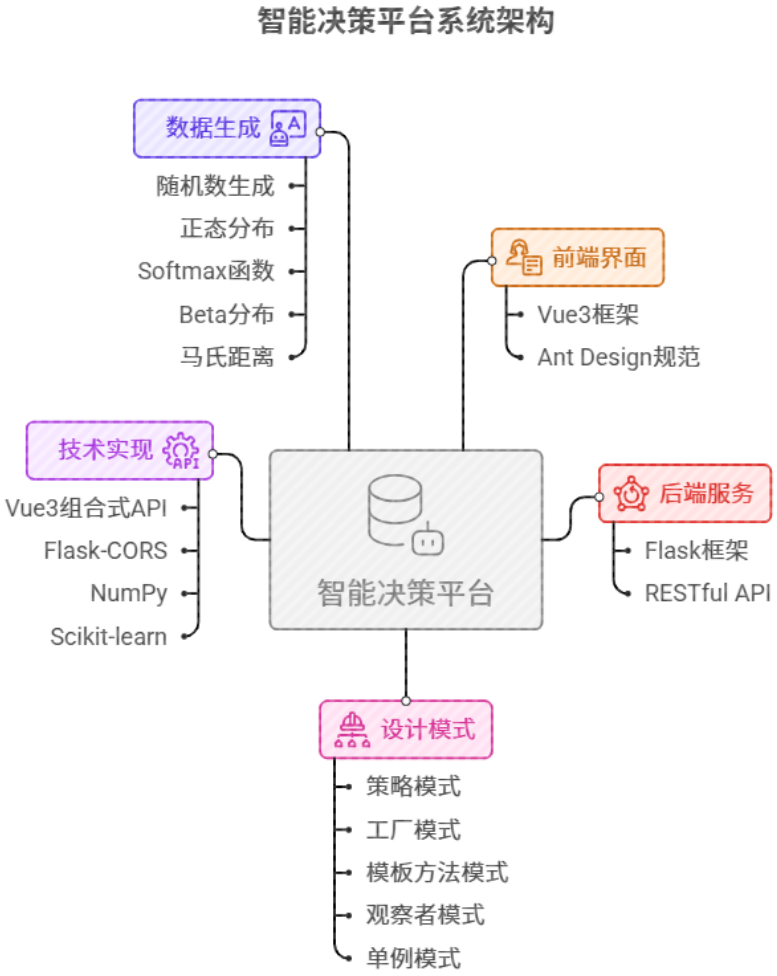

图 1: 本系统架构总览

在系统的整体设计思路上，我们遵循了软件工程中的关注点分离原则和单一职责原则。前端界面采用了参考Ant Design设计规范的现代化清爽风格，使用白色作为主背景色，蓝色作为品牌主色调，通过卡片化的布局方式和精致的交互效果为用户

提供流畅的操作体验。后端服务则专注于业务逻辑的处理和数据管理，通过设计模式的合理应用确保了代码的可维护性和可扩展性。系统提供了从任务选择、策略配置、模型执行到结果分析的完整业务流程，涵盖了数据预测、图像分类、智能推荐和异常检测四种典型的AI应用场景，每种场景都能够支持四种不同复杂度的算法策略供用户灵活选择。

从技术实现角度来看，系统采用了成熟稳定的技术栈组合。前端使用Vue3的组合式API编写组件逻辑，通过Pinia进行全局状态管理，使用Vue Router实现页面路由导航，借助Chart.js库实现数据的可视化展示，整个前端应用由Vite构建工具进行打包优化。后端基于Flask3.0框架搭建，使用Flask-CORS处理跨域请求，通过NumPy进行科学计算，利用Scikit-learn提供机器学习算法支持。系统实现了完整的十七个RESTful API接口，涵盖了配置管理、任务执行、结果查询、历史记录、系统日志、统计分析、数据导出等核心功能模块，形成了一个功能完备的企业级应用系统框架。

在设计模式的应用方面，系统并非简单地罗列各种模式，而是根据实际业务需求将它们有机地结合在一起。策略模式被应用于AI算法的动态选择和切换场景中，通过定义统一的策略接口和多个具体的策略实现类，使得系统能够在运行时根据用户的选择动态地改变算法行为。工厂模式则负责根据不同的任务类型创建相应的AI模型实例，将对象的创建过程与使用过程分离，降低了系统各模块之间的耦合度。模板方法模式规范了所有AI模型的执行流程，确保每个模型都按照数据加载、数据预处理、模型推理、结果输出这一标准化流程进行处理。观察者模式实现了结果数据的自动更新通知机制，当AI模型执行完成后，系统会自动通知前端界面刷新展示最新的结果数据。单例模式则用于管理系统的全局配置信息，确保整个应用程序只存在一个配置管理器实例，避免了配置数据的不一致问题。

由于时间与算力的原因，系统的数据生成机制采用了基于NumPy科学计算库的随机数生成算法，而非简单的固定数值模拟。对于每次模型执行，系统都会基于当前时间戳生成随机种子，从而产生略有差异但符合统计规律的结果数据。在策略性能指标方面，不同的算法策略具有不同的基准准确率，并在此基础上添加符合正态分布的随机波动，使得每次执行的结果都有微小变化，更加真实地模拟了实际AI模型的运行特性。在分类任务中，系统使用Softmax函数对随机生成的logits进行归一化，得到和为1的概率分布。在推荐任务中，使用Beta分布生成偏向高分的推荐得分。在异常检测任务中，通过计算样本点的马氏距离来生成异常分数，这些方法都确保了数据的科学性和合理性。

2 运行截图

系统的首页展示了整个平台的概览信息，通过四个统计卡片直观地呈现了任务总数、已完成任务数、可用策略数以及平均准确率等关键指标。这些统计数据并非静态展示，而是通过调用后端API接口实时获取，能够动态反映系统的当前运行状态。页面采用了网格布局方式，将各个信息模块合理分布，既保证了信息的完整性，又避免了视觉上的拥挤感。统计卡片采用了不同的颜色标识，蓝色代表任务相关信息，绿色表示完成状态，橙色显示策略配置，紫色展示性能指标，通过色彩的差异化帮助用户快速识别不同类型的信息。

在当前系统配置展示区域，用户可以清楚地看到当前选中的任务类型和算法策略，以及系统的执行次数和运行状态。这个模块采用了卡片式设计，使用浅灰色背景与白色主题形成对比，重要信息使用蓝色高亮显示，确保用户能够一眼看到关键参数。右侧的设计模式应用说明模块以列表形式展示了系统中应用的五种设计模式及其各自的作用，每个模式都配有相应的图标标识和简短的功能描述，帮助用户理解系统的架构设计思想。页面底部提供了三个快速导航卡片，分别指向任务中心、策略配置和系统架构页面，用户点击后即可跳转到相应的功能模块，实现了流畅的页面导航体验。

任务中心页面展示了系统支持的四种AI任务类型，每个任务以卡片形式呈现，包含了任务图标、任务名称、任务描述以及分类和难度标签。用户可以通过点击任务卡片来选择要执行的任务类型，被选中的任务卡片会显示蓝色边框高亮效果，并在底部显示选中标识。任务卡片在鼠标悬停时会产生轻微的上浮动画效果和阴影增强效果，为用户提供视觉反馈。数据预测任务被标记为时序分析类别，难度为中等级别，主要用于基于历史数据进行未来趋势的预测分析。图像分类任务属于计算机视觉领域，难度较高，需要对图像进行智能识别和分类处理。智能推荐任务归类于推荐系统范畴，难度中等，能够根据用户行为进行个性化内容推荐。异常检测任务属于异常分析类型，难度相对简单，专注于发现数据中的异常模式和离群点。

策略配置页面以网格形式展示了四种不同的AI算法策略，每种策略都详细列出了其准确度、运行速度、内存占用和算法复杂度等多维度的性能指标。基础模型采用传统机器学习算法实现，具有快速运行和低内存占用的特点，准确率约为百分之七十五，适合对实时性要求较高但精度要求不太严格的应用场景。深度学习模型使用神经网络架构，准确率提升至百分之八十八，但相应地运行速度降为中等水平，内存占用也增加到中等级别，算法复杂度属于中等程度。Attention增强模型结合了

注意力机制技术，将准确率进一步提升至百分之九十二，但这也带来了较高的内存占用和算法复杂度，运行速度保持在中等水平。集成学习模型通过融合多个子模型的预测结果，达到了最高的百分之九十五的准确率，但代价是运行速度较慢，内存占用和算法复杂度都达到了高等级。

页面下方的策略对比分析图表以柱状图的形式直观展示了四种策略的准确率差异，被选中的策略对应的柱状条会显示为蓝色，而未选中的策略则显示为灰色，这种视觉对比帮助用户快速理解不同策略之间的性能差异。每个柱状条的高度与策略的准确率成正比，底部标注了策略名称，顶部显示了具体的准确率百分比数值。用户选择完策略后，页面底部会出现一个蓝色的主操作按钮，引导用户继续进入模型执行阶段。整个页面的设计遵循了渐进式披露的原则，只在用户完成当前步骤后才显示下一步的操作入口，避免了界面的复杂性。

AI运行控制台页面提供了模型执行的控制界面，顶部展示了当前配置的任务类型和算法策略，确保用户在执行前能够确认参数设置的正确性。页面中央是一个醒目的运行按钮，采用了渐变蓝色背景和白色文字，按钮在鼠标悬停时会产生轻微的缩放效果和阴影增强效果，提供了良好的交互反馈。当用户点击运行按钮后，系统会按照模板方法模式定义的标准流程依次执行数据加载、数据预处理、AI推理和结果生成四个步骤，每个步骤完成后会在进度列表中显示对号标识和绿色背景，让用户清楚地了解当前的执行进度。

在模型运行过程中，页面会显示一个旋转的加载动画图标和运行中的文字提示，告知用户系统正在处理请求。执行完成后，页面会展示一个绿色的成功提示框，包含对号图标、完成标题和提示信息，并提供了一个绿色的查看结果按钮，引导用户跳转到结果分析页面。整个执行过程的用户体验设计充分考虑了等待时的心理预期管理，通过进度展示和动画效果减少用户的焦虑感。系统状态监控区域实时显示了系统的初始化状态和运行状态，这些信息通过调用后端API获取，能够反映系统的真实运行情况。

结果分析页面以数据可视化为核心，通过多种图表类型展示AI模型的执行结果和性能指标。性能指标卡片以网格布局形式展示了准确度、精确度、召回率和F1分数四个关键指标，每个指标都包含了具体的数值和可视化的进度条，使用蓝色作为主色调保持了视觉的统一性。预测结果可视化图表采用了Chart.js库绘制的折线图，横轴表示时间点，纵轴表示预测值，折线下方填充了半透明的蓝色区域，增强了数据的可读性。图表支持交互操作，用户可以通过鼠标悬停查看具体数据点的数值信息。

详细数据展示区域以列表形式呈现了任务类型、使用的模型名称以及置信度等补充信息，这些信息采用了标签值的展示方式，标签使用灰色文字，值使用深色文字并加粗显示，形成了清晰的视觉层次。策略对比功能允许用户同时选择多个策略进行性能对比分析，通过勾选相应的策略复选框并点击开始对比按钮，系统会并行执行多个策略的模型推理，并将结果以柱状图的形式展示出来。对比图表中不同的策略使用不同的颜色进行区分，每组指标并排显示，便于用户直观地比较各个策略在不同性能维度上的表现差异。这种对比分析功能体现了观察者模式的应用价值，当新的结果数据产生时，图表组件会自动接收通知并更新显示内容。

系统架构页面详细阐述了整个系统的设计思想和技术实现方案，通过文字描述和架构图示相结合的方式，帮助读者全面理解系统的内部结构。页面首先介绍了系统的核心思想，强调了多设计模式融合的重要性以及各个模式在系统中扮演的角色。系统架构部分按照分层的方式展示了前端层、API层、设计模式层和AI模型层的职责划分，每一层都列出了包含的主要组件和功能模块。前端层包括六个核心页面组件，API层提供了十七个RESTful接口，设计模式层实现了五种经典模式的代码结构，AI模型层封装了四种不同任务类型的模型实现。

设计模式详解部分针对每一种设计模式都进行了深入的说明，包括模式的应用场景、具体实现方式、核心代码片段以及带来的优势。策略模式部分详细说明了如何定义AIStrategy抽象基类和四个具体的策略实现类，以及AIContext类如何在运行时动态选择和切换策略。工厂模式部分展示了AIModelFactory类如何根据任务类型字符串创建相应的模型实例，实现了对象创建逻辑的集中管理。模板方法模式部分说明了AIModel抽象类中的execute方法如何定义了算法的骨架结构，而将具体的步骤实现延迟到子类中完成。观察者模式部分介绍了ResultSubject和ResultObserver类的协作机制，以及如何通过通知机制实现数据的自动更新。单例模式部分说明了SystemConfig类如何通过静态方法和私有构造函数确保全局只存在一个配置实例。

页面流程图以可视化的方式展示了用户从进入系统到完成一次完整的AI任务执行的操作路径，包括首页浏览、任务选择、策略配置、模型运行和结果查看五个主要环节，每个环节之间通过箭头连接，形成了清晰的业务流程链。系统亮点部分总结了项目在UI设计、功能实现、架构规划、可扩展性等方面的优势特点，强调了系统不仅仅是一个简单的设计模式演示程序，而是一个功能完整、可实际应用的企业级系统原型。适用课程部分说明了该系统可以作为软件设计模式、软件工程课程设计、AI应用系统设计、前端与后端综合开发等多门课程的实践项目参考案例。

3 源代码

本系统后端主文件基于Flask框架，配置了基础API路由及设计模式的集成。实现包括：系统配置全局单例、任务/日志/数据的内存存储，以及核心API如健康检查、配置获取与设置、任务与策略信息、模型执行、结果/历史/日志/统计等查询与管理。日志条目有限制，保证内存不会无限增长。功能接口均有简单参数校验与反馈，系统支持数据导出和监控基础指标，关键统计如平均准确率等均由历史数据实时计算得出。

系统五大设计模式简述如下： - 单例模式：SystemConfig类保证全局配置唯一实例，懒加载，支持配置信息读取与设置。 - 策略模式：AIStrategy抽象基类及其四个具体实现，分别模拟基础、深度学习、注意力、集成四类策略，支持动态切换，分别有各自的指标与预测逻辑。AIContext负责管理和切换策略对象，统一调用预测和指标方法。 - 工厂+模板方法模式：AIModel为抽象基类，定义数据加载、预处理、推理和结果输出的统一流程，每种任务类型（如预测、分类、推荐、异常检测）都有对应的数据生成和处理策略。AIModelFactory按任务类型实例化具体模型，简化模型创建。 - 观察者模式：ResultSubject作为主题，管理通知与数据推送，ResultObserver作为观察者更新结果历史，支持多观察者注册与自动推送及时刷新。

前端主组件简明分为顶部导航和内容区，常用Vue3+Pinia+Chart.js技术栈。导航栏白底固定，含logo、路由菜单和用户信息，主区由router-view渲染。样式清爽，布局自适应，交互动画流畅。

主要页面结构包括： - 首页：统计卡片网格，展示任务数/完成量/策略数/准确率等指标，配置信息和设计模式介绍清晰分区，快速导航卡片直达核心功能。 - 任务中心/策略配置：网格卡片布局，v-for渲染任务和策略，选中高亮，策略性能直观对比。 - AI运行控制台：展示当前配置，控制运行、步骤进度条和运行状态，结果完成后支持跳转，进度状态实时更新。 - 结果分析：集成Chart.js动态数据可视化，支持策略多选对比。所有交互和数据获取均采用Composition API组织，代码简洁清晰，样式采用BEM和CSS变量规范。

整体设计注重结构清晰与模块解耦，核心复杂逻辑如策略动态切换、模型数据模拟、实时结果通知，均通过设计模式实现，提升了扩展性与规范性。代码细节保持精简，重点突出系统关键功能与架构特色。

4 结构图

系统的整体架构采用了经典的三层结构设计，从上到下依次是展示层、业务逻辑层和数据访问层，这种分层架构确保了各层之间的职责清晰、耦合度低、可维护性强。展示层位于架构的最顶层，由Vue3框架构建的单页应用组成，负责与用户进行交互，接收用户的输入操作并将处理结果以可视化的形式呈现给用户。展示层包含了六个核心页面组件，分别是首页Dashboard、任务中心TaskCenter、策略配置StrategyLab、AI运行控制台AIRuntime、结果分析ResultAnalysis和系统架构SystemDesign，每个页面组件都是一个独立的Vue单文件组件，封装了模板、脚本和样式三个部分。

展示层通过Vue Router进行页面路由管理,实现了不同页面之间的导航跳转功能,路由配置采用了前端路由模式,所有的页面切换都在客户端完成而不需要向服务器请求新的HTML文档,从而提供了更快的页面切换速度和更流畅的用户体验。展示层使用Pinia进行状态管理,将应用的共享状态集中存储在Store中,各个组件通过访问Store来读取状态数据或调用Actions来修改状态,这种集中式的状态管理模式避免了组件之间通过Props和Events进行复杂的数据传递,简化了组件的设计和实现。展示层还集成了Chart.js图表库用于绘制各种数据可视化图表,通过将后端返回的数据转换为图表配置对象,实现了折线图、柱状图、饼图等多种图表类型的动态渲染。

业务逻辑层位于架构的中间层,由Flask框架搭建的RESTful API服务组成,负责处理来自展示层的HTTP请求,执行相应的业务逻辑,并将处理结果返回给展示层。业务逻辑层定义了十七个API端点,涵盖了系统的所有功能模块,包括配置管理、任务执行、结果查询、历史记录、系统日志、统计分析和数据导出等。每个API端点都是一个路由处理函数,使用Flask的装饰器语法将URL路径和HTTP方法与函数进行绑定,函数内部实现了具体的业务逻辑处理。业务逻辑层通过Flask-CORS扩展处理跨域资源共享问题,允许运行在不同域名或端口上的前端应用访问后端API,这是前后端分离架构的必要配置。

业务逻辑层的核心是五种设计模式的实现代码,这些设计模式类构成了系统的设计模式层,是业务逻辑层的重要组成部分。单例模式通过SystemConfig类实现了全局配置的统一管理,确保整个应用程序只存在一个配置实例,避免了配置数据的不一致和重复创建的开销。策略模式通过AIStrategy抽象类和四个具体策略类实现了AI算法的封装和动态切换,AIContext类作为策略的使用者,在运行时根据配置选择合适的策略对象并调用其方法,这种设计使得添加新的算法策略变得非常简单,只需要创建一个新的策略类并在映射字典中注册即可。

工厂模式通过AIModelFactory类实现了AI模型对象的创建逻辑集中管理,根据任务类型参数动态决定创建哪一个具体的模型类实例,这种设计将对象的创建和使用分离,降低了代码的耦合度,也使得添加新的任务类型变得容易。模板方法模式通过AIModel抽象类的execute方法定义了AI模型执行的标准流程,将流程中的某些步骤抽象为待子类实现的方法,这种设计既保证了所有模型都遵循统一的执行流程,又允许不同的模型根据自身特点定制某些步骤的具体实现,在复用和灵活性之间取得了良好的平衡。观察者模式通过ResultSubject和ResultObserver类实现了结果数据的发布订阅机制,当AI模型执行完成产生新的结果数据时,主题对象会自动通知所有注册的观察者对象,观察者对象收到通知后更新自己维护的数据,这种设计实现了结果生成者和结果使用者之间的解耦,使得系统可以方便地添加新的结果处理逻辑而不影响现有代码。

数据访问层在本系统中以内存数据存储的形式存在,使用Python的列表和字典数据结构保存任务历史记录、系统日志和数据记录等信息,虽然这种方式在应用重启后会丢失数据,但对于演示和教学目的来说已经足够,在实际的生产环境中可以很容易地替换为数据库访问层而不影响上层的业务逻辑代码。系统使用NumPy库进行科学计算和数据处理,NumPy提供了高效的多维数组对象和丰富的数学函数,是Python科学计算生态系统的基础库,系统中的数据生成、数据预处理、概率计算等操作都依赖于NumPy的功能。

从数据流的角度来看,一次完整的AI任务执行流程是这样的,用户在展示层的任务中心页面选择一个任务类型,系统调用配置设置API更新后端的当前任务配置,用户接着在策略配置页面选择一个算法策略,系统再次调用API更新当前策略配置,用户在AI运行控制台页面点击运行按钮,展示层发送运行请求到业务逻辑层,业务逻辑层首先从单例配置管理器中读取当前的任务和策略配置,然后使用工厂模式根据任务类型创建相应的模型实例,使用策略模式根据策略类型创建算法上下文对象,调用模型实例的execute方法执行模板方法定义的标准流程,在流程中调用策略对象进行实际的AI计算,计算完成后将结果通过观察者模式发布给观察者对象,将执行记录添加到历史列表中,最后将结果数据返回给展示层,展示层收到结果后更新界面显示并跳转到结果分析页面,结果分析页面从Store中读取最新的结果数据并使用Chart.js绘制可视化图表。

从技术栈的角度来看,前端使用了Vue3作为核心框架,Vue3相比Vue2在性能、类型支持、组合式API等方面都有显著提升,组合式API通过setup函数和组合函数的方式组织组件逻辑,相比选项式API更加灵活和易于复用。Vue Router4是Vue3配套的官方路由库,提供了声明式路由配置、编程式导航、路由守卫等功能,支持HTML5 History模式和Hash模式两种路由模式。Pinia是Vue3推荐的状态管理库,相比Vuex更加轻

量和类型友好,提供了更简洁的API和更好的TypeScript支持。

Vite是新一代的前端构建工具,利用浏览器原生的ES模块支持实现了快速的冷启动和热更新,相比传统的webpack构建工具有明显的性能优势,Vite还内置了对Vue单文件组件、CSS预处理器、图片资源等的支持,开箱即用无需复杂配置。Chart.js是一个简单而灵活的JavaScript图表库,支持八种常见的图表类型,提供了丰富的配置选项和插件扩展机制,可以满足大多数数据可视化需求。Axios是一个基于Promise的HTTP客户端库,支持浏览器和Node.js环境,提供了请求拦截、响应拦截、请求取消、超时设置等功能,是前端应用中最常用的HTTP请求库之一。

后端使用了Flask3作为核心框架,Flask是一个轻量级的Web应用框架,遵循WSGI规范,提供了路由系统、请求处理、响应生成、模板渲染等Web开发的基础功能,Flask的设计哲学是保持核心简单,通过扩展机制提供额外功能,这使得Flask既适合小型应用也适合大型应用。Flask-CORS是Flask的跨域资源共享扩展,通过简单的装饰器或全局配置就可以为API接口添加CORS支持,解决前后端分离架构中的跨域问题。NumPy是Python的科学计算基础库,提供了多维数组对象和矩阵运算函数,支持向量化操作以提高计算效率,是数据科学和机器学习领域必不可少的工具库。

Scikit-learn是Python的机器学习库,提供了分类、回归、聚类、降维等常见机器学习算法的实现,虽然本系统中主要使用NumPy进行数据生成和计算,但在实际应用中可以很容易地集成Scikit-learn的算法模型。从部署的角度来看,前端应用可以通过Vite的build命令打包成静态文件,然后部署到任何支持静态文件托管的服务器上,比如Nginx、Apache或云存储服务。后端应用可以使用Gunicorn或uWSGI等WSGI服务器运行,配合Nginx作为反向代理服务器,实现负载均衡和HTTPS支持。

从开发流程的角度来看,系统的开发遵循了前后端分离的开发模式,前端和后端可以由不同的开发人员并行开发,只需要事先约定好API接口的规范即可,这种开发模式提高了开发效率,也使得前端和后端可以使用各自最擅长的技术栈。在开发阶段,前端使用Vite的开发服务器运行在3000端口,后端使用Flask的开发服务器运行在5000端口,通过Vite的代理配置将前端的API请求转发到后端服务器,实现了开发环境的前后端联调。在测试阶段,可以使用Postman或curl等工具测试后端API接口的功能和性能,使用浏览器的开发者工具查看前端的网络请求和控制台输出,使用Vue Devtools插件查看组件树和状态数据。

5 小组成员及分工

本项目由软件工程23-15班的三名同学共同完成,项目团队采用敏捷开发的协作模式,通过定期的团队会议进行需求讨论、进度同步和问题解决,确保项目按计划顺利推进。团队成员之间建立了良好的沟通机制,使用即时通讯工具进行日常交流,使用代码版本控制系统进行代码共享和协作开发,使用项目管理工具跟踪任务进度和问题列表,形成了高效的团队协作流程。

组长王佳兴同学在项目中承担了核心架构设计和后端开发的主要职责,负责整个系统的技术选型、架构设计和开发规范制定等基础性工作。在项目启动阶段,王佳兴同学进行了详细的需求分析和可行性研究,调研了多种技术方案并进行对比评估,最终确定了使用Vue3和Flask构建前后端分离架构的技术路线,明确了系统要实现的功能范围和性能指标。在架构设计阶段,王佳兴同学绘制了系统的整体架构图、模块划分图和数据流程图,设计了API接口的规范文档,包括接口的URL路径、请求方法、请求参数、响应格式等详细信息,为后续的开发工作提供了清晰的指导。

在后端开发阶段,王佳兴同学负责搭建Flask应用的基础框架,配置了跨域支持、日志记录、错误处理等基础设施,实现了五种设计模式的代码结构,编写了所有的API路由处理函数,开发了数据生成和处理逻辑。王佳兴同学在编写设计模式代码时特别注重代码的规范性和可读性,遵循Python的PEP8编码规范,为每个类和方法编写了详细的文档字符串,使用类型注解增强代码的类型安全性,通过抽象基类和接口定义明确了各个模块之间的契约关系。在开发过程中,王佳兴同学还承担了技术难题的攻关工作,比如如何使用NumPy生成符合统计规律的随机数据,如何实现观察者模式的通知机制,如何设计灵活的工厂方法等,通过查阅文档、搜索资料和编写测试代码,成功解决了这些技术问题。

组员潘嘉鑫同学主要负责前端界面的开发和用户体验的优化工作,在UI设计方面展现了出色的审美能力和细节把控能力。潘嘉鑫同学首先学习了Ant Design等主流设计系统的设计规范,总结了现代Web应用界面设计的通用原则和最佳实践,比如使用浅色背景、保持足够的留白、采用卡片化布局、提供清晰的视觉层次、使用合理的颜色搭配等。基于这些设计原则,潘嘉鑫同学使用Figma工具绘制了系统各个页面的设计稿,包括布局结构、配色方案、字体样式、图标选择等视觉元素,经过多次迭代和优化,最终形成了清爽专业的界面风格。

在前端开发阶段,潘嘉鑫同学使用Vue3框架将设计稿转化为实际的代码实现,编写了六个页面组件的模板、脚本和样式代码,实现了各种交互效果和动画效果,比如按

钮的hover效果、卡片的阴影效果、列表的展开收起效果等。潘嘉鑫同学特别注重代码的组件化和复用性,将一些通用的UI元素抽取为独立的子组件,比如统计卡片组件、数据列表组件、操作按钮组件等,通过Props传递数据和事件回调,提高了代码的复用性和可维护性。在样式编写方面,潘嘉鑫同学采用了BEM命名规范组织CSS类名,使用了CSS变量定义主题颜色,使用了Flexbox和Grid布局实现响应式设计,确保页面在不同屏幕尺寸下都能正常显示。

潘嘉鑫同学还负责集成Chart.js图表库,学习了Chart.js的配置选项和使用方法,编写了图表组件的封装代码,实现了从后端数据到图表展示的自动转换。为了提供良好的用户体验,潘嘉鑫同学在页面中添加了加载状态提示、错误提示、成功提示等反馈信息,使用了骨架屏技术在数据加载时显示占位内容,使用了防抖和节流技术优化高频操作的性能,这些细节处理显著提升了应用的用户体验。在开发过程中,潘嘉鑫同学还进行了浏览器兼容性测试,确保应用在Chrome、Firefox、Safari、Edge等主流浏览器中都能正常运行。

组员邢纹源同学主要负责系统集成测试和文档编写工作,在保证项目质量方面发挥了重要作用。在测试阶段,邢纹源同学制定了详细的测试计划和测试用例,覆盖了系统的所有功能模块和业务流程,进行了功能测试、性能测试、兼容性测试等多种类型的测试。功能测试验证了每个功能是否按照需求文档正确实现,比如任务选择功能是否能够正确设置当前任务,策略切换功能是否能够正确改变算法行为,模型执行功能是否能够正确返回结果数据等。邢纹源同学在测试过程中发现了多个bug和问题,比如某些情况下页面显示异常、某些API接口返回错误、某些边界条件未正确处理等,及时反馈给开发人员进行修复,确保了系统的稳定性和可靠性。

性能测试评估了系统在不同负载下的响应时间和吞吐量,邢纹源同学使用了性能测试工具模拟多个并发用户访问系统,观察系统的CPU使用率、内存占用、网络延迟等指标,识别出性能瓶颈并提出优化建议。兼容性测试验证了系统在不同操作系统、不同浏览器、不同屏幕分辨率下的表现,邢纹源同学在Windows、macOS、Linux等操作系统上进行了测试,在Chrome、Firefox、Safari、Edge等浏览器中进行了测试,在笔记本电脑、台式机显示器、平板设备等不同设备上进行了测试,确保了系统的广泛兼容性。

在文档编写方面,邢纹源同学负责撰写了项目的需求规格说明书、设计文档、开发文档、测试报告、用户手册等多份技术文档,这些文档详细记录了项目的需求分析、系统设计、实现细节、测试结果、使用方法等信息,为项目的开发、维护和使用提供了重要的参考资料。需求规格说明书明确了系统要实现的功能需求和非功能需求,采

用用例图和用例描述的方式详细说明了用户与系统的交互过程。设计文档描述了系统的架构设计、模块设计、接口设计、数据库设计等内容,使用UML类图、时序图、组件图等图形化工具辅助说明设计思路。

开发文档记录了代码的目录结构、命名规范、编码规范、构建流程、部署步骤等开发相关的信息,帮助新加入的开发人员快速了解项目的技术细节。测试报告总结了测试的范围、方法、用例、结果和缺陷统计,使用表格和图表的形式清晰呈现测试数据,提出了改进建议和后续的测试计划。用户手册以用户的视角介绍了系统的功能特性和操作步骤,使用大量的截图和示例帮助用户理解系统的使用方法,采用了通俗易懂的语言避免过多的技术术语,确保了文档的可读性和实用性。邢纹源同学还整理了项目的常见问题解答文档,收集了在开发和测试过程中遇到的各种问题及其解决方案,为后续的维护工作提供了宝贵的参考。

三位团队成员在项目开发过程中保持了密切的沟通和协作,定期举行团队会议讨论项目进展和遇到的问题,及时调整开发计划和任务分配,确保项目按时保质完成。团队成员之间互相学习和帮助,王佳兴同学向其他成员分享了设计模式的应用经验和后端开发的技术细节,潘嘉鑫同学向其他成员展示了前端开发的最新技术和设计趋势,邢纹源同学向其他成员传授了测试方法和文档写作的技巧,通过这种知识共享和技能传递,每个成员都得到了成长和提升。在项目即将完成时,三位成员共同进行了系统的最终测试和文档的最终审核,确保了交付成果的质量和完整性,为项目画上了圆满的句号。

通过本次课程设计项目,三位团队成员不仅掌握了Web应用开发的完整流程和关键技术,深入理解了设计模式在实际项目中的应用价值,还锻炼了团队协作能力、问题解决能力和项目管理能力,为将来从事软件开发工作打下了坚实的基础。项目成果充分体现了团队的技术水平和专业素养,展示了现代软件工程的规范化开发过程,达到了课程设计的教学目标和能力培养要求,是一次成功的实践教学活动。