CS304 Project Proposal

Intelligent Coursework Grading System: MujiCLASS

01.07.2024

Chen Kangrui, Gong Linghu, Kuang Liang, Yu Siyao & Xue Zhe

Contents

1 Prelimilary Requirement Analysis

Functional Requirements

1. Integrated Coursework Management

All-in-one resource management for coursework, including resource releasing, assignment creation, submission, grading, and feedback.

2. Study with Classmates

Share your notes on lecture slides, discuss with classmates, and ask questions in the discussion zone. All of the notes will be saved in the cloud, and can be viewed by all students in the class in the form of bullet chatting.

3. Open-review Arguement

Students can review the grading results of their assignments, and argue with the teachers if they think the grading is unfair. Students in the same class can also mark the review and join the discussion.

Functional Requirements

4. AI grading

Powered by Deepseek-R1 API, The system will provide AI marking for the assignments, enabling the teachers configure the marking rules and the system will automatically mark the assignments according to the rules.

5. Multi-platform

Use with ease on web browser, Windows, Linux and MacOS.

Non-functional Requirements

1. Simplified Functions

Traditional course management systems, such as Blackboard and Sakai, come with **numerous redundant features** that are not utilized in teaching at SUSTech. For MujiCLASS, we aim to retain essential functionalities to **simplicity**.

2. Easy to Use

We will design a simple, modern UI and user-friendly operating logic to make it easy for both students and teachers to use.

3. High-performance System

Systems like Blackboard or Sakai often require **refreshing the entire webpage**, which makes them slow to use. They can also **be slow when loading lots of data**, like grades, or when many people are **using them at the same time**, causing delays in submissions. We will focus on improving performance by using caching and high-performance frameworks for both the front and back end.

Data Requirements

1. Student, Teacher and Administrator Accounts

We need information about students, teachers, and administrators, including their names, email addresses, departments, etc. for testing. **These data can be easily mocked by Python `faker` library.**

2. Coursework Resources

We need information about coursework resources, including slides assignment names, descriptions, deadlines, etc. for testing. **We have collected several existing course materials for this.**

3. Open-review Arguement

We need data for open-review arguement testing. **These data can also be easily mocked.**

4. AI-assisted Grading

We need to manually make rules for AI grading, and test the system with these rules. **This kind of data will be collected while building and testing AI grading system.**

Technical Requirements

The system will be developed based on the classic **front-end and back-end separation architecture.**

1. Front-end

- **Multi-platform Support**: Electron
- **Web-based**: Vue.js + Element Plus + Vite
- **Store**: Pinia
- **Requests**: Axios
- **PDF support**: pdf.js

2. Back-end

- **Language**: Python
- **Framework**: FastAPI (w/ Pydantic) + Uvicorn
- **Database & ORM**: SQLModel + PostgreSQL
- **Caching**: Redis
- **AI**: Deepseek-R1 API

Technical Requirements

3. Deployment

- **Front-end**: A Web-based version will be deployed on a server with CDN (e.g. Vercel), and a desktop version will be released for Windows, Linux, and MacOS.
- **Back-end**: The back-end will be deployed on a server with Docker, and the database will be deployed on a separate server. A Redis server will be deployed for caching. Nginx will be used as a reverse proxy.
- **CI/CD**: GitHub Actions.

4. Project Planning &

- **API Docmentation**: Swagger (generated by FastAPI) and Apifox.
- **Testing**: Pytest + Apifox