**Student: 2018314410 박병규**

**2017312428전승태**

**WEB PROJECT**

**REPORT**

**[Edu Note]**

# Login and Register Page

## [Register Page]

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[Main Function]

* Users must log in to use edunote.
* Use the email as user ID. user must meet the email standards to sign up for member.
* The password must be at least 8 characters in English, numbers, and capital letters, which are also checked.
* If user have an account, user can go to the login page.

## [Login Page]

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[Main Function]

* Enter user email and password to proceed with the login process.
* If the account does not exist, you can go to the membership page.
* If the ID password does not match, login is not possible.
* If the ID password matches, log in and go to the home screen.

# Home Page

## [Home Page]

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

[Main Function]

* There are basically three features in the sidebar: Home screen, My Note, and Logout
* User can refer to the date information in the calendar on the sidebar.
* Press the Start Note button to go to the Note page.

# My Notes

## [My Notes]

**노트 사진 구현후 추가작성**

## [Add Note]

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

[Main Function]

* user can upload the title of the note and the lecture file by pressing the Add Note button.

## [Message Page]

텍스트, 스크린샷, 소프트웨어, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

[Main Function]

* user can exchange conversations on the message page.
* Clova summarizes the lecture contents of the file uploaded by the user.

# Database with fast api

## What is FastAPI

FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.6+ based on standard Python type hints. Developed by Sebastián Ramírez and first released in December 2018, it has quickly become one of the most popular frameworks for API development due to its speed, ease of use, and robust feature set. The key features of FastAPI are show below.

1. High Performance: Comparable to NodeJS and Go, FastAPI is one of the fastest web frameworks for Python, thanks to its Starlette framework for the web parts and Pydantic for the data parts.
2. Type Checking and Validation: Utilizes Python 3.6+ type hints to validate data, reduce bugs, and enable auto-completion in editors, enhancing developer productivity and code quality.
3. Automatic Documentation: Offers automatic generation of interactive API documentation using Swagger UI and ReDoc. This feature is invaluable for both testing and documenting your API endpoints.
4. Ease of Use: Designed to be easy to use and learn. It has a simple and intuitive API, reducing the learning curve for new developers.
5. Asynchronous Code Support: Natively supports asynchronous request handling, making it an excellent choice for handling high volumes of traffic and I/O-bound operations, enhancing scalability and performance.
6. Data Conversion and Validation with Pydantic: Incorporates Pydantic for data validation and settings management, using Python type annotations to validate, serialize, and deserialize data, and automatically convert it to and from JSON.
7. Security and Authentication: Provides built-in support for authentication and authorization, including support for OAuth2 with JWT tokens and HTTP Basic Auth.
8. Dependency Injection System: Offers a powerful dependency injection system, allowing you to share logic (like database sessions) and data across endpoints with ease.

We have made 7 api calls as shown below.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

## [Signup]

This method receives a user's id and password from the body of a JSON request. It conducts a validity check by searching for the entered id in the server. A folder is created for each user's id, and the website is structured based on the files within this folder. The password is stored in a hidden file to maintain security. This method generates a JSON file indicating the existence of the id and the success of the registration, which is then sent as a response.

## [Signin]

This method also retrieves a user's id and password from the body of a JSON request. It first searches for a folder with the same user name, and then compares the password inside the folder to proceed with the login process. Similar to the Signup method, it generates a JSON file indicating the existence of the id and the success of the login, which is sent as a response.

## [GetNoteList]

This method fetches the entire list of notes created by the user. Utilizing the GET method, it is structured to request with the user's id in the URL and retrieves a list of all the notes in the server's folder. The list of notes, along with the creation and modification times of each note, is packaged into a JSON object and sent as a response.

## [GetNote]

This method retrieves the content of scripts uploaded by the user. These script files are initially stored when an audio file is uploaded. The script files are converted into text files via Text-to-Speech (TTS) technology. This method is responsible for reading the text file's entire content and delivering it as a string.

## [UploadNote]

This method was devised to save simple memos that could be written alongside the notes. However, it is not frequently used as it diverges from the main purpose of this web service.

## [UploadSpeech]

This method plays a pivotal role in storing audio files on the server. It represents the core API call, encompassing the processes of copying the audio file, converting the audio file to text, and training based on the text using the langchain model. All files are stored on the server, and the qa mentioned later is based on the data learned in this process.

## [GetQa]

This method performs question and answering. EduNote is extremely useful for modern people who lack time to revisit lectures. By asking EduNote questions, users can get answers based on the lecture content, aiding in a deeper understanding of the core material. All qas are conducted through the previously mentioned langchain and utilize the **Faiss Vector DB**.

# Tech and How to use

* At the front end, it was conducted based on react and type script. React query, axios, recoil, react-router-dom, tailwind, and material-ui were used.
* At the back end, it was conducted based on FastAPI and Faiss Vector DB. Backend execute fundamental functions, such as login, and also get answer by querying.
* In frontend folder
* Npm install
* Npm start
* In backend folder
* pip install fastapi
* pip install "uvicorn[standard]"
* pip install langchain==0.0.300
* pip install openai==0.28.0
* pip install python-multipart
* uvicorn backend:app --reload --host=localhost --port=8080

# Code History

You can check the development history in our Github link.

<https://github.com/pbk95120/wep-final>