

Spring과 Vue.js를 활용한 언어번역 웹사이트 프로젝트

multicampus

소속명	지능형 웹서비스 플스택 개발 B반		
팀명	PAGO BOOKS	개발기간	2022. 2. 03. ~ 2022. 3. 28
팀장	정다훈	개발자	정다훈, 김희태, 강하중, 장지원, 이동근, 신혜지

목차

1. 프로젝트 배경	
1.1 프로젝트 주제 및 선정 배경	3
1.2 프로젝트 목적	3
1.3 프로젝트 개요	3
2. 프로젝트 팀 구성 및 역할	
2.1 팀 구성 및 역할	3
3. 프로젝트 수행 절차 및 방법	
3.1 일정 계획	4
3.2 노력 추정	4
4. 수행 결과	
4.1 프로젝트 범위	5
4.2 데이터베이스 설계	5
4.3 요구사항	9
4.4 정책	10
4.5 웹 페이지 계층 구조	12
4.6 실행 화면	13
4.7 소스 코드	26
4.8 시연	40

1. 프로젝트 배경

1.1 프로젝트 주제 및 선정 배경

온라인 수업에서 다양한 매체와 자료를 교재로 활용하게 됨에 따라 수업의 질이 기존 방식에 비해 비약적으로 향상되었습니다. 그러나 그 다양한 자료들을 학생들이 필요에 따라 텍스트화나 번역 또는 이미지화를 해야 할 때 쉽게 도움을 받을 수 있는 플랫폼은 아직 빈약하다는 점에서 기인하여 이번 프로젝트를 계획하게 되었습니다.

1.2 프로젝트 목적

프로젝트 명, 가칭 트랜스파스트는 OCR, STT, TTS, 파파고 번역기 API 기능을 하나의 플랫폼에서 제공하는 비대면 수업자료의 문서화 서비스입니다. 저희 서비스를 통해 사용자가 비대면 수업에서 발생하는 자료들을 쉽게 문서화, 또는 자료 변환을 할 수 있도록 돕고, 번역기능을 제공하여 다국적 자료에도 적극적으로 대응할 수 있도록 할 예정입니다. 또한 타 서비스들에서 번거롭게 여겨졌던 과도한 기능들을 최대한 배제하고, 직관적인 구성을 통해 이용성과 편리성을 극대화하는 방향으로 개발하려고 합니다.

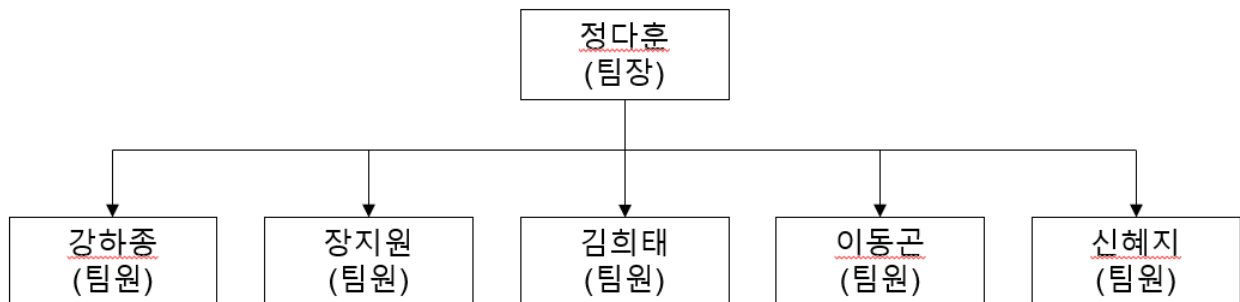
1.3 프로젝트 개요

언어번역을 지향하며 프론트 엔드는 Vue.js, 백엔드는 Spring Boot, 데이터베이스는 MYSQL을 이용해 만들 예정입니다.

2. 프로젝트 팀 구성 및 역할

2.1 팀 구성 및 역할

각 팀원은 맡은 부분을 팀 회의에서 설계한 방식대로 사용자 인터페이스와 기능을 구현하여 완성된 후 구현부를 통합하기로 했다.

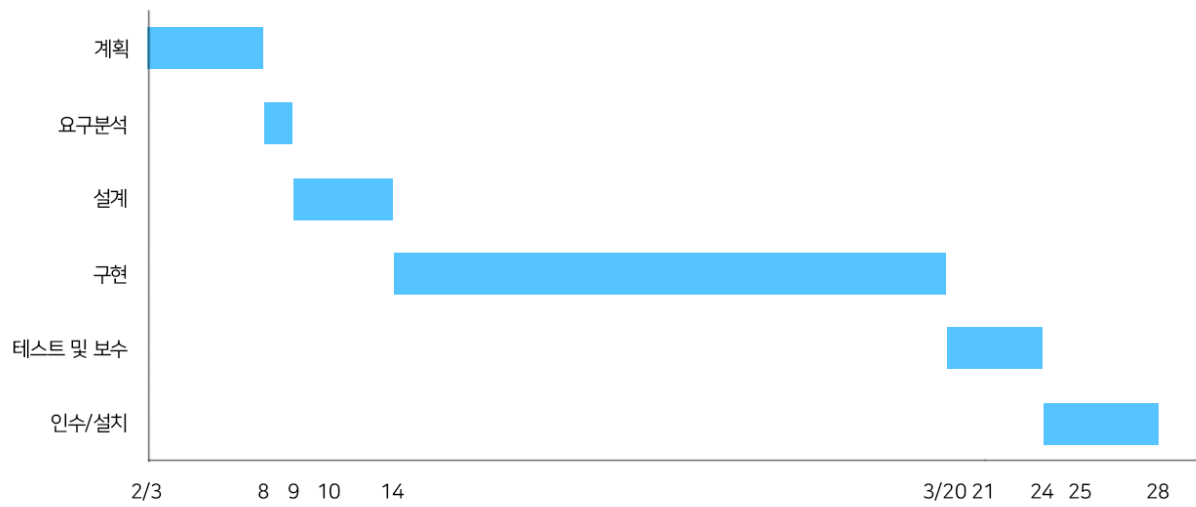


이름	Description
정다훈(팀장)	1. 로그인 회원가입 로직 기능을 제공한다. 2. ClovaAPI를 사용한 문서변환 기능을 제공한다 3. 로그인을 통해 변환된 데이터를 보관함에 저장 가능하게 한다.
김희태(팀원)	1. ClovaAPI를 사용한 매체변환 기능을 제공한다. 2. 로그인을 통해 변환된 데이터를 보관함에 저장 가능하게 한다.
강하중(팀원)	1. vue.js 를 이용한 Front page 개발한다. 2. 웹 페이지 아이콘 및 디자인
장지원(팀원)	1. ClovaAPI를 사용한 음성변환 기능을 제공한다. 2. 로그인을 통해 변환된 데이터를 보관함에 저장 가능하게 한다.
이동곤(팀원)	1. Papago API를 사용한 다국적 사용자를 위해 번역기능을 제공한다. 2. 로그인을 통해 번역된 데이터를 보관함에 저장 가능하게 한다.
신혜지(팀원)	1. 웹 페이지 아이콘 및 디자인

3. 프로젝트 수행 절차 및 방법

3.1 일정 계획

총 개발 기간	2022. 2. 03. ~ 2022. 3. 28. (54일)	
프로젝트명	언어번역 웹사이트	
프로젝트 개발기간	작업내용	개발 기간
	프로젝트 계획 및 요구분석	2022-02-03 ~ 2022-02-09
	설계	2022-02-10 ~ 2022-02-14
	프로그램 구현	2022-02-14 ~ 2022-03-20
	기능 시험 및 오류 수정	2022-03-21 ~ 2022-03-24
	데버깅 및 프로젝트 발표	2022-03-24 ~ 2022-03-28
작업 순서	주제선정 → 계획 → 설계 → 구현 → 시험 및 유지 보수	
필요 자원	Spring Boot 2.6.2, Vue.js 2.6.14, Git JAVA 8, My sql	



〈그림 1. 간트 차트〉

3.2 노력추정

구분	세부내역	산출금액
직접인건비(A)	기획자 : 1명 X 2일 X 150,000원(단가) 개발자 : 6명 X 15일 X 100,000원(단가)	9,300,000 (원)
제경비(B)	직접인건비 * 100%	9,300,000 (원)
기술료(C)	(직접인건비 + 제경비) * 20%	3,720,000 (원)
합계 금액(A+B+C)		22,320,000 (원)

4. 수행 결과

4.1 프로젝트 범위

유저

구현기능	상세 내용
회원	로그인 로그아웃
기능	문서변환 음성변환 매체번역 간단번역 보관함(저장기능)

구현기능	상세 내용
비회원	단순 문서변환 단순 음성변환 단순 매체번역 단순 간단번역

4.2 데이터베이스 설계

개체	USER					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고
기본키	회원 ID	ID	INT			
속성	회원 Email	EMAIL	VARCHAR	50		
	회원 비밀번호	PASSWORD	VARCHAR	200		암호화 처리
	회원 프로필	PROFILE	VARCHAR	200		Auth 2.0 사용

개체	ROLE					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고
기본키		ID	INT			
후보키		NAME	VARCHAR	30		

개체	USER_ROLES					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고
기본키 후보키		USER_ID	INT			
		ROLE_ID	INT			

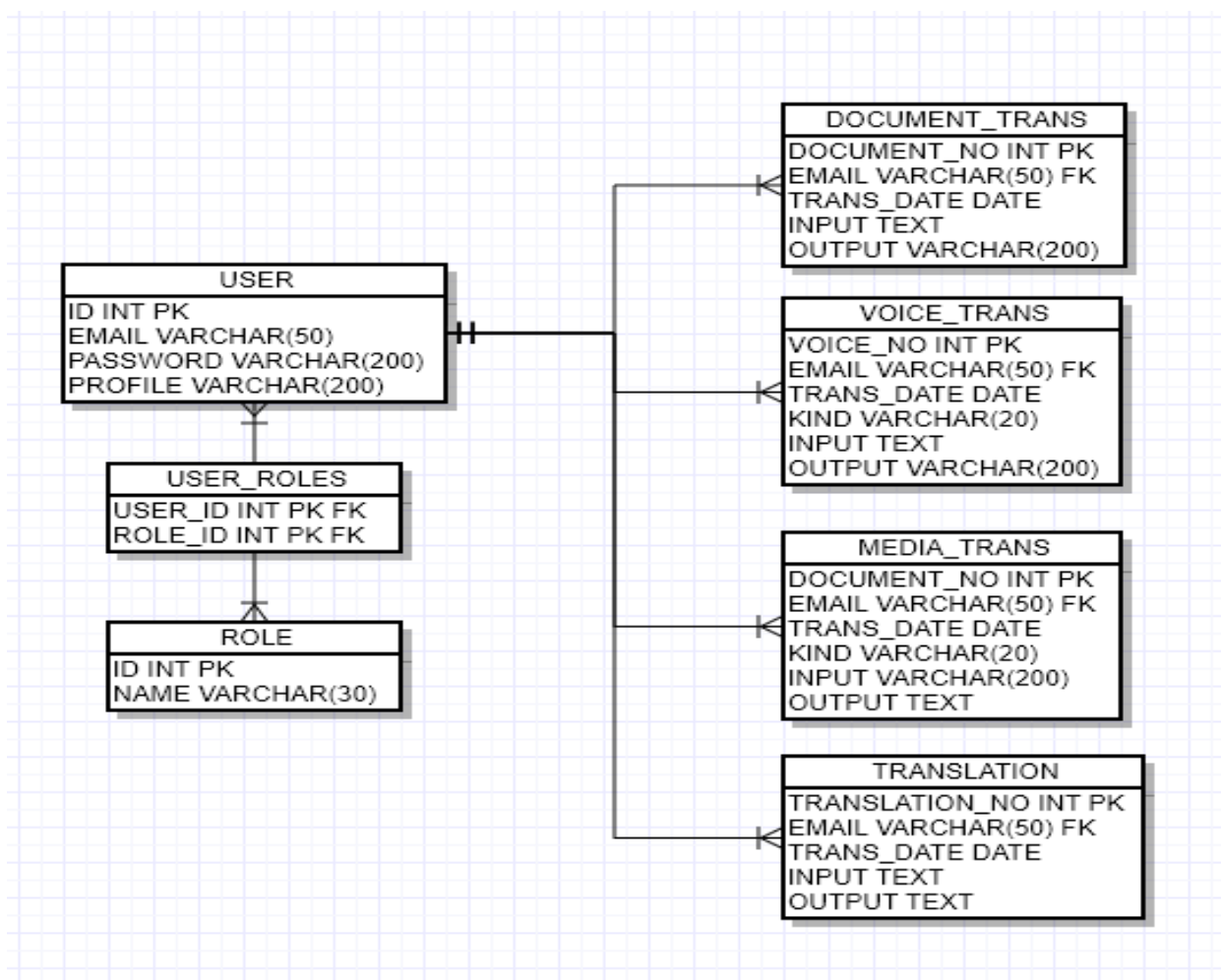
개체	DOCUMENT_TRANS					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고

기본키		DOCUMENT_NO	INT			
후보키		EMAIL	VARCHAR	50		
속성	변환 시각	TRANS_DATE	DATE			
	입력 자료	INPUT	TEXT			
	출력 자료	OUTPUT	VARCHAR	200		

개체	MEDIA_TRANS					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고
기본키		DOCUMENT_NO	INT			
후보키		EMAIL	VARCHAR	50		
속성	변환 시각	TRANS_DATE	DATE			
		KIND	VARCHAR	20		
	입력 자료	INPUT	VARCHAR	200		
	출력 자료	OUTPUT	TEXT			

개체	VOICES_TRANS					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고
기본키		VOCIE_NO	INT			
후보키		EMAIL	VARCHAR	50		
속성	변환 시각	TRANS_DATE	DATE			
		KIND	VARCHAR	20		
	입력 자료	INPUT	TEXT			
	출력 자료	OUTPUT	VARCHAR	200		

개체	TRANSLATION					
구분	Logical	Physical				
		Column	Data type	Size	Null	비고
기본키		TRANSLATION_NO	INT			
후보키		EMAIL	VARCHAR	50		
속성	변환 시각	TRANS_DATE	DATETIME			
	입력 자료	INPUT	TEXT			
	출력 자료	OUTPUT	TEXT			



〈그림 2. DB 모델링〉

4.3 요구 사항

요구사항 명	내용
로그인	<ul style="list-style-type: none"> ◆ Email 과 Password 를 입력받는다 ◆ 회원가입을 제공하지 않고 구글, 네이버, 깃허브 로그인 API 를 사용한다
문서변환	<ul style="list-style-type: none"> ◆ 이미지를 업로드 받는다 ◆ 확장자 JPG, PNG, PDF, TIFF 를 허용한다 ◆ 이미지 파일 사이즈는 20MB 까지 허용한다 ◆ 인식 언어는 한국어, 영어, 일본어로 제한된다 ◆ 필기체일 경우 인식 언어는 한국어, 일본어이다 ◆ 다운로드 및 미리보기를 지원한다
음성변환	<ul style="list-style-type: none"> ◆ 변환할 텍스트를 입력한다 ◆ 확장자 MP3, WAV ◆ 65개의 Speaker 중 택한다 ◆ 변환 가능 언어는 한국어, 영어, 일본어, 중국어, 대만어, 스페인어로 총 6개이다 ◆ 음성 볼륨, 속도 조절이 가능하다 -5 ~ 5 *정수값 ◆ 음성 감정 조절이 가능하다 0 ~ 2 (기본 0, 어두운 1, 밝은 2) ◆ 다운로드 및 미리듣기를 지원한다
매체변환	<ul style="list-style-type: none"> ◆ 텍스트로 입력받은 파일을 업로드한다 ◆ 확장자는 MP3, AAC, AC3, OGG, FLAC, WAV ◆ 인식 언어는 한국어, 일본어, 중국어, 영어이다 ◆ 최대 12MB의 용량, 시간은 60초로 제한된다 ◆ 다운로드 및 미리보기를 지원한다
간단번역	<ul style="list-style-type: none"> ◆ 변환할 텍스트를 입력한다 ◆ 확장자 .txt ◆ 변환 가능 언어는 한국어, 영어, 중국어, 일본어로 총 4개이다 ◆ 1회 권장 최대 60,000자를 지원한다(UTF-8기준 공백포함 200,000bytes) ◆ 다운로드 및 미리보기를 지원한다

4.4 정책

4.4.1 권한

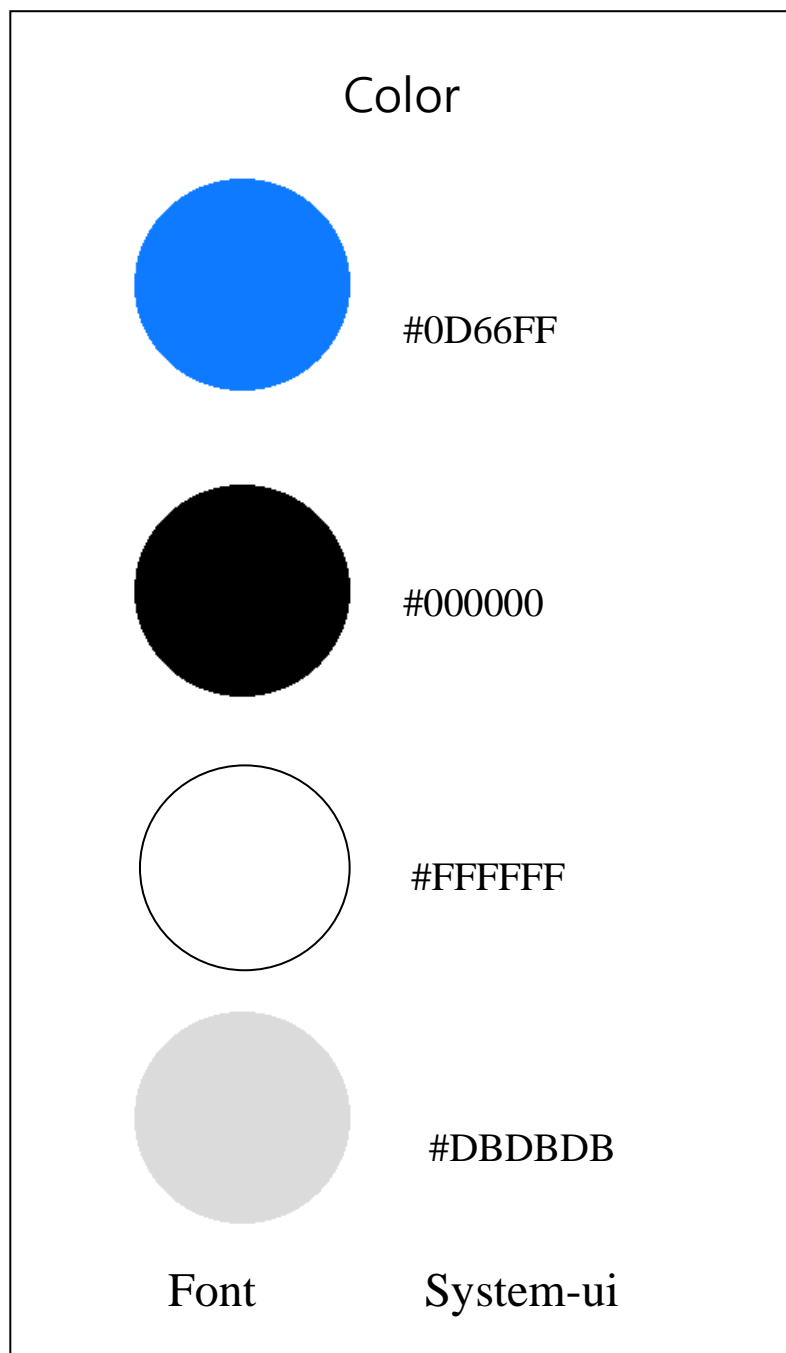
번호	정책코드	세부항목	사용자	권한			
				보기	수정	삭제	쓰기
1	front-page-01	메인페이지	로그인	0	X	X	0
			비로그인	0	X	X	0
2	front-page-02	문서변환페이지	로그인	0	X	X	0
			비로그인	0	X	X	0
3	front-page-03	음성변환페이지	로그인	0	X	X	0
			비로그인	0	X	X	0
4	front-page-04	매체번역페이지	로그인	0	X	X	0
			비로그인	0	X	X	0
5	front-page-05	간단번역페이지	로그인	0	X	X	0
			비로그인	0	X	X	0
6	front-page-06	보관함페이지	로그인	0	X	0	X
			비로그인	X	X	X	X
7	front-page-07	문서변환 보관함 페이지	로그인	0	X	0	X
			비로그인	X	X	X	X
8	front-page-08	음성변환 보관함 페이지	로그인	0	X	0	X
			비로그인	X	X	X	X
9	front-page-09	매체번역 보관함 페이지	로그인	0	X	0	X
			비로그인	X	X	X	X
10	front-page-10	간단번역 보관함 페이지	로그인	0	X	0	X
			비로그인	X	X	X	X

4. 4. 3 디자인 가이드

1) 로고

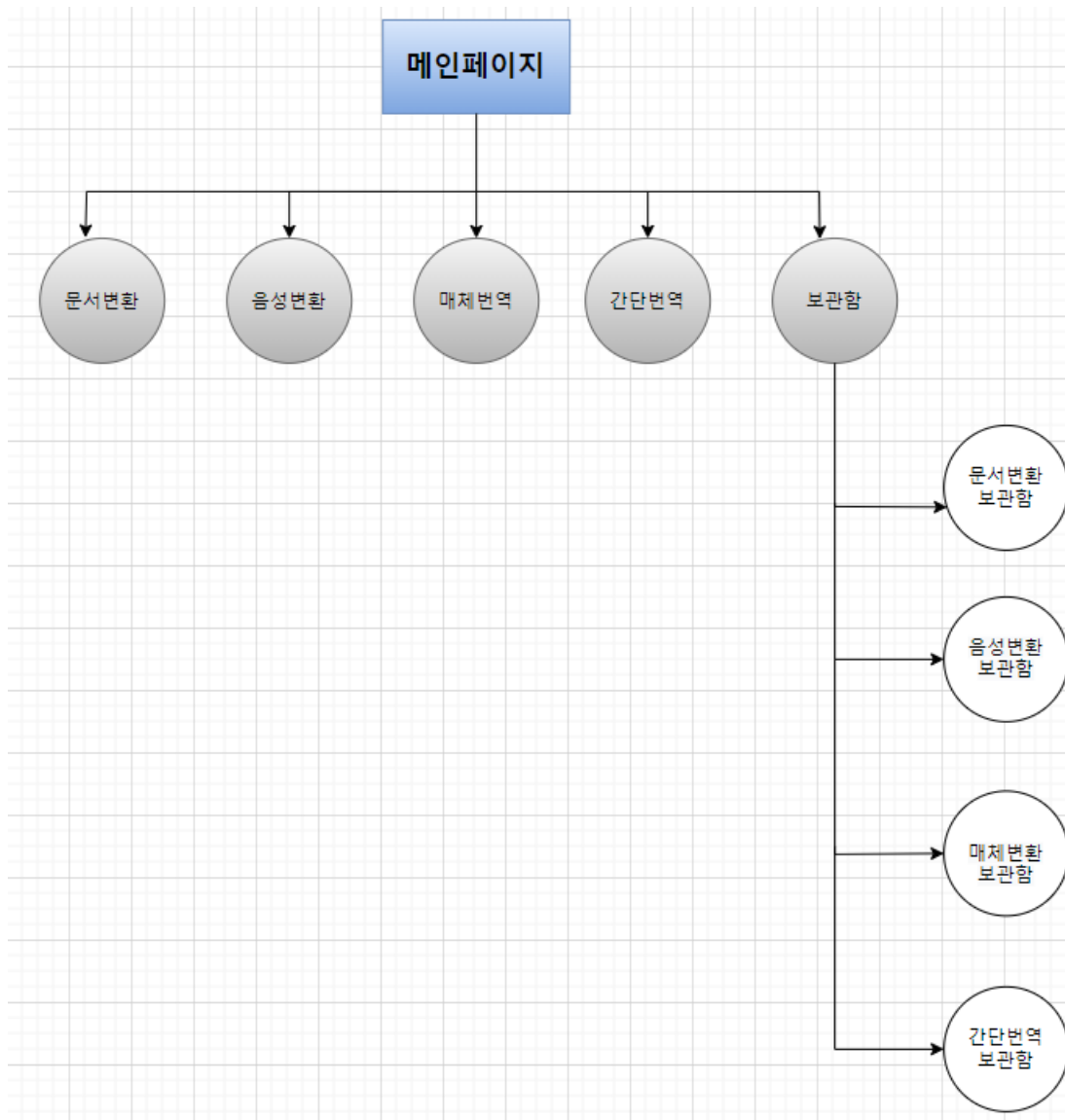


2) 색상 및 폰트



4. 5

웹 페이지 계층 구조



4.6 실행 화면



Footer

[Tip : 로그인을 하시면 자료를 보관하고 내려받을 수 있습니다]

[파고북스 이용약관](#) [의견제안](#) [개인정보처리방침](#) [책임의 한계와 법적고지](#) [준수사항](#)

로그인모달

PAGO BOOKS

당신의 자료를 손쉽게 번역하고, 변환하고, 저장해보세요!

로그인

1

kakao

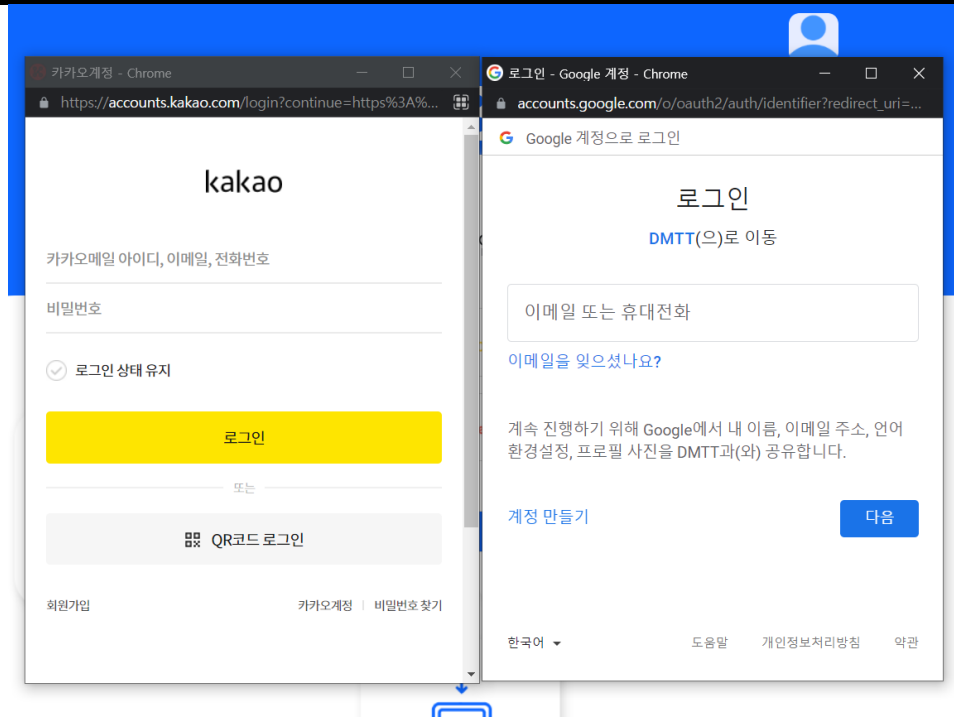
2

Google

닫기

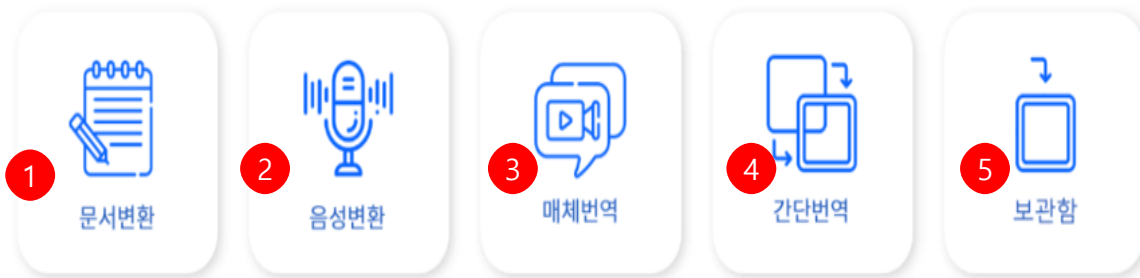
[Tip : 로그인을 하시면 자료를 보관하고 내려받을 수 있습니다]

로그인



- 1.카카오 로그인 : 카카오 로그인 팝업창 띄우기
- 2.구글 로그인 : 구글 로그인 팝업창 띄우기

메인화면(상세)



- 1.문서변환 : 문서변환 페이지로 이동
- 2.음성변환 : 음성변환 페이지로 이동
- 3.매체번역 : 매체번역 페이지로 이동
- 4.간단번역 : 간단번역 페이지로 이동
- 5.보관함 : 보관함 페이지로 이동(로그인한 이용자만 사용가능)

문서변환

1

채팅하듯 술술 읽히는
채팅형 독서 콘텐츠

친구

아니, 그게 뭐야?

나

“밀린 책” 들어봤어?

친구

채팅처럼 특특 읽는 쉬운 채팅형 독서 콘텐츠야!

나

읽기 어려운 책도 쉽게 읽을 수 있겠네?

나

그럼! 어려운 영어서부터 말랑말랑 willie랑 한소설까지 다 있어!

파일 가져오기

2

채팅하듯 술술 읽히는
채팅형 독서 콘텐츠

나

“밀린 책” 들어봤어?

친구

아니, 그게 뭐야?

채팅처럼 특특 읽는 쉬운 채팅형 독서 콘텐츠야!

친구

읽기 어려운 책도 쉽게 읽을 수 있겠네?

나

그럼! 어려운 영어서부터 말랑말랑 willie랑 한소설까지 다 있어!

한국어

영어

3

번역하기

4

저장하기

localhost:8081 내용:
저장이 완료되었습니다.

확인

localhost:8081 내용:
지원하지 않는 번역기입니다.

확인

1. 파일 가져오기

파일 탐색기에서 입력받을 이미지를 선택함. OCR API 정의서 참조

2. 문서변환

처리가 완료되면 자동으로 변환된 글자가 OUTPUT 박스에 출력
해당 기능이 완료되기 전까지 로딩창을 띄움

3.간단번역

2번에 담긴 언어를 왼쪽 드롭다운 메뉴로 선택하고 번역할 언어를 오른쪽 드롭다운 메뉴에 선택하여 번역하기를 클릭하면 2번에 담긴 언어가 오른쪽 드롭다운 메뉴에 선택된 언어로 번역된다.
언어는 PAPAGO API 정의서 참조 정의서에 정의되어 있지 않은 기능 사용시 예외처리 하여 alert창 생성

4.저장하기

로그인 되어 있는 계정에 한하여 보이는 버튼
1번과 2번에 담겨있는 자료를 로그인된 계정 개인의 문서변환 보관함에 담긴다.

-17-

음성변환

1.텍스트 입력

음성 변환할 텍스트를 입력한다.

기본적인 레퍼런스는 placeholder로 설명되어있음.

2.음성 변환

입력받은 텍스트와 음성종류, 속도, 볼륨을 드롭다운 메뉴로 설정하여 음성변환하기 버튼을 클릭하면 저장하기 버튼과 음성변환된 audio플레이어, mp3다운로드 버튼이 생성됨

음성종류 속도 볼륨은 TTS API 정의서 참조

해당 기능이 완료되기 전까지 로딩창을 띄움

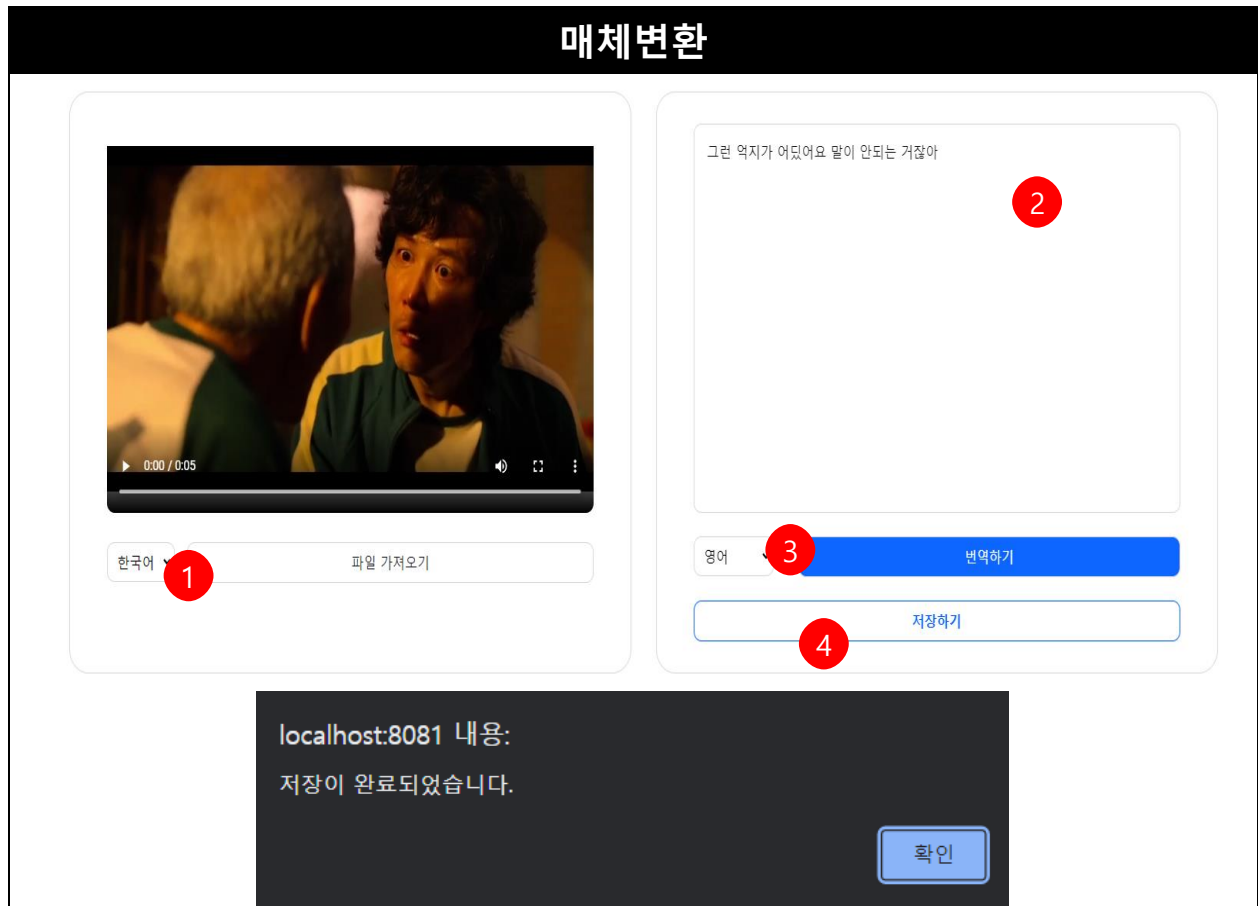
3.MP3 다운로드

입력받은 텍스트와 음성종류, 속도, 볼륨을 토대로 만들어진 audio 플레이어가 생성되며 다운로드 버튼을 클릭하면 해당 mp3파일을 다운로드 한다.

4.저장하기

로그인 되어 있는 계정에 한하여 보이는 버튼

1번의 자료와 음성변환 된 mp3 파일을 로그인된 계정 개인의 음성변환 보관함에 담긴다.



1.파일 가져오기

변환할 매체파일의 언어를 드롭다운메뉴로 설정 후 파일 가져오기 버튼을 클릭하여 변환할 매체파일을 선택한다. 해당 기능이 완료되기 전까지 로딩창을 띄움.

언어와 파일의 종류는 STT API 정의서 참조

2.매체변환

입력 받은 언어와 파일을 토대로 변환 되어 텍스트가 생성됨.

3.간단번역

2번에 담긴 언어를 왼쪽 드롭다운 메뉴로 선택하고 번역할 언어를 오른쪽 드롭다운 메뉴에 선택하여 번역하기를 클릭하면 2번에 담긴 언어가 오른쪽 드롭다운 메뉴에 선택된 언어로 번역된다.

언어는 PAPAGO API 정의서 참조

4.저장하기

로그인 되어 있는 계정에 한하여 보이는 버튼

1번과 2번에 담겨있는 자료를 로그인된 계정 개인의 매체변환 보관함에 담긴다.

간단번역

한국어

우리는 최종 프로젝트를 성공적으로 마쳤습니다.

번역하기

↔

영어

We have successfully completed the final project.

저장하기

localhost:8081 내용:
저장이 완료되었습니다.

확인

1.번역할 텍스트 입력

번역할 언어의 텍스트를 왼쪽의 드롭다운 메뉴로 선택하고 텍스트를 입력한다.

2.번역하기

1번 자료에 담긴 언어를 번역하고 싶은 언어를 오른쪽 드롭다운메뉴로 선택하고 번역하기를 클릭하면

2번 박스에 변환된 텍스트가 입력됨. 해당 기능이 완료되기 전까지 로딩창을 띄운다.

해당 API 레퍼런스는 PAPAGO API 정의를서를 참조

3.변환

1번에 담긴 자료와 2번에 담긴 자료의 위치를 바꾼다.

4.저장하기


로그인 되어 있는 계정에 한하여 보이는 버튼

1번과 2번에 담겨있는 자료를 로그인된 계정 개인의 간단번역 보관함에 담긴다.

보관함 페이지


1 문서변환
2 음성변환
3 매체번역
4 간단번역

1




로그인을 하시면 파일 저장과 보관함 이용이 가능합니다

2



파일을 저장을 했다면, 상단 메뉴에서 보관함을 선택하세요

3



저장한 파일을 확인하고 다운 받아 사용하세요

localhost:8081 내용:
로그인이 필요한 서비스입니다.

확인

1. 문서변환 페이지로 이동 (로그인된 회원만 이용 가능)
2. 음성변환 페이지로 이동 (로그인된 회원만 이용 가능)
3. 매체번역 페이지로 이동 (로그인된 회원만 이용 가능)
4. 간단번역 페이지로 이동 (로그인된 회원만 이용 가능)

문서변환 보관함

문서변환

음성변환

매체변역

간단번역

문서변환 2022-03-21T13:46:21.774

3 제

채팅하듯 술술 읽히는
채팅형 독서 콘텐츠



친구

아니, 그게 뭐야?

'밀리 챗북' 들어봤어?

1

텍스트 보기

다운로드

2

문서변환

음성변환

매체변역

간단번역

문서변환 2022-03-21T13:46:21.774

삭제

채팅하듯 술술 읽히는
채팅형 독서 콘텐츠

나

'밀리 챗북' 들어봤어?

친구

아니, 그게 뭐야?

채팅처럼 톡톡 읽는 쉬운 채팅형
독서 콘텐츠야!

친구

읽기 어려운 책도 쉽게 읽을
수 있겠네?

나

그럼 이걸로 먼저 시작해볼까요?

사진 보기

다운로드

1. 보관함 파일형식 변환

저장한 문서변환 파일을 텍스트 보기 버튼을 클릭하면 텍스트로 변환되어 보여지고 사진 보기 버튼으로 바뀌게 된다.
사진 보기 버튼을 클릭하면 텍스트로 되어 있는 자료가 보관되어 있는 자료로 바뀌고 텍스트 보기 버튼으로 바뀐다

2. 다운로드

다운로드 버튼을 클릭하게 되면 자료가 다운된다

사진 자료로 되어 있을 경우 해당 파일의 확장자로 다운이 된다

텍스트 자료로 변환하여 다운로드 버튼을 클릭하게 되면 .txt로 저장되어 다운이 된다

3. 삭제

해당 보관함에 있는 자료를 삭제한다

음성변환 보관함

문서변환

음성변환

매체번역

간단번역

음성변환 2022-03-21T13:59:19.663 3

▶ 0:00 / 0:07

🔊 ⋮

1 텍스트 보기

다운로드 2

문서변환

음성변환

매체번역

간단번역

음성변환 2022-03-21T13:59:19.663 삭제

가야 할 때가 언제인가를
분명히 알고 가는 이의
뒷모습은 얼마나 아름다운가.

오디오 보기

다운로드

1. 보관함 파일형식 변환

저장한 문서변환 파일을 텍스트 보기 버튼을 클릭하면 텍스트로 변환되어 보여지고 오디오 보기 버튼으로 바뀌게 된다.
오디오 보기 버튼을 클릭하면 텍스트로 되어 있는 자료가 보관되어 있는 자료로 바뀌고 텍스트 보기 버튼으로 바뀐다

2. 다운로드

다운로드 버튼을 클릭하게 되면 자료가 다운된다

오디오 자료로 되어 있을 경우 해당 파일의 확장자로 다운이 된다

텍스트 자료로 변환하여 다운로드 버튼을 클릭하게 되면 .txt로 저장되어 다운이 된다

3. 삭제

해당 보관함에 있는 자료를 삭제한다

매체번역 보관함

문서변환

음성변환

매체번역

간단번역

매체번역 2022-03-21T00:00:00

삭제 3



텍스트 보기

1

다운로드

2

문서변환

음성변환

매체번역

간단번역

매체번역 2022-03-21T00:00:00

삭제

그런 억지가 어딴어요 말이 안되는 거잖아

파일 보기

다운로드

1. 보관함 파일형식 변환

저장한 문서변환 파일을 텍스트 보기 버튼을 클릭하면 텍스트로 변환되어 보여지고 오디오 보기 버튼으로 바뀌게 된다.
파일 보기 버튼을 클릭하면 텍스트로 되어 있는 자료가 보관되어 있는 자료로 바뀌고 텍스트 보기 버튼으로 바뀐다

2. 다운로드

다운로드 버튼을 클릭하게 되면 자료가 다운된다

오디오,비디오 자료로 되어 있을 경우 해당 파일의 확장자로 다운이 된다

텍스트 자료로 변환하여 다운로드 버튼을 클릭하게 되면 .txt로 저장되어 다운이 된다

3. 삭제

해당 보관함에 있는 자료를 삭제한다

간단번역 보관함

[문서변환](#)[음성변환](#)[매체번역](#)[간단번역](#)

간단번역 2022-03-17T15:09:51.722

[삭제](#)

2

우리는 최종 프로젝트를 성공적으로 마쳤습니다.

We have successfully completed the final project.

[원문 다운로드](#)

1

[번역문 다운로드](#)

1. 다운로드

다운로드 버튼을 클릭하게 되면 자료가 다운로드

원문 다운로드를 클릭하게 되면 위쪽의 원문이 다운로드.

번역문 다운로드를 클릭하게 되면 아래쪽의 번역문이 다운로드.

2. 삭제

해당 보관함에 있는 자료를 삭제한다

4.7 소스코드

OCR API
<p>request 설정 부분 생략...</p> <pre>//RestAPI response 받기 int responseCode = con.getResponseCode(); BufferedReader br; if (responseCode == 200) { br = new BufferedReader(new InputStreamReader(con.getInputStream())); } else { br = new BufferedReader(new InputStreamReader(con.getErrorStream())); } String inputLine; StringBuffer response = new StringBuffer(); while ((inputLine = br.readLine()) != null) { response.append(inputLine); } br.close(); //response.data 텍스트만 추출 JSONParser jsonParser= new JSONParser(); JSONObject jsonObject = (JSONObject) jsonParser.parse(response.toString()); JSONArray imageInfoArray = (JSONArray) jsonObject.get("images"); response = new StringBuffer(); for(int i=0; i<imageInfoArray.size(); i++) { JSONObject fields = (JSONObject) imageInfoArray.get(i); JSONArray fieldInfoArray = (JSONArray) fields.get("fields"); String temp = ""; for(int j=0; j<fieldInfoArray.size(); j++) { JSONObject fieldObject = (JSONObject) fieldInfoArray.get(j); //temp+=(String)fieldObject.get("inferText"); response.append((String)fieldObject.get("inferText")); if((boolean)fieldObject.get("lineBreak")) { //result.add(temp); temp = ""; response.append(System.getProperty("line.separator")); } } result = response.toString(); } catch (Exception e) { Log.error(e.toString()); } finally { return result; }</pre>

CLOVA OCR은 이미지 파일을 받아서 이미지 안에 텍스트를 추출하여 반환합니다.

OCR API는 HTTP 기반의 REST API이며, 사용자 인증(로그인)이 필요하지 않은 비로그인 Open API입니다.

1. Front에서 인자 값을 받아온다
 2. 앱 클라이언트 아이디값과 비밀번호를 헤더에 담아 요청한다.
 3. Front에서 받아온 인자 값 File을 Image에 담아 API URL에 요청한다.
 4. 정상적으로 입력이 되어 요청이 됐을시 Json 형식으로 응답받는다.
 5. Json Data 중에서 텍스트만 추출한다.
- * Json Data 의 lineBreak 값으로 개행을 판별한다.
- * StringBuffer 에 담아 텍스트를 반환한다.
6. 오류 발생시 400, 500 코드오류를 출력한다.

STT API

```
String imgFile = path;
File voiceFile = new File(imgFile);
String language = lang; // 언어 코드 ( Kor, Jpn, Eng, Chn )
String apiUrl = "https://naveropenapi.apigw.ntruss.com/recog/v1/stt?lang=" + language
URL url = new URL(apiUrl);
URLConnection conn = (URLConnection) url.openConnection();
conn.setUseCaches(false);
conn.setDoOutput(true);
conn.setDoInput(true);
conn.setRequestProperty("Content-Type", "application/octet-stream");
conn.setRequestProperty("X-NCP-APIGW-API-KEY-ID", clientId);
conn.setRequestProperty("X-NCP-APIGW-API-KEY", clientSecret);
OutputStream outputStream = conn.getOutputStream();
FileInputStream inputStream = new FileInputStream(voiceFile);
byte[] buffer = new byte[4096];
int bytesRead = -1;
while ((bytesRead = inputStream.read(buffer)) != -1) {
    outputStream.write(buffer, 0, bytesRead);
}
outputStream.flush();
inputStream.close();
BufferedReader br = null;
int responseCode = conn.getResponseCode();
if (responseCode == 200) { // 정상 호출
    br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
} else { // 오류 발생
    System.out.println("error!!!!!! responseCode= " + responseCode);
    br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
}
String inputLine;
if (br != null) {
    StringBuffer response = new StringBuffer();
    while ((inputLine = br.readLine()) != null) {
        response.append(inputLine);
        result=response.toString();
    }
    br.close();
    System.out.println(result);
} else {
    System.out.println("error !!!");
}
} catch (Exception e) {
    System.out.println(e);
}
return result;
}
```

CLOVA STT은 비디오,오디오 파일을 받아서 파일의 음성을 텍스트로 추출하여 반환합니다.

STT API는 HTTP 기반의 REST API이며, 사용자 인증(로그인)이 필요하지 않은 비로그인 Open API입니다.

1. Front에서 인자 값을 받아온다
2. 앱 클라이언트 아이디값과 비밀번호를 헤더에 담아 요청합니다.
3. Front에서 받아온 인자 값 File ,Lang(언어)을 변수에 담아 API URL에 요청합니다.
4. 정상적으로 입력이 되어 요청이 됐을시 Json 형식으로 응답받는다.
5. 변수에 담긴 자료들을 토대로 임시파일을 생성 합니다.
5. 파일을 읽어 들이면서 StringBuffer에 Text를 계속 담는다.
- * StringBuffer 에 담아 텍스트를 반환한다. 후에 임시 파일을 삭제합니다.
6. 오류 발생시 400, 500 코드오류를 출력합니다.

PAPAGO API

```
private String post(String apiUrl, Map<String, String> requestHeaders, String text, String from_language, String to_language) {  
  
    HttpURLConnection con = connect(apiUrl);  
    String postParams = "source="+from_language+"&target="+to_language+"&text=" + text; //일본어: 한국어 (ko) -> 목적어: 영어 (en)  
  
    try {  
        con.setRequestMethod("POST");  
        for(Map.Entry<String, String> header : requestHeaders.entrySet()) {  
            con.setRequestProperty(header.getKey(), header.getValue());  
        }  
        con.setDoOutput(true);  
        try (DataOutputStream wr = new DataOutputStream(con.getOutputStream())) {  
            wr.write(postParams.getBytes());  
            wr.flush();  
        }  
  
        int responseCode = con.getResponseCode();  
        if (responseCode == HttpURLConnection.HTTP_OK) { // 정상 응답  
            return readBody(con.getInputStream());  
        } else { // 에러 응답  
            return readBody(con.getErrorStream());  
        }  
    } catch (IOException e) {  
        throw new RuntimeException("API 요청과 응답 실패", e);  
    } finally {  
        con.disconnect();  
    }  
}
```

PAPAGO API는 텍스트를 받아서 텍스트를 추출하여 반환합니다.

PAPAGO API는 HTTP 기반의 REST API이며, 사용자 인증(로그인)이 필요하지 않은 비로그인 Open API입니다.

1. Front에서 인자 값을 받아옵니다.
2. 앱 클라이언트 아이디값과 비밀번호를 헤더에 담아 요청합니다.
3. Front에서 받은 인자 값 텍스트, 번역할 언어, 번역될 언어를 변수에 담아 API URL에 요청합니다.
4. 정상적으로 입력이 되어 요청이 됐을시 Json 형식으로 응답 받습니다.
5. 정상적으로 입력이 되어 요청이 됐을시 지정경로에 파일생성이 됩니다.
6. 오류 발생시 400,500 코드오류를 출력합니다.

TTS API

```

1 package com.bcpr.backend.TTS.helper;
2
3 //네이버 음성합성 Open API 예제
4 import java.io.BufferedReader;
5
6
7
8
9
10
11
12
13
14
15
16
17
18 @Slf4j
19 public class ttsHelper {
20
21     public String getTTSHelper(String tts, String voice, String speed, String volume, String path) { //Front에서 인자 값을 받아옴
22         System.out.println(tts);
23         System.out.println(voice);
24         System.out.println(speed);
25         System.out.println(volume);
26         //요청 헤더
27         String clientId = "zvvh4atvgh"; //애플리케이션 클라이언트 아이디값;
28         String clientSecret = "MWIUUGirdLgCIE4ASgRrklRn4XdQFwFwmceH9vN"; //애플리케이션 클라이언트 시크릿값;
29         try {
30             String text = URLEncoder.encode(tts, "UTF-8"); //변환할 TEXT값을 tts에 담는다
31             String apiURL = "https://naveropenapi.apigw.ntruss.com/tts-premium/v1/tts"; //요청 API URL
32             URL url = new URL(apiURL);
33             HttpURLConnection con = (HttpURLConnection)url.openConnection();
34             con.setRequestMethod("POST");
35             con.setRequestProperty("X-NCP-APIGW-API-KEY-ID", clientId);
36             con.setRequestProperty("X-NCP-APIGW-API-KEY", clientSecret);
37             // post request
38             String postParams = "speaker="+voice+"&volume="+volume+"&speed="+speed+"&pitch=0&format=mp3&text=" + text;
39             con.setDoOutput(true);
40             DataOutputStream wr = new DataOutputStream(con.getOutputStream());
41             wr.writeBytes(postParams);
42             wr.flush();
43             wr.close();
44             int responseCode = con.getResponseCode();
45             BufferedReader br;
46             if(responseCode==200) { // 정상 호출
47                 InputStream is = con.getInputStream();
48                 int read = 0;

```

CLOVA TTS는 음성으로 변환할 텍스트를 입력받은 후 인자값으로 지정된 음색과 속도로 음성을 합성하여

그 결과를 반환합니다.

TTS API는 HTTP 기반의 REST API이며, 사용자 인증(로그인)이 필요하지 않은 비로그인 Open API입니다.

1. Front에서 인자 값을 받아온다
 2. 앱 클라이언트 아이디값과 비밀번호를 헤더에 담아 요청한다.
 3. Front에서 받아온 인자 값 텍스트, 성우(voice), 볼륨(volume), 속도(speed)를 하나의 변수에 담아 API URL에 요청한다.
 4. 파일생성시 저장할 경로와 파일명, 확장자를 설정한다.
 5. 정상적으로 입력이 되어 요청이 됐을시 지정경로에 파일생성이 되어진다.
- * 사용자마다 경로가 달라 상대경로로 설정
 - * 절대경로 : C:\study_boot\BCPR3\DMTT\backend\src\main\webapp\resources
6. 오류 발생시 400, 500 코드오류를 출력한다.

Spring Security Config

```
@Override
protected void configure(HttpSecurity http) throws Exception{
    CustomAuthenticationFilter customAuthenticationFilter = new CustomAuthenticationFilter(authenticationManagerBean());
    customAuthenticationFilter.setFilterProcessesUrl("/api/login");

    http.httpBasic().disable();
    http.cors().configurationSource(corsConfigurationSource());
    http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    http.csrf().disable();
    http.sessionManagement().sessionCreationPolicy(STATELESS);
    http.authorizeRequests().antMatchers("/").permitAll();

    //OCR
    http.authorizeRequests().antMatchers("/api/ocr").permitAll();
    http.authorizeRequests().antMatchers("/api/ocr/**").permitAll();
    http.authorizeRequests().antMatchers("/api/ocr/download/**").hasAnyAuthority("ROLE_USER");

    //TTS
    http.authorizeRequests().antMatchers("/api/tts").permitAll();
    http.authorizeRequests().antMatchers("/api/tts/**").permitAll();
    http.authorizeRequests().antMatchers("/api/tts/server").permitAll();
    http.authorizeRequests().antMatchers("/api/tts/download/**").hasAnyAuthority("ROLE_USER");

    //STT
    http.authorizeRequests().antMatchers("/api/Stt").permitAll();
    http.authorizeRequests().antMatchers("/api/Stt/**").permitAll();
    http.authorizeRequests().antMatchers("/api/Stt/download/**").hasAnyAuthority("ROLE_USER");

    //PAPAGO
    http.authorizeRequests().antMatchers("/api/papago").permitAll();
    http.authorizeRequests().antMatchers("/api/papago/**").permitAll();
    http.authorizeRequests().antMatchers("/resources/**").permitAll();

    //USER
    http.authorizeRequests().antMatchers("/api/login").permitAll();
    http.authorizeRequests().antMatchers("/api/login/**", "/api/token/refresh/**").permitAll();
    http.authorizeRequests().antMatchers(GET, "/api/users").hasAnyAuthority("ROLE_USER");
    http.authorizeRequests().antMatchers(GET, "/api/user/**").permitAll();
    http.authorizeRequests().antMatchers(POST, "/api/role/**").hasAnyAuthority("ROLE_ADMIN");
    http.authorizeRequests().antMatchers(POST, "/api/user/save/normal").permitAll();
    http.authorizeRequests().anyRequest().authenticated();
    http.addFilter(customAuthenticationFilter);
    http.addFilterBefore(new CustomAuthorizationFilter(), UsernamePasswordAuthenticationFilter.class);
}

// CORS 허용 적용
@Bean
public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration configuration = new CorsConfiguration();

    //configuration.addAllowedOrigin("*");
    configuration.addAllowedOriginPattern("*");
    configuration.addAllowedHeader("*");
    configuration.addAllowedMethod("*");
    configuration.setAllowCredentials(true);
    configuration.addExposedHeader("content-disposition");

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", configuration);
    return source;
}
```

Spring Security를 사용하여 URL 접근 권한과 CORS 및 필터를 설정합니다.

1. 기본 로그인 창 사용 안함, CORS 적용, 세션 정책 STATELESS, CSRF 보호 기능 활성화를 설정합니다.
2. antMatchers("URL").hasAnyAuthority("권한명") 으로 권한제한 또는 antMatchers("URL").permitAll() 으

로 권한해제 합니다.

3. 인증 필터 CustomAuthenticationFilter 를 추가 합니다.

4. 아이디, 패스워드 인증 전에 권한 필터 CustomAuthoricationFilter 를 추가 적용합니다.

Spring Security 인증 Filter

```
public class CustomAuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    private final AuthenticationManager authenticationManager;

    public CustomAuthenticationFilter(AuthenticationManager authenticationManager){
        this.authenticationManager = authenticationManager;
    }

    @Override
    public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response)
        throws AuthenticationException {
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        Log.info("email is: {}",email);
        Log.info("Password is: {}",password);
        UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(email, password);
        return authenticationManager.authenticate(authenticationToken);
    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain, Authentication
        throws IOException, ServletException {
        User user = (User) authentication.getPrincipal();
        Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());
        String access_token = JWT.create()
            .withSubject(user.getUsername())
            .withExpiresAt(new Date(System.currentTimeMillis()+10*60*1000))
            .withIssuer(request.getRequestURI().toString())
            .withClaim("roles",user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()))
            .sign(algorithm);
        String refresh_token = JWT.create()
            .withSubject(user.getUsername())
            .withExpiresAt(new Date(System.currentTimeMillis()+30*60*1000))
            .withIssuer(request.getRequestURI().toString())
            .withClaim("roles",user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()))
            .sign(algorithm);
        Map<String, String> tokens = new HashMap<>();
        tokens.put("access_token",access_token);
        tokens.put("refresh_token",refresh_token);
        Log.info("test access : {}",access_token);
        Log.info("test refresh : {}",refresh_token);
        response.setContentType(APPLICATION_JSON_VALUE);
        new ObjectMapper().writeValue(response.getOutputStream(),tokens);
    }
}
```

JWT 토큰 발급을 위해서 Spring Security의 인증 필터를 재정의 합니다.

1. email과 password를 통해 DB에 등록 되어있는 유저인지 확인하고 인증 결과를 반환합니다.

2. 인증 결과가 참인 경우 User 정보를 토대로 JWT 토큰을 생성하여 발급합니다.

* 유저 아이디, 만료기간, 요청 URL, 권한정보, 암호화 알고리즘 정보를 담습니다.

* access_token 과 refresh_token을 같이 발급하여 refresh_token의 만료기간이 더 길게 설정합니다.(재 인증 시 사용)

Spring Security 권한 Filter

```
public class CustomAuthorizationFilter extends OncePerRequestFilter {
    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {

        if(request.getServletPath().equals("/api/login") ||
            request.getServletPath().equals("/api/token/refresh/**")){

            filterChain.doFilter(request, response);

        }else{
            String authorizationHeader = request.getHeader(AUTHORIZATION);
            if(authorizationHeader != null && authorizationHeader.startsWith("Bearer ")){
                try {
                    String token = authorizationHeader.substring("Bearer ".length());
                    Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());

                    JWTVerifier verifier = JWT.require(algorithm).build();
                    DecodedJWT decodedJWT = verifier.verify(token);
                    String username = decodedJWT.getSubject();
                    String[] roles = decodedJWT.getClaim("roles").asArray(String.class);
                    Collection<SimpleGrantedAuthority> authorities = new ArrayList<>();

                    stream(roles).forEach(role -> {
                        if(role!=null)
                            authorities.add(new SimpleGrantedAuthority(role));
                    });

                    UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
                        username,
                        null,
                        authorities
                    );
                    SecurityContextHolder.getContext().setAuthentication(authenticationToken);
                    filterChain.doFilter(request,response);
                }catch(Exception exception){
                    log.error("Error logging in: {}", exception.getMessage());
                    response.setHeader("error", exception.getMessage());
                    response.setStatus(FORBIDDEN.value());
                    //response.sendError(FORBIDDEN.value());
                    Map<String, String> error = new HashMap<>();
                    error.put("error_message",exception.getMessage());
                    response.setContentType(APPLICATION_JSON_VALUE);
                    new ObjectMapper().writeValue(response.getOutputStream(),error);
                }
            }else{
                filterChain.doFilter(request,response);
            }
        }
    }
}
```

JWT 토큰으로 인가된 사용자를 판별하기 위해서 Spring Security의 권한 필터를 재정의 합니다.

1. URL이 "/api/login" 또는 "/api/token/refresh/**" 인 경우 기존 권한 필터를 사용합니다.
2. 그 외의 인가가 필요한 URL 인 경우 request Header의 Authorization 을 확인하여 인가합니다.

* JWT 토큰을 받아서 해독 후 권한을 비교하여 인가를 판별합니다.

* 권한 잘못된 경우 403 오류를 반환합니다.


```
@Entity
@Data
@NoArgsConstructor @AllArgsConstructor
@Table(uniqueConstraints = {@UniqueConstraint(columnNames = {"EMAIL"})})
public class User {
    @Id @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String email;
    private String password;
    private String profile;

    @ManyToMany(fetch = FetchType.EAGER)
    private Collection<Role> roles = new ArrayList<>();
}
```

JPA 를 활용하여 테이블을 자동 생성 합니다.

@Entity, @Table 객체와 테이블 매핑

@Id 기본 키 매핑

@ManyToMany 연관관계 매핑

Controller Download 파일 전송

```
@GetMapping("/ocr/download/{email}/{media_no}/{kind}")
public void download(
    @PathVariable("email") String email,
    @PathVariable("media_no") int media_no,
    @PathVariable("kind") String kind,
    HttpServletRequest request,
    HttpServletResponse response) throws Exception {

    Media_Trans mt = mapper.getMedia_Trans(email, media_no);

    String osName = System.getProperty("os.name").toLowerCase();

    FileSaveHelper fsh;
    if(osName.contains("win")) {
        fsh = new FileSaveHelper(request.getServletContext().getRealPath("resources"));
    }else {
        fsh = new FileSaveHelper();
    }

    String path = fsh.makePath("media_trans", email, mt.getInput(), mt.getTrans_date(), kind);

    File file = new File(path);
    if(!kind.equals("input")) {
        fsh.saveFile(path, file, mt.getOutput());
    }

    byte[] fileByte = FileUtils.readFileToByteArray(file);

    response.setContentType("application/octet-stream");
    response.setHeader("Content-Disposition", "attachment; fileName=\"" + URLEncoder.encode(file.getName(), "UTF-8")+"\"");
    response.setHeader("Content-Transfer-Encoding", "binary");

    response.getOutputStream().write(fileByte);
    response.getOutputStream().flush();
    response.getOutputStream().close();

    if(kind.equals("output")){
        fsh.deleteFile(file);
    }
}
```

Front 단으로 파일을 전송합니다.

1. response Header 를 파일형식으로 설정한다
2. File 을 이진 파일로 변환하여 반환한다.

로딩 화면 구현

```
<div class="layerPopup" v-show="isLoading">
  <div class="spinner"></div>
</div>
```

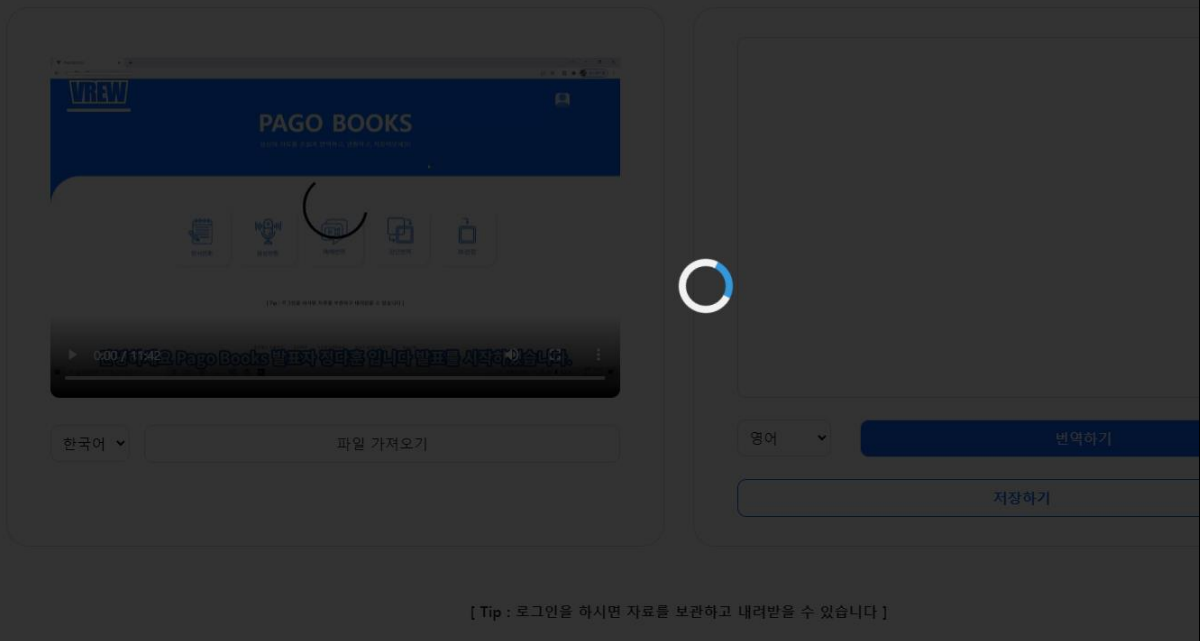
```
isLoading() {
  return this.$store.state.isLoad;
},
```

```
state: {
  access_token: '',
  refresh_token: '',
  userInfo: null,
  isLogin: false,
  isLoginError: false,
  isLoad: false,
},
mutations: {
  onLoad(state) {
    state.isLoad = true;
  },
  offLoad(state) {
    state.isLoad = false;
  },
},
```

```
.layerPopup {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.8);
  z-index: 1000;
  justify-content: center;
  align-items: center;
  margin: 0 0 0 0;
}

.spinner {
  position: absolute;
  top: 50%;
  left: 50%;
  border: 8px solid #f3f3f3; /* Light gray */
  border-top: 8px solid #3498db; /* Blue */
  border-radius: 50%;
  width: 60px;
  height: 60px;
  animation: spinner 2s linear infinite;
}

@keyframes spinner {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}
```



데이터가 처리 중이라는 것을 유저에게 알리고 유저의 불필요한 이벤트를 막기 위해서 로딩 화면을 구현 하였습니다.

1. Vuex 의 state 를 사용하여 Loading 상태를 computed 에 isLoading을 통해 데이터의 변화를 캐치

합니다.

2. Axios 전 후로 onLoad, offLoad 를 호출 하여 isLoad의 값을 변화시킵니다.
3. v-show 속성으로 디스플레이 값을 변경합니다.
4. @keyframes 을 이용하여 애니메이션 효과를 줍니다.

```

Axios interceptors

axios.interceptors.request.use(
  function (config) {
    if (store.state.isLogin && Object.prototype.hasOwnProperty.call(localStorage, "access_token")) {
      config.headers.Authorization = 'Bearer ' + localStorage.getItem("access_token");
      if (localStorage.getItem("access_token") === "") {
        config.headers.Authorization = 'Bearer ' + localStorage.getItem("refresh_token");
      }
    }
    return config;
  },
  function (error) {
    return Promise.reject(error);
  }
);

// Add a response interceptor
axios.interceptors.response.use(
  function (response) {
    return response;
  },
  async function (error) {
    try {
      const errorAPI = error.response.config;
      console.log("response ERROR!!! " + error.response.status + ", " + errorAPI.retry);
      if ((error.response.status === 403 && errorAPI.retry === undefined) && localStorage.getItem('refresh_token')) {
        errorAPI.retry = true;
        store.dispatch('getRefreshToken');
        return await axios(errorAPI);
      }
    } catch (err) {
      console.error('[axios.interceptors.response] error : ', err.message);
    }
    return Promise.reject(error);
  }
);
```

Axios 통신에 JWT 토큰을 교환하기 위해서 interceptors 를 사용하였습니다.

interceptors.request 는 request 전에, interceptors.response 는 response 직후에 호출 됩니다.

1. interceptors.request 는 요청 전 JWT 토큰을 담습니다.
2. interceptors.response 는 응답 직후 JWT 토큰 기간 만료 오류 시 토큰 재발급 요청을 보냅니다.

SNS 로그인을 위한 Index.html 설정

```
<script src="https://apis.google.com/js/platform.js" async defer></script>
<meta name="google-signin-client_id"
  content='[REDACTED]'>
<script src="https://developers.kakao.com/sdk/js/kakao.js"></script>
<script>
  window.Kakao.init("[REDACTED]");
</script>
```

SNS 로그인 API 를 사용하기 위해 index.html에 API와 개발키를 등록합니다.

SNS 로그인 함수 구현

```

kakaoLogin() {
  window.Kakao.Auth.login({
    scope: "profile_image, account_email",
    success: this.kakaoInfo,
  });
  this.$emit("closeModal");
},
async kakaoInfo(authObj) {
  console.log(authObj);
  const userInfo = {
    email: null,
    profile: null,
  };
  await window.Kakao.API.request({
    url: "/v2/user/me",
    success: (res) => {
      const kakao_account = res.kakao_account;
      userInfo.email = kakao_account.email;
      userInfo.profile = kakao_account.profile.thumbnail_image_url;
    }
  });
  //구글 버튼
  googleLogin() {
    var self = this;
    window.gapi.signin2.render("my-signin2", {
      scope: "profile email",
      width: 240,
      height: 50,
      longtitle: true,
      theme: "dark",
      onsuccess: this.googleInfo,
      onfailure: this.googleLogout,
    });
    setTimeout(function () {
      if (!self.googleLoginCheck) {
        const auth = window.gapi.auth2.getAuthInstance();
        auth.isSignedIn.get();
        const btn = document.querySelector(".abcRioButton");
        if (btn != null) btn.click();
      }
    }, 100);
  },
  //구글 로그인 이후 실행되는 콜백함수(성공)
  async googleInfo(googleUser) {
    //const user_join_type = "g"
    const profile = googleUser.getBasicProfile();
    const googleEmail = profile.getEmail();
    const googleProfile = profile.getImageUrl();
  }
}

```

카카오는 window.Kakao.Auth.login 을

구글은 window.gapi.signin2.render 를 재정의 하여 구현하였습니다.

Success 란에 로그인 성공 시 호출되는 함수를 선언하고 정의해주시면 됩니다.

파일 다운로드 함수 구현

```
download() {  
  let str = "/api/tts/download/" + this.email + "/" + this.voice_no + "/";  
  if (this.showInput) str += "input";  
  else str += "output";  
  
  axios  
    .get(str, {  
      responseType: "blob",  
    })  
    .then((res) => {  
      const name = res.headers["content-disposition"]  
        .split("fileName=")[1]  
        .replace("/"/g, "");  
      const url = window.URL.createObjectURL(new Blob([res.data]));  
      const link = document.createElement("a");  
      link.href = url;  
      link.setAttribute("download", name);  
      document.body.appendChild(link);  
      link.click();  
      link.remove();  
      console.log("다운로드 성공");  
    })  
    .catch((err) => {  
      err;  
      console.log("다운로드 실패");  
    });  
},
```

파일 다운로드 함수 입니다.

파일 형식을 받기 위해서 responseType: "blob" 을 헤더에 적어줍니다.

받은 파일을 바로 유저가 다운로드 받을 수 있도록 동적으로 <a> 태그를 생성하여 클릭 이벤트를 자동발생시킵니다.

4.8 배포환경



**NAVER
CLOUD
PLATFORM**







호스팅 서버
Naver Cloud Micro

운영체제
CentOS 7.8

설치된 소프트웨어
MySQL 8.0 community-server
Java 11.0.14.1
Nginx 1.20.2

도메인 주소
<http://www.pagobooks.shop>

유튜브 주소
<https://youtu.be/ccRG8pZ7ZCU>

GitHub 주소
<https://github.com/dahun3013/BCPR3>

4.9 시연

<http://www.pagobooks.shop>

4.9.1 깃 주소

이름	주소
정다훈	https://github.com/dahun3013
강하종	https://github.com/monoha724
장지원	https://github.com/JangG1
이동곤	https://github.com/LeeDongGon
신혜지	https://github.com/quiet-space
김희태	https://github.com/KimHuiTae

4.9.2 시연 동영상

<https://youtu.be/ccRG8pZ7ZCU>