



ocrstagram

Team 3

CONTENTS

01 팀원 소개

02 개발 환경

03 구현 내용

04 협업 내용

05 향후 계획

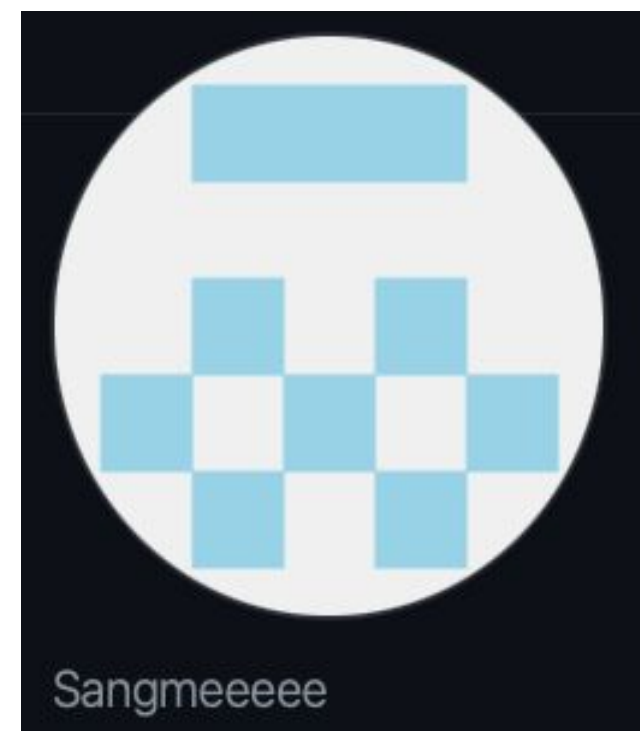
01

팀원 정보

■ 팀원 정보

ocrstagram

팀장



이상민

프론트엔드



김도형

백엔드



안상준

백엔드



손승우

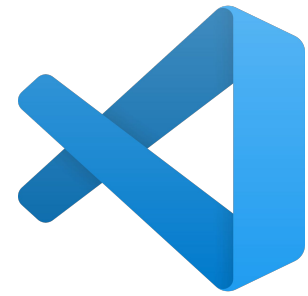
프론트엔드

02

개발환경

■ 개발환경

ocrstagram



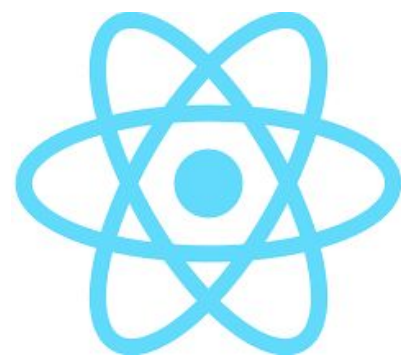
VS Code

- Node.js 개발에 사용하는 대표적인 에디터
- 서버 개발에 편의성을 위해 사용



Node.js

- 서버단 로직을 처리할 수 있는 Node.js 선택
- 다양한 모듈을 제공하여 개발 속도와 효율성이 매우 높다



React.js

- 사용자와 상호작용할 수 있는 UI를 손쉽게 만들 수 있는 라이브러리
- 다양한 프레임워크나 라이브러리와 혼용하여 사용하기 쉽다



Amazon
EC2

EC2

- AWS EC2를 사용하여 본 프로젝트 ocrstagram의 호스팅을 하였다

03

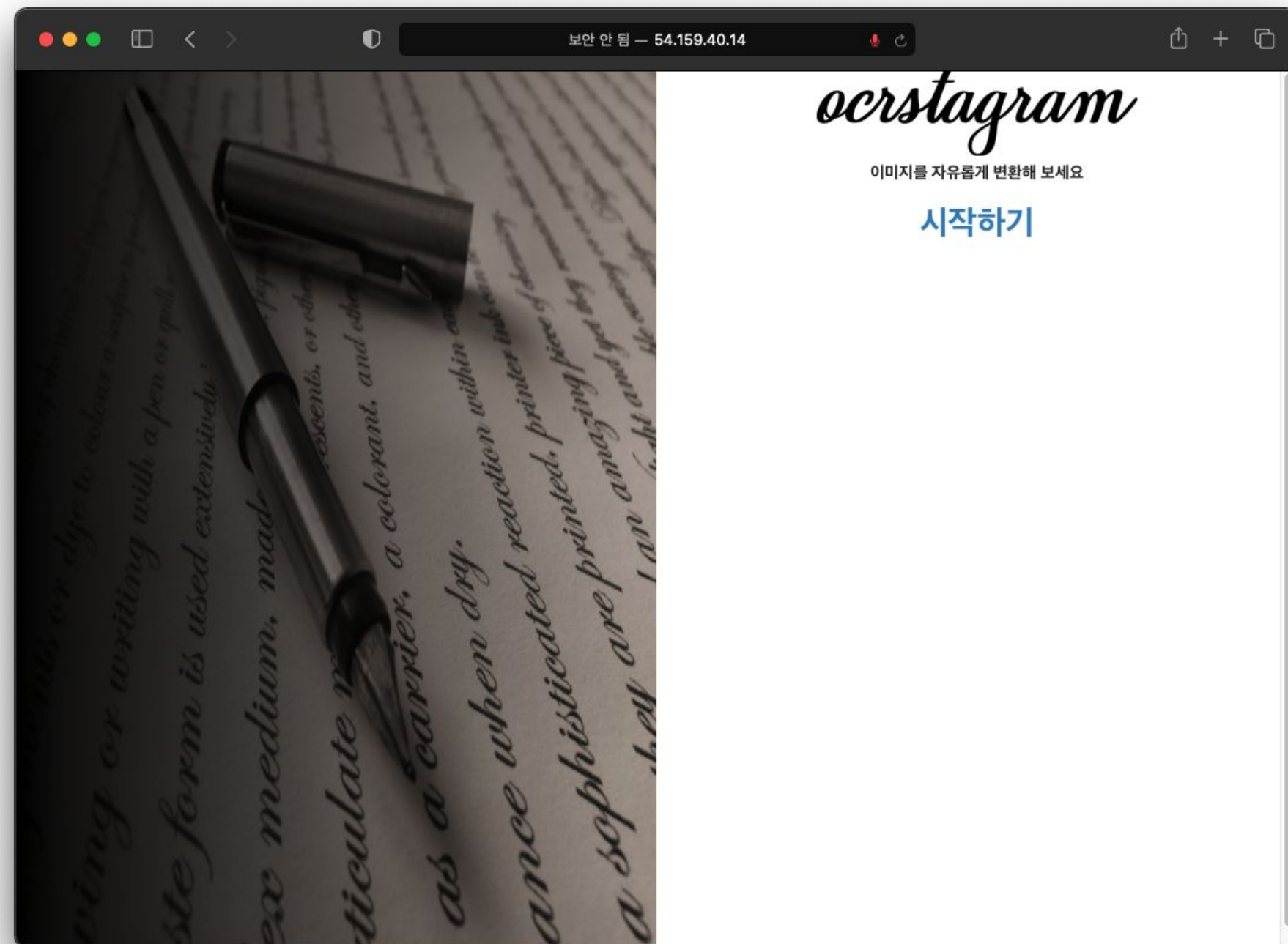
구현내용

- 로그인
 - 회원가입
 - 사용자 화면
 - 포스팅
 - OCR
 - 번역
 - DB
-

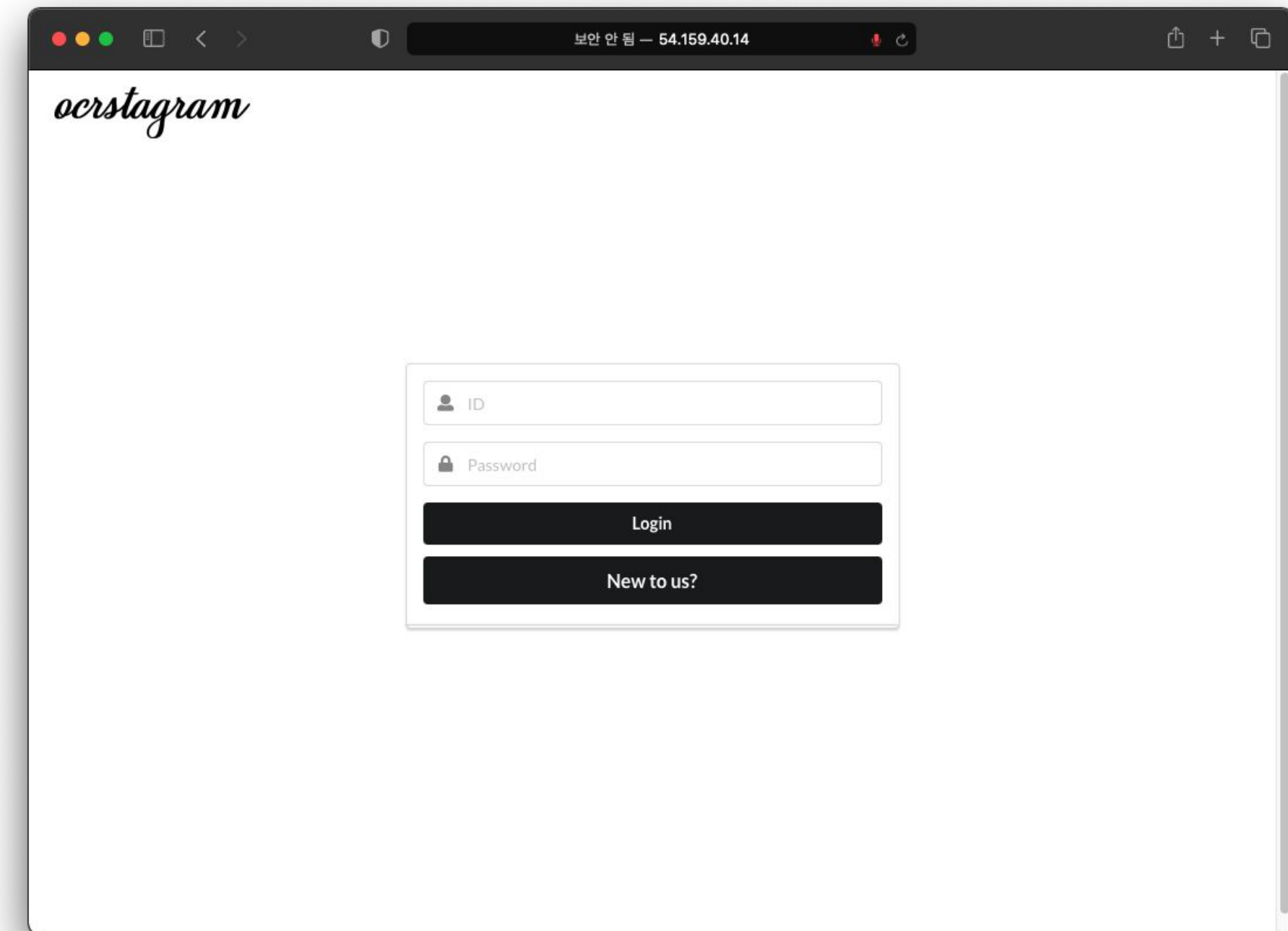
■ 로그인

ocrstagram

메인 화면



로그인 화면

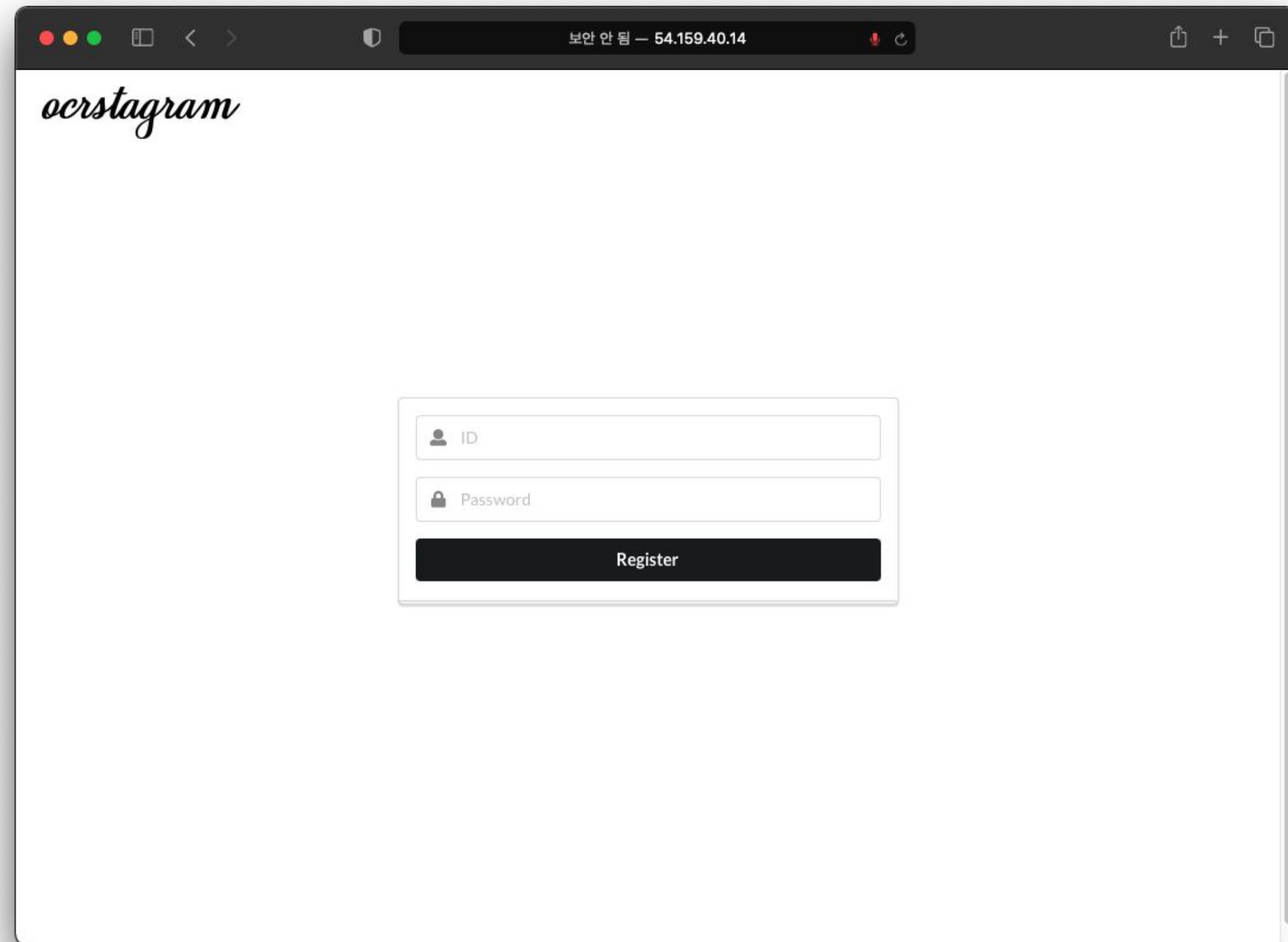


- 프론트의 라우팅처리를 통한 렌더링
- MVC방식을 사용 , Post요청에 대한 Controller로 처리

■ 회원가입

ocrstagram

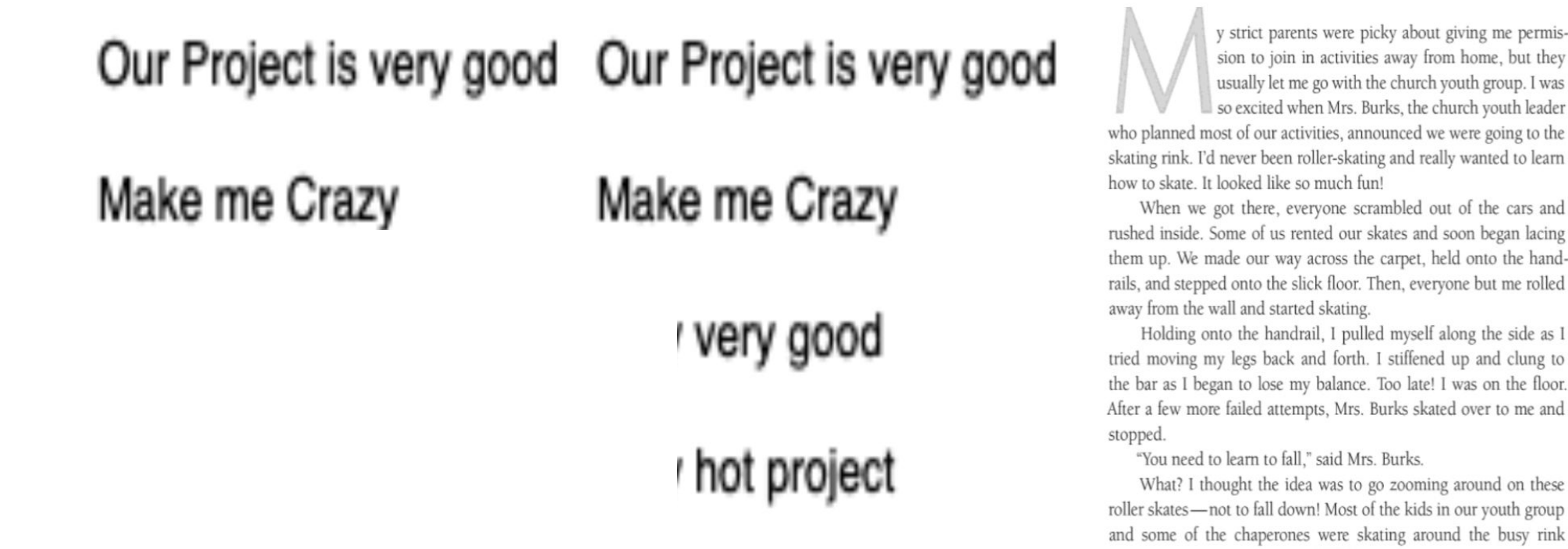
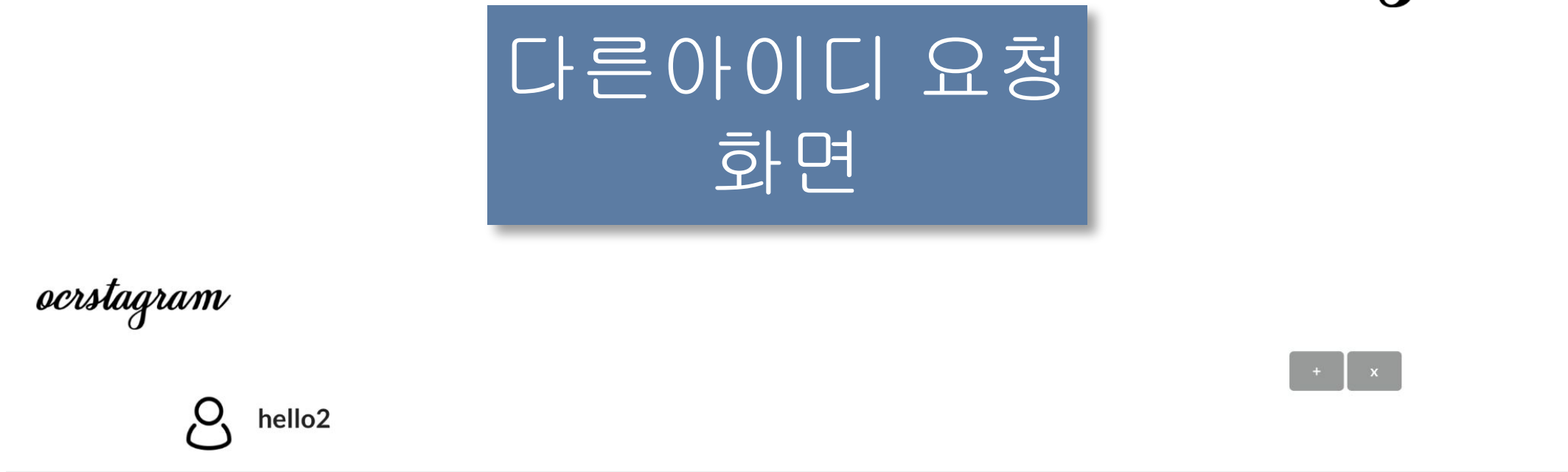
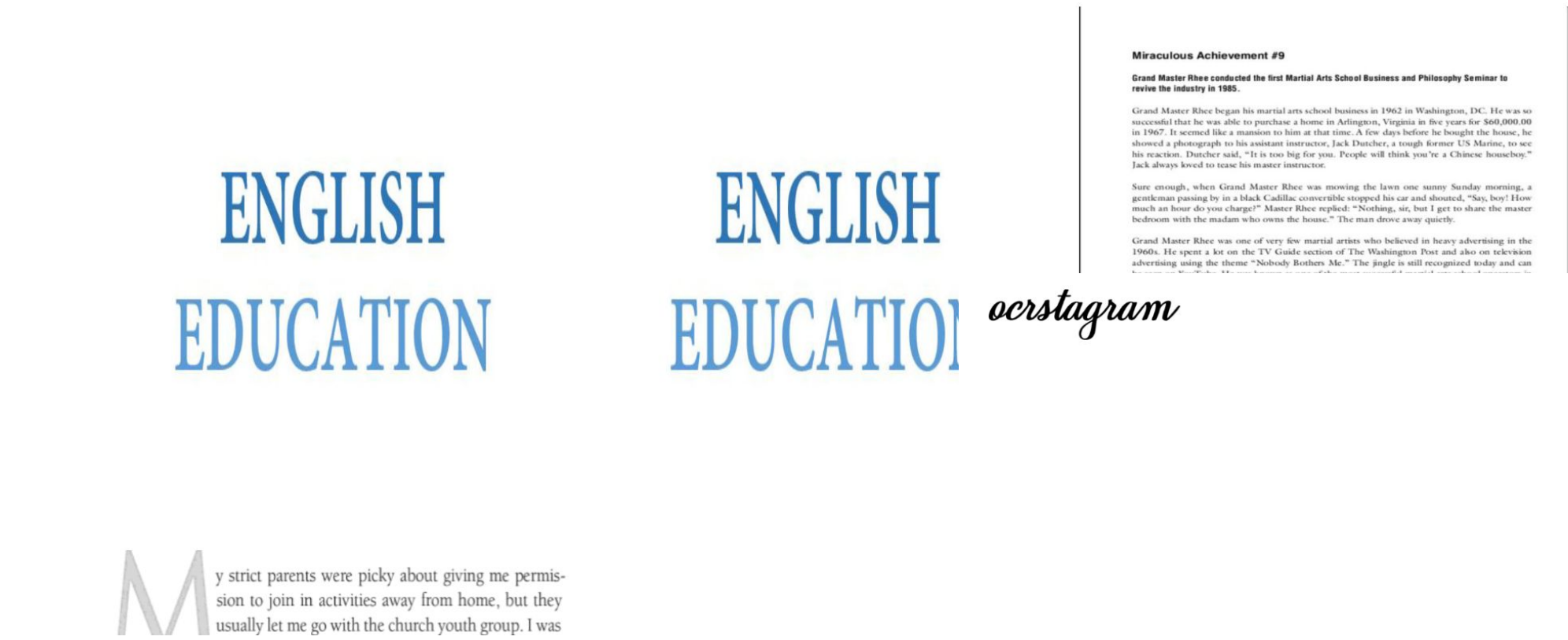
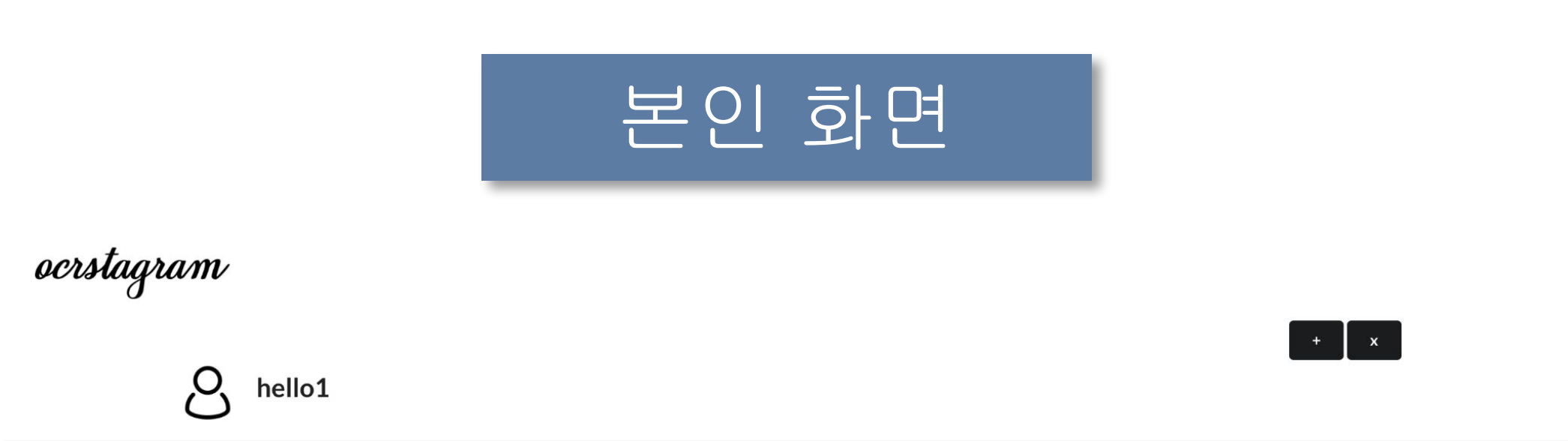
회원가입 화면



The screenshot shows a web browser window with the URL "보안 안 됨 - 54.159.40.14". The page displays the "ocrstagram" logo in the top left corner. In the center, there is a registration form with two input fields: "ID" (with a user icon) and "Password" (with a lock icon). Below these fields is a black "Register" button.

- Post방식으로 ID와 Password를 전송
- bcrypt방식으로 Password의 암호화

■ 사용자 화면

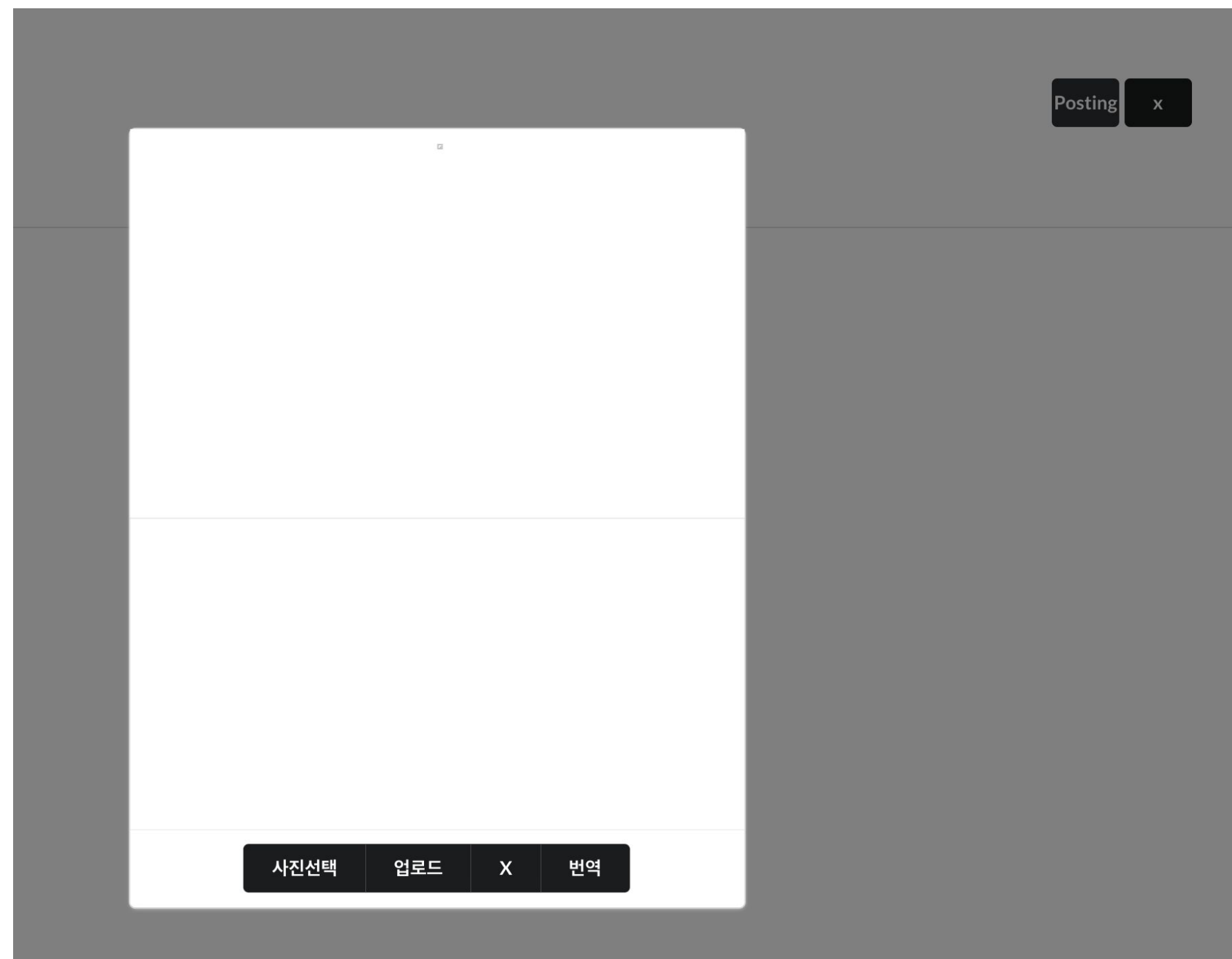


없는 아이디

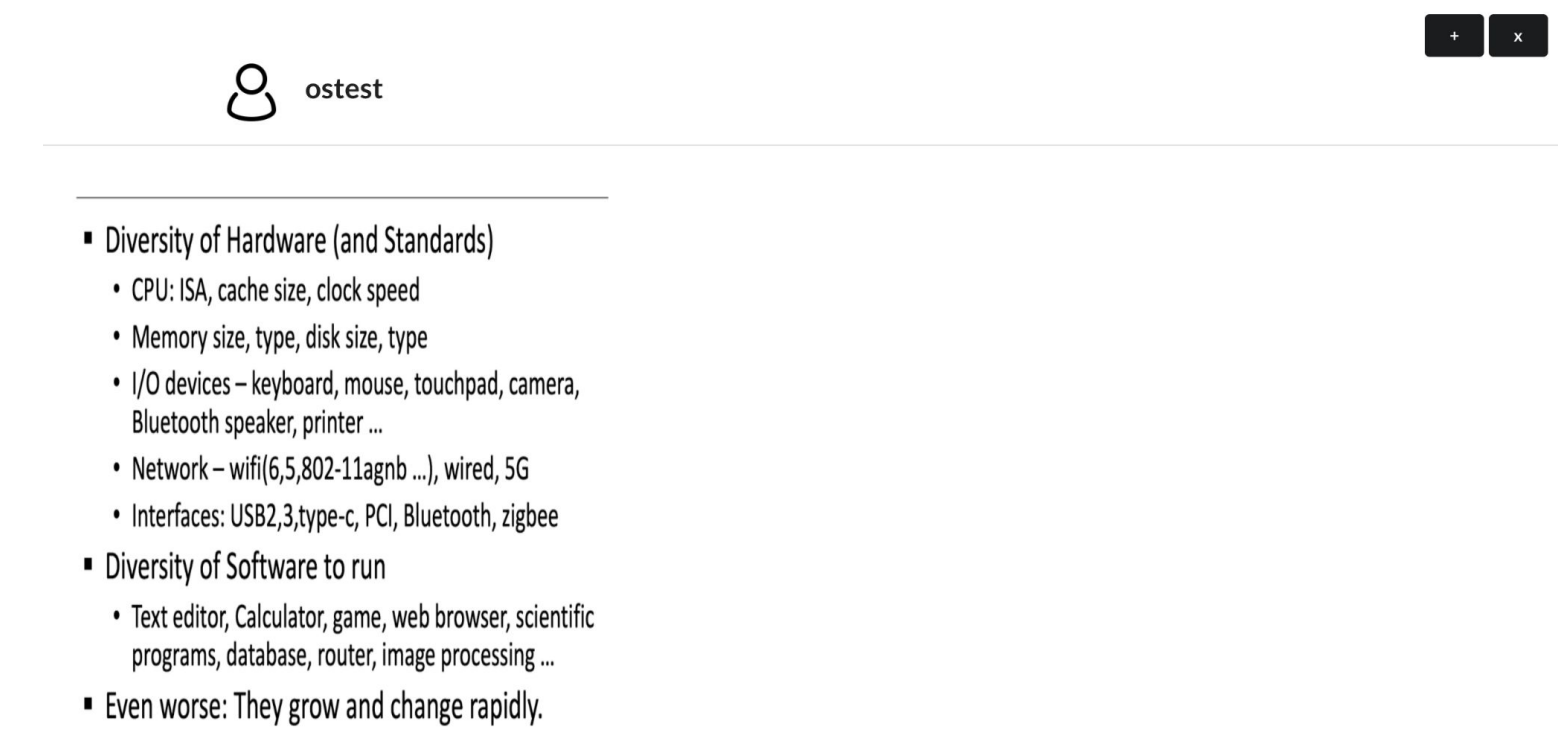
■ 포스팅

ocrstagram

포스팅전



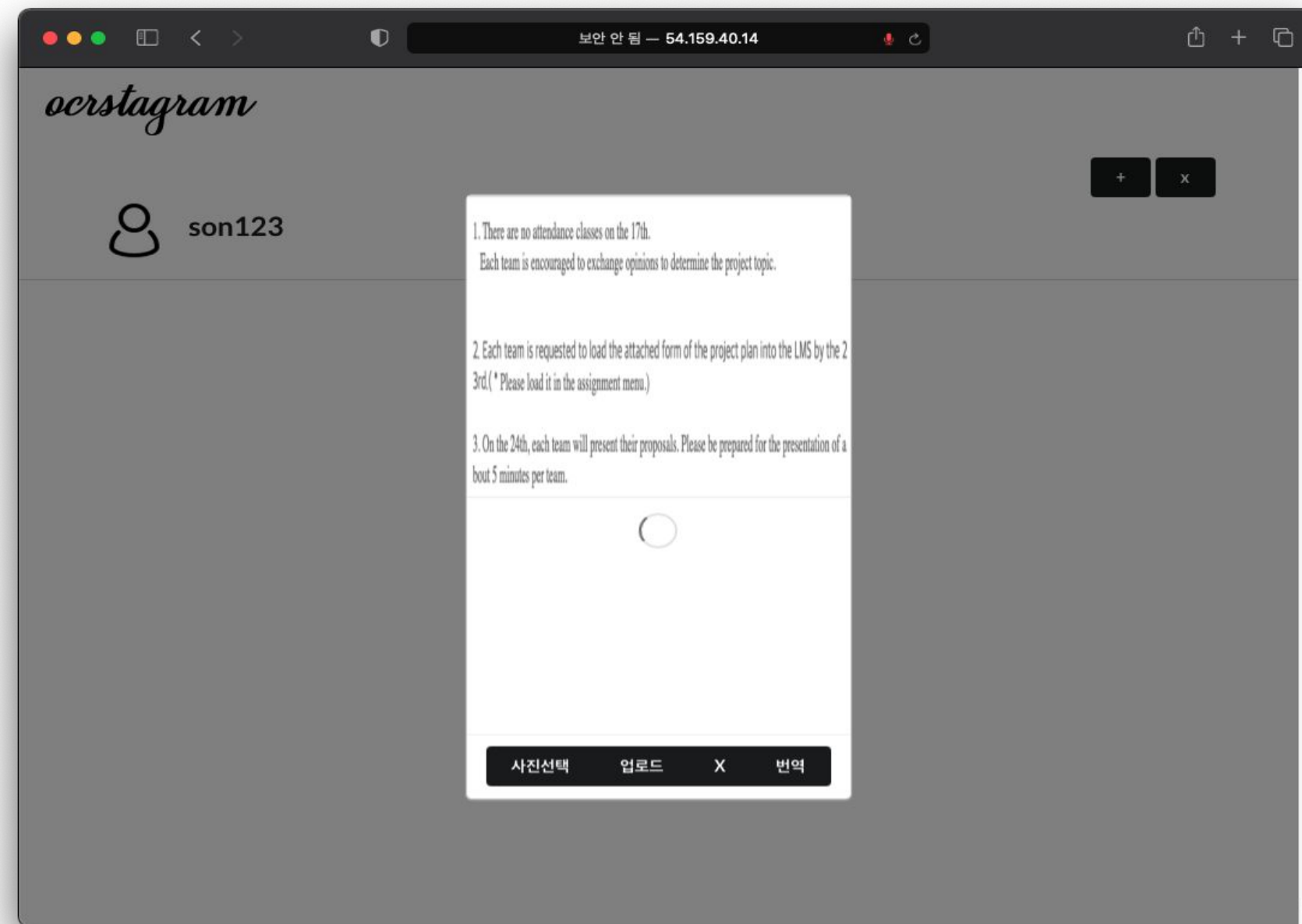
포스팅후



■ OCR

ocrstagram

OCR 진행 중



- tesseract.js를 활용하여 텍스트를 추출
- recognize 함수를 통해 추출한 데이터를 서버에 전송

OCR 완료

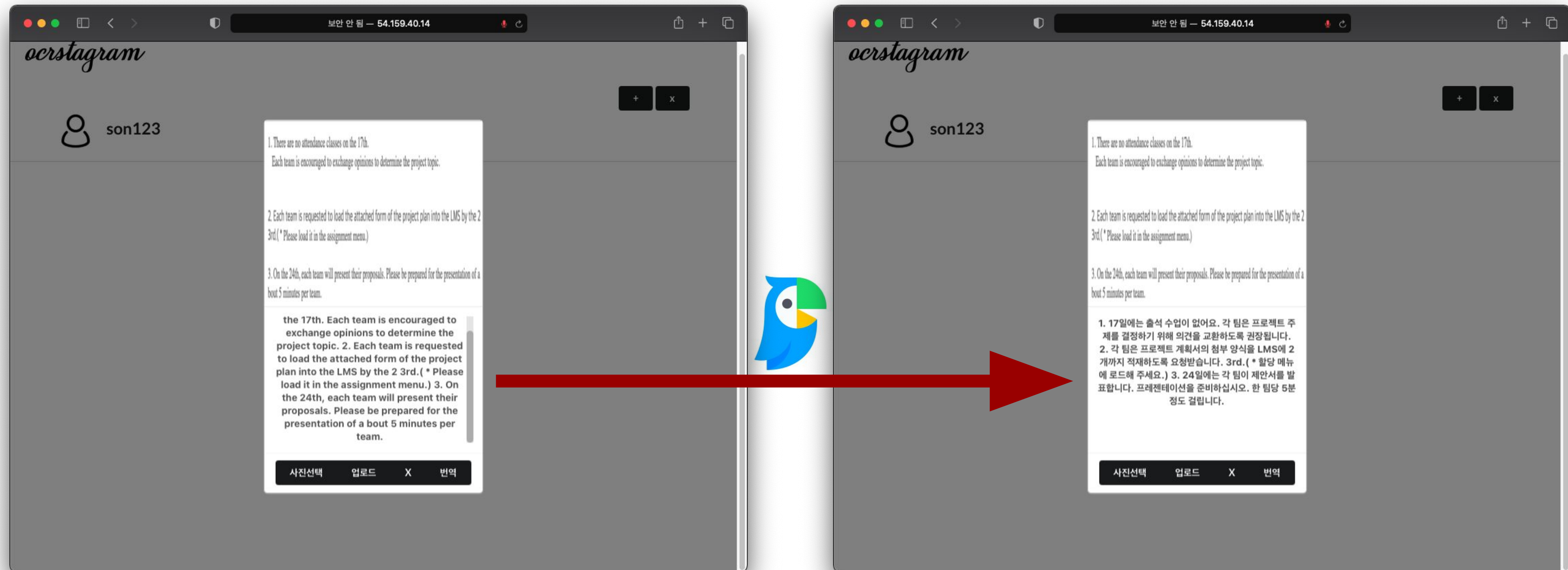


- 비동기 처리를 위해 express-async-handler를 사용
- 서버를 통해 받아온 결과를 화면에 출력

■ 번역

ocrstagram

Papago를 통한 번역 기능



- 파파고 API를 사용하여 번역 실행
- 파파고 API를 호출하기 위한 url을 선언
- post 요청을 보내기 위해 request 라이브러리 사용

DB

ocrstagram

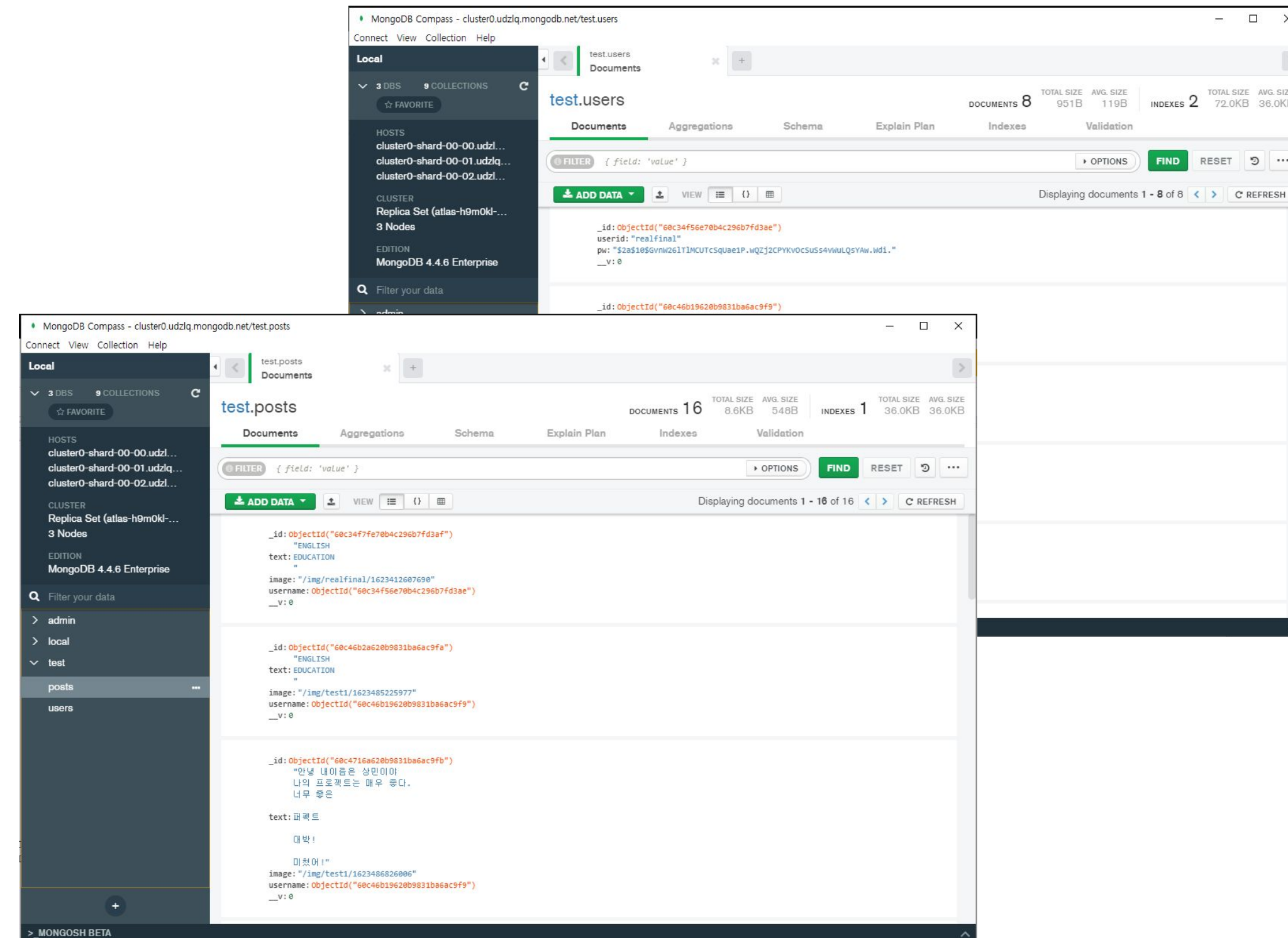
```
const mongoose = require('mongoose')
const Schema = mongoose.Schema;
const bcrypt = require('bcryptjs')

const UserSchema = new Schema({
  userid: { // pass in config object. and put in validation rules
    type: String,
    required: true,
    unique: true
  },
  pw: {
    type: String,
  }
});

// note: no lambda func! (no arrow func)
UserSchema.pre('save', function(next) {
  const user = this
  bcrypt.hash(user.pw, 10, function(err, hash) {
    user.pw = hash
    next()
  });
});

const BlogPost = mongoose.model('Post', PostSchema);
module.exports = BlogPost

const User = mongoose.model('User', UserSchema);
module.exports = User
```



DB 구성

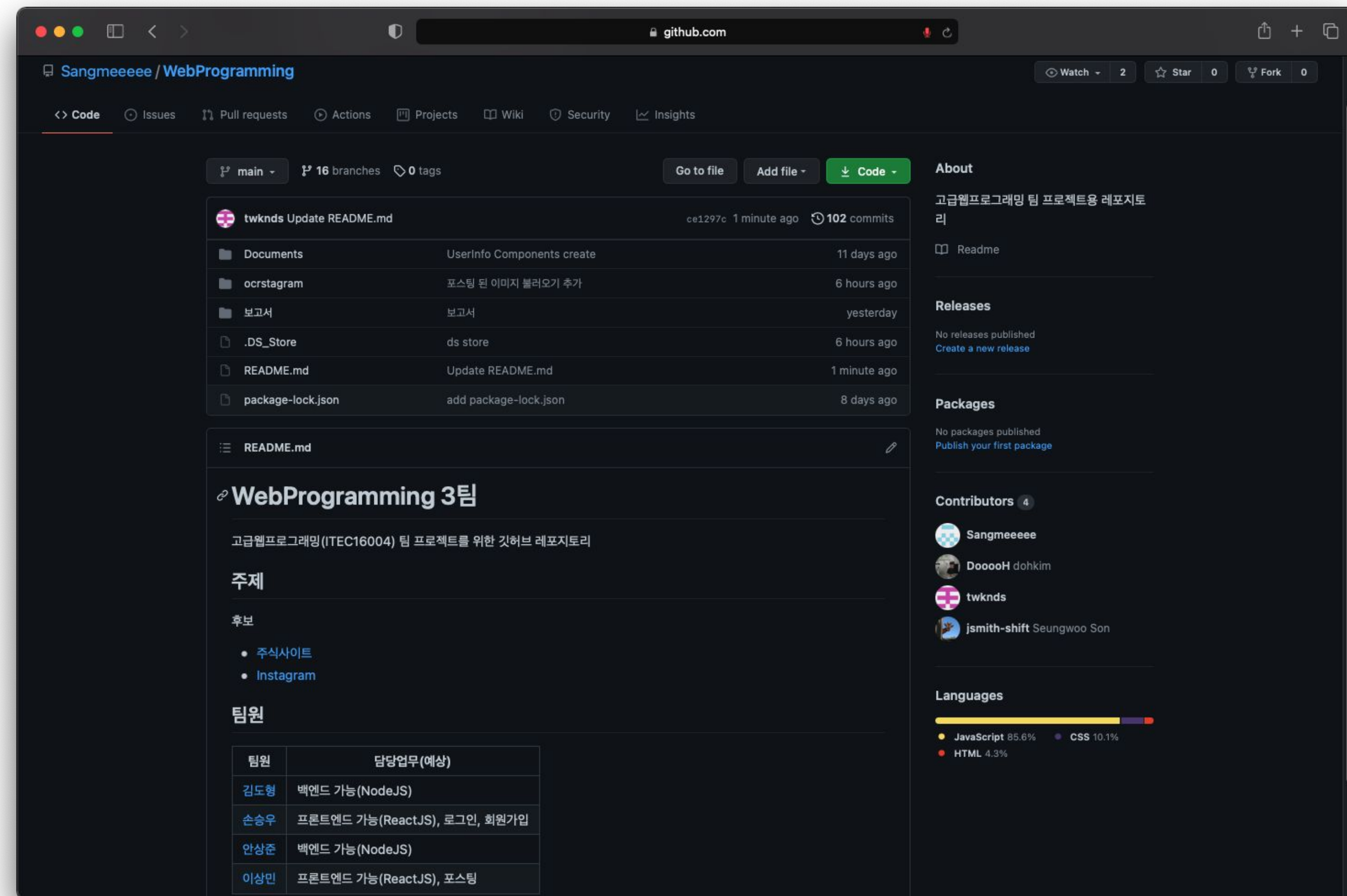
몽고DB

04

협업 내용

- 깃허브

■ 깃허브



05

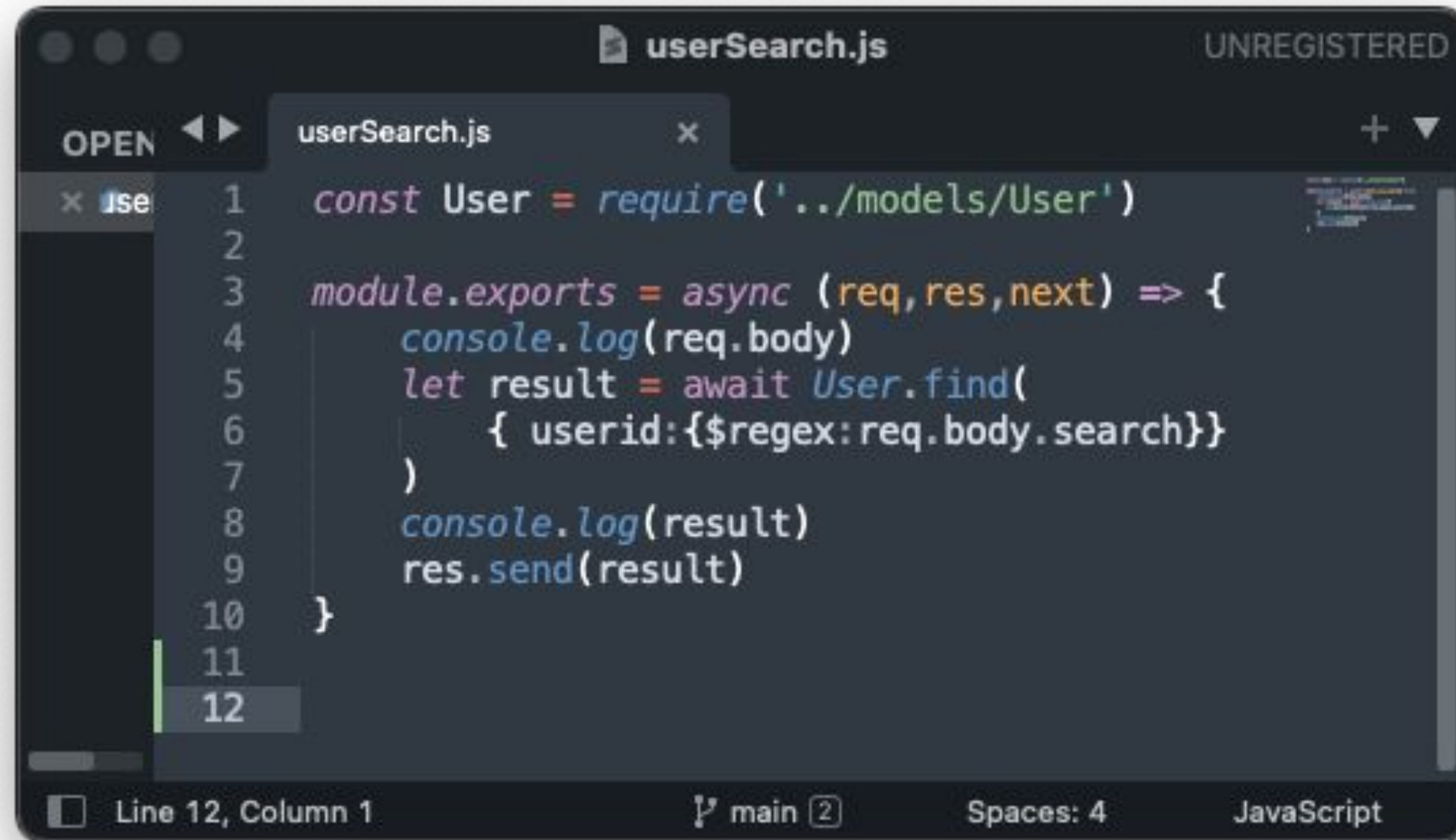
향후 계획

- 유저 검색창 구현

■ 향후 계획

ocrstagram

유저 검색창 구현



```
1  const User = require('../models/User')
2
3  module.exports = async (req, res, next) => {
4    console.log(req.body)
5    let result = await User.find(
6      { userid: {$regex: req.body.search}}
7    )
8    console.log(result)
9    res.send(result)
10  }
11
12
```

Line 12, Column 1 main 2 Spaces: 4 JavaScript

- 유저 검색을 통해 가입된 유저들의 피드에 접근할 수 있도록 구현할 계획

ocrstagram

감사합니다.

Github 주소 :
<https://github.com/Sangmeeeeeee/WebProgramming>
