정용훈 | Backend Engineer

저는 FastAPI, Docker, NGINX 등을 활용해 개인 서버와 클라우드 환경에서 서비스를 배포하고 운영한 경험이 있으며, 다양한 DB를 활용해본 경험을 바탕으로 서비스에 적합한 데이터베이스를 선택하고 설계할 수 있는 능력을 갖추고 있습니다. 새로운 프레임워크나 기술을 빠르게 습득하고 적용할 수 있는 능력도 제 강점입니다.

Github : https://github.com/hun9008

Blog : https://only-advnc.tistory.com/

Linkedin : http://www.linkedin.com/in/hun-55b69623a

Awards

- 2024.08 / SKT FLY AI 개인부분 / 우수상
- 2024.09 / Side Impact Spark 부분 / Round 1 통과
- 2024.06 / 생성 AI 활용 소프트웨어 아이디어 공모전 / 장려상
- 2024.03 / 국가우수장학금(이공계) / 2년 전액 장학
- 2023.10 / AjouTon (아주 교내 해커톤) / 우수상

Core Projects

진행한 프로젝트 중, 핵심 프로젝트입니다. (⭐ : 제가 사용한 기술 스택입니다.)

Maitutor - LLM 기반 중학생 수학 교육 지원 서비스

Info

Github : https://github.com/hun9008/SKT_FLY_AI_Maitutor

Date : 2024.07 ~ 2024.09

Teck Stack : ReactNative, FastAPI⭐, MySQL⭐, MongoDB⭐, Docker⭐, NGINX⭐, Llama3.1⭐, Azure⭐

Motivation

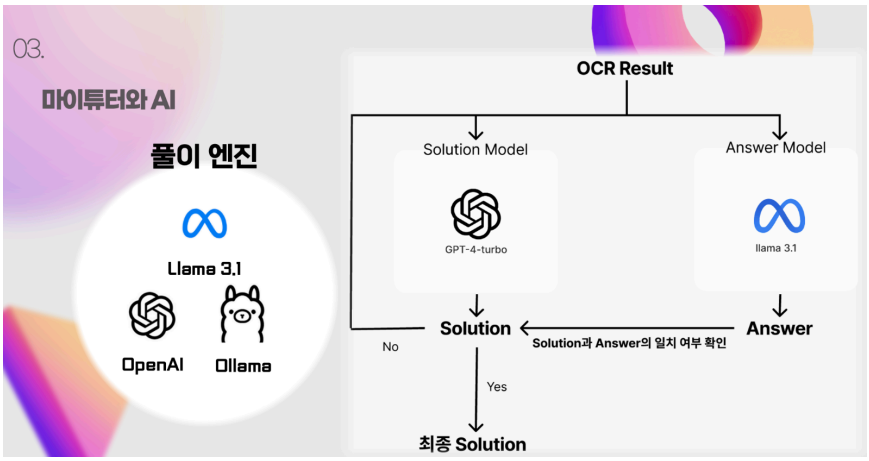
교육 취약 지역의 많은 학생들은 과외 선생님이나 학원등의 인프라 부족으로 수학 학습에 어려움을 겪고 있습니다. 이를 해결하고자 LLM 기반의 개인화된 수학 학습 지원 서비스를 만들고자 했습니다.

Service Key Features

- AI 기반 튜터링 : 텍스트와 손으로 쓴 수학 문제를 LLM 기반으로 문제를 해결하고 개인화된 풀이 제공.
- OCR : Google Vision, OpenCV를 통합하여 카메라를 통해 문제와 유저의 풀이를 인식.
- 유저 풀이 진행상황 피드백 : 유저의 풀이 OCR 결과에 대해 문제 풀이 상태를 {doing, wrong, solve, delay} 중 하나로 판단해 모니터링.
- 학습 데이터 분석 : 문제를 푸는 동안 유저의 학습 정보를 수집해 저장 및 분석 보고서 제공.
- 흥미 유발 요소 : 학생들의 적극적 참여를 위해 학교별 대회 및 랭킹 시스템 도입 및 스코어보드 제공.

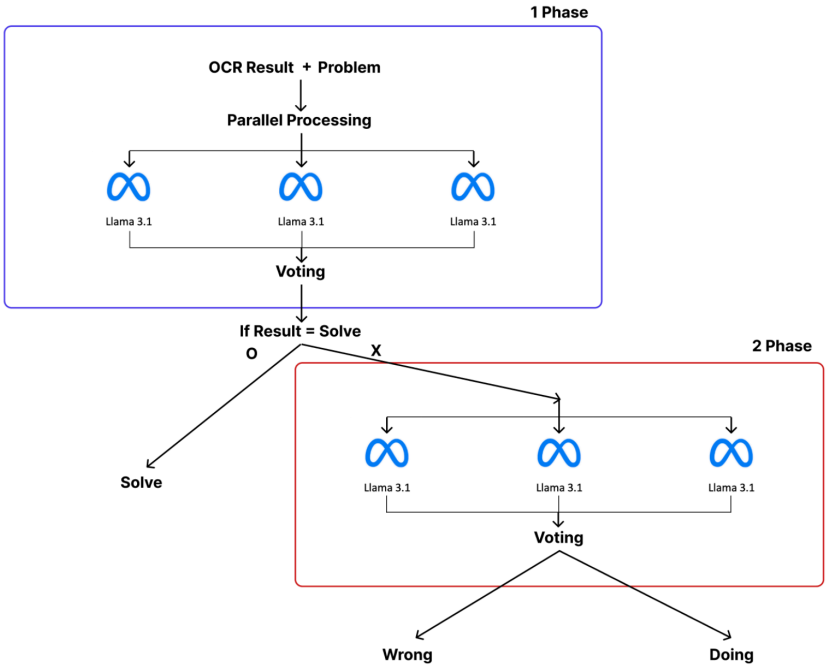
What did I do

- LLM 풀이 엔진 설계 및 구현 - 수학 문제 풀이의 정확도를 높이기 위한 풀이 엔진 설계



- 여러 모델(InternLM, Mistral, EXAONE, OpenAI API 등)을 고려했으나, 단일 모델로 수학 문제의 정답과 풀이를 모두 생성하는 데는 한계.
- OpenAI API를 이용한 문제 풀이 생성. (MATH benchmark SOTA)
- Llama 3.1을 이용한 문제 정답 생성. (8b 모델 중, GSM8K benchmark 상위권)
- 정답 모델의 정답과 풀이 모델의 풀이의 정답이 다른 경우, 풀이를 다시 생성하도록 하여 잘못된 답에 대한 풀이가 생성되지 않도록 구현하여 문제를 해결.
- FastAPI로 model 서버의 API 구현. Github /problem_solver API

- 유저 진행상황 판단 mechanism 설계 및 구현

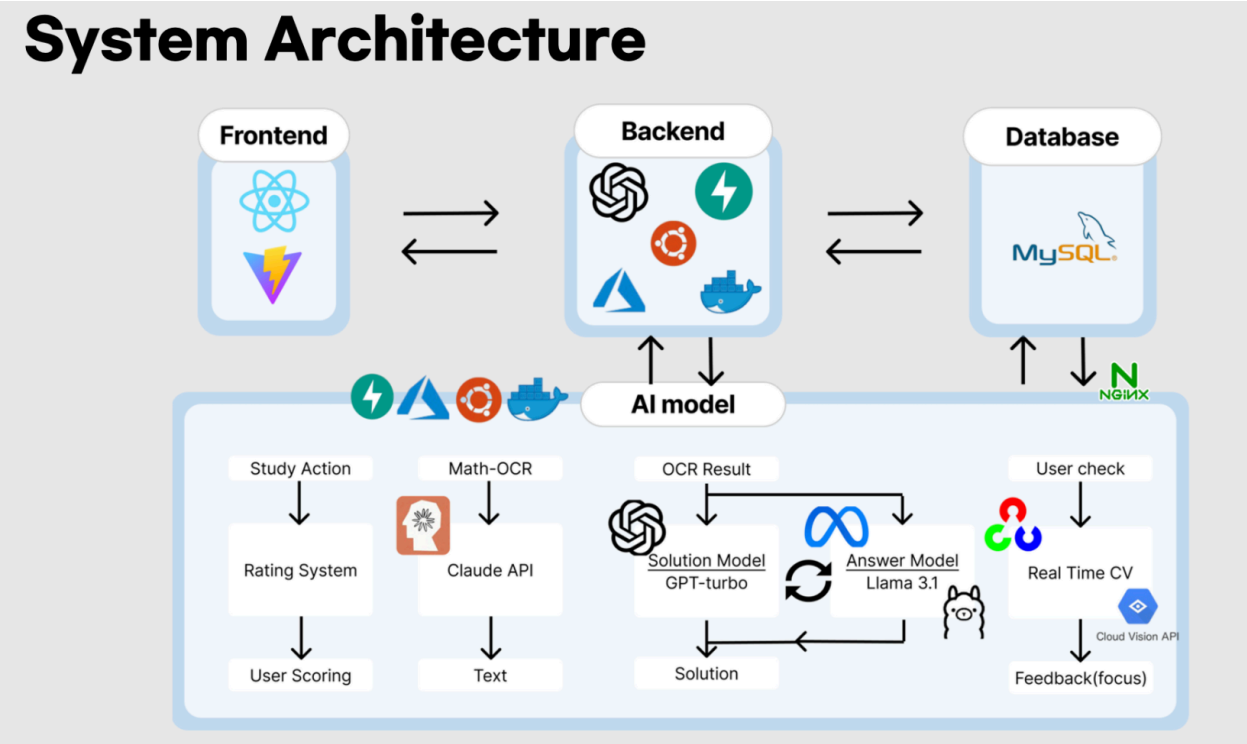


- GoogleVision OCR API를 통한 유저 손글씨 인식.
- OCR과 문제를 LLM input으로 넣어 진행상황 판단.
- LLM의 output이 Stable하지 않은 문제 발생.
- LLM output의 결과와 inference time을 고려하여 가장 빠르고 정확했던 2Phase, 3Voting으로 선택.
- FastAPI로 model 서버의 2Phase 3Voting API 구현. Github /hand_ocr API, 해당 Commit

- DB Migration
  - 초기에는 간단한 유저 정보만 가지고 있어, NoSQL (MongoDB Atlas)를 이용.
  - 개인화된 풀이 제공을 위해 유저 학습 정보를 저장 및 관리해야 하는 요구사항과 대회 및 랭킹 시스템이 추가되어 여러 object가 한 object에 중첩이 되는 문제 발생.
  - RDB (MySQL)로 migration을 진행하고 정규화를 통해 중첩된 Entity를 분리. Commit

- 대회 및 랭킹 시스템 설계 및 구현
  - Backend 서버의 background로 하루에 한번 문제 DB에서 레벨 별 10문제씩 뽑아 그날 대회 문제집 생성 구현. Github
  - 대회 문제 조회 및 채점 API 구현. 대회 API Commit, Scoring System 개선 Commit

System Architecture



AI VAT - 강아지 백내장 진단 앱 서비스

Info

Github : https://github.com/hun9008/Dev_Cataract

Demo Page : https://aivet.hunian.site/

Date : 2024.05 ~ 2024.09

Teck Stack : ReactNative, NextJS⭐, FastAPI⭐, MongoDB⭐, Docker⭐, NGINX⭐

Motivation

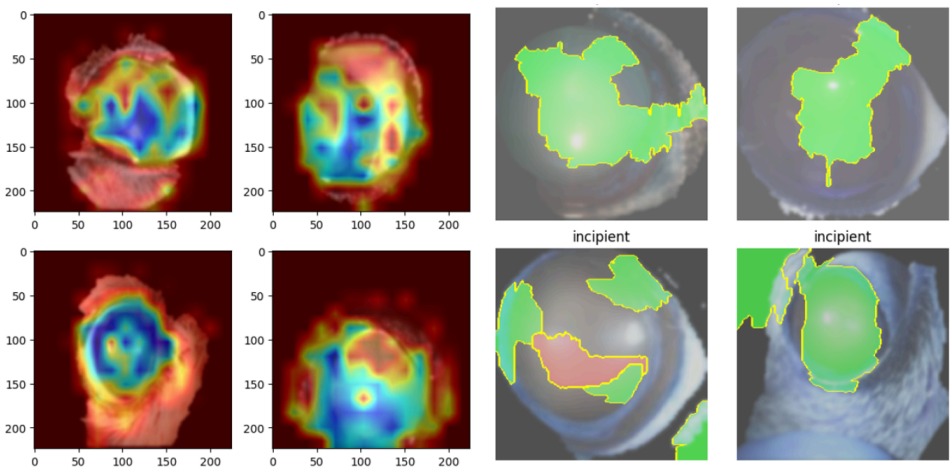
반려동물 양육인구 증가에 따라 반려동물의 건강 관리의 필요성을 AI 모델을 통한 앱 서비스로 해결하고자 개발을 진행했습니다. ViT(Vision Transformer)모델을 통한 백내장을 {초기, 중기, 말기, 정상} 으로 진단해주고, GradCAM과 LIME 기법을 통해 모델의 예측 이유를 시각적으로 보여줍니다.

Service Key Features

- AI 진단: AI 기반의 24시간 반려견 안구 질환 진단 서비스로, 동물병원을 방문하지 않고도 신속하게 반려견의 건강 상태를 확인할 수 있습니다. 반려견의 눈 사진을 업로드하면 ViT 모델을 통한 백내장 상태 진단을 제공하고 모델의 판단결과를 시각적으로 보여줍니다.
- 전문가 상담: AI 진단 후 필요 시 전문가와의 상담을 통해 추가적인 치료 계획을 세울 수 있습니다.
- 커뮤니티: 진단 결과를 커뮤니티에 공유하여 다른 사용자들과 상호작용하거나 전문가의 추가 의견을 얻을 수 있습니다.

What did I do

- ViT 모델 Finetuning 및 모델 배포

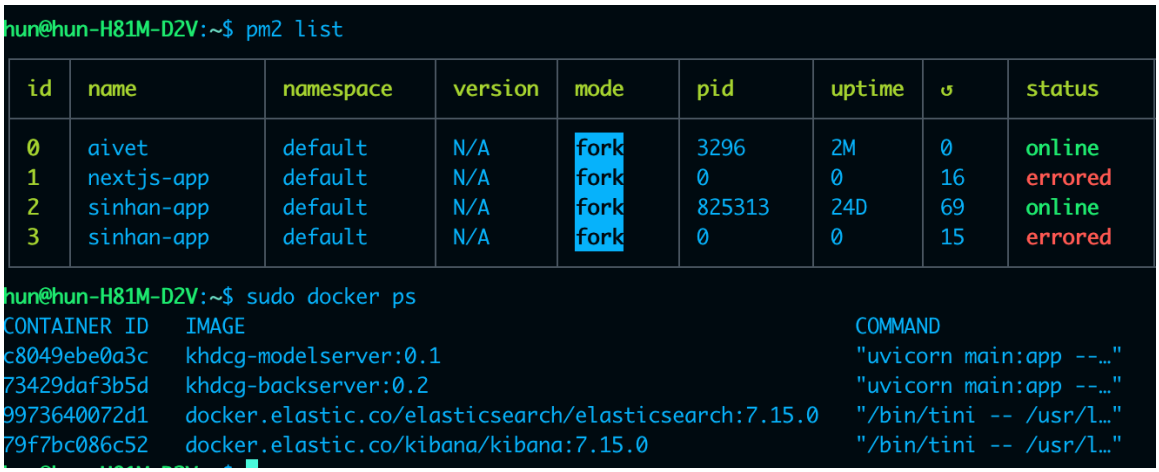


  - ImageNet으로 PreTrain된 vit_b_16 모델의 마지막 layer에 대해 4개 class, 8000장의 사진으로 Fine Tuning. Github
  - 모델 컴퓨팅 리소스가 많아, 백엔드와는 구분되는 별도의 모델 서버 구성.

- FastAPI로 모델 서버의 inference 요청 API 구성. Github /inference
- Inference에 1분 30초 정도 소요되는 문제를 해결하기 위해, 모델 서버에서 직접 DB에 결과를 저장하도록 했습니다. 또한, 동일 유저가 다시 요청할 경우 모델을 재구동하지 않고 DB에 저장된 사진을 바로 제공하도록 구현.

- 회원관리 CRUD API 개발
  - 유저 관리에 대한 Login, SignUp, Pet Update API 개발.
  - 초기에 일반 유저와 의사 유저를 별도의 Entity로 개발했으나, 대부분의 속성이 동일한 문제를 해결하기 위해 하나의 유저 Entity로 통합하고 type 속성을 추가하여 의사와 일반 유저를 구분하도록 변경했습니다. Github Backend

- Backend, Model 서버 및 데모 페이지 배포



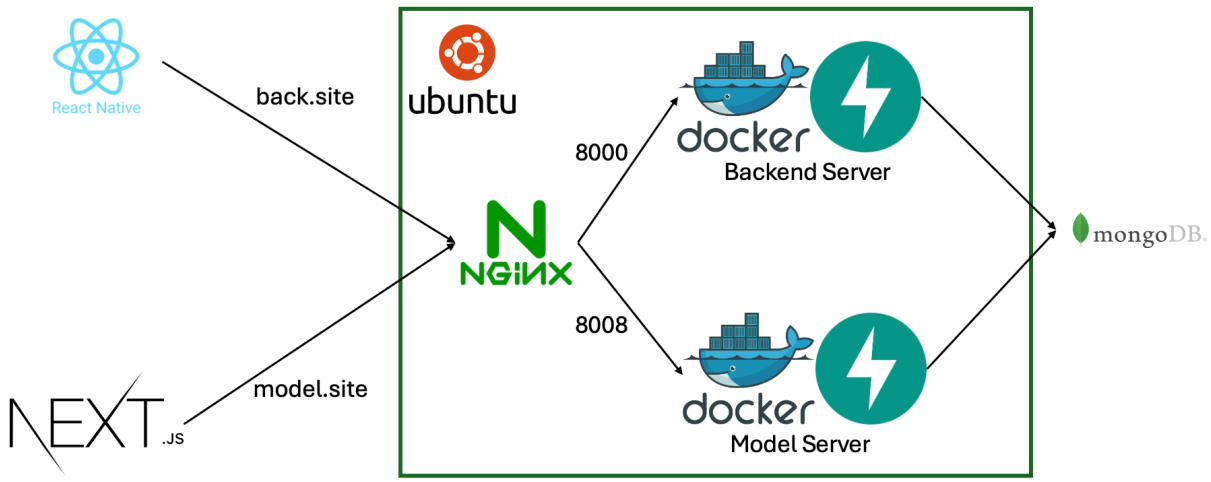
  - 서비스 배포를 위해 개인 서버에 Backend와 Model 서버를 Docker로 특정 포트에 구동하고, NGINX로 도메인 요청을 포워딩.
  - 개발 OS와 서버의 OS와 Library 지원 사항이 달라 힘들었으나, 개발 가상환경의 라이브러리를 서버 OS와 최대한 맞춰 설정하여 호환성 문제를 해결.
  - 데모 페이지의 경우 pm2를 이용해 구동시키고 마찬가지로 특정 포트에 실행하고 NGINX로 포워딩.

- 데모 페이지 제작

> aivet.hunian.site
> AI 기반 앱은 반려동물의 백내장을 쉽게 감지하고 모니터링하는 데 도움이 됩니다. 조기 진단을 받고 상태의 진행 상황을 추적하세요.
> https://aivet.hunian.site

  - Vercel V0를 이용해 전체적인 아웃라인을 잡고, 디자인과 세부사항을 수정.
  - 테스트를 위해 모델 서버로의 요청(Request) 기능을 구현. Github

System Architecture



Skills

Strong

C++ / Python / FastAPI / MySQL / MongoDB / Docker / NGINX

Knowledgeable

Java / JavaScript / Spring / React / NextJS / Elastic Search

ETC

Git / AWS / Azure / PM2

Side Projects

핵심 프로젝트 외에 진행한 프로젝트 입니다.

개인 우분투 서버 제작

Info

Github : https://github.com/hun9008/Personal_Ubuntu_Server

Date : 2024.07

Teck Stack : Ubuntu, NGINX

Motivation

평소 AWS EC2나 Azure와 같은 클라우드 서비스를 자주 이용했지만, 요금이 부담되고 자유도가 떨어진다는 생각이 들어 방학때 개인 서버를 만들어봤습니다. 현재 기숙사에 살고있어 서버 운용에 별다른 비용이 들지 않아, 바로 당근에서 10만원짜리 중고 윈도우 PC를 사서 서버 구축을 시작했습니다.

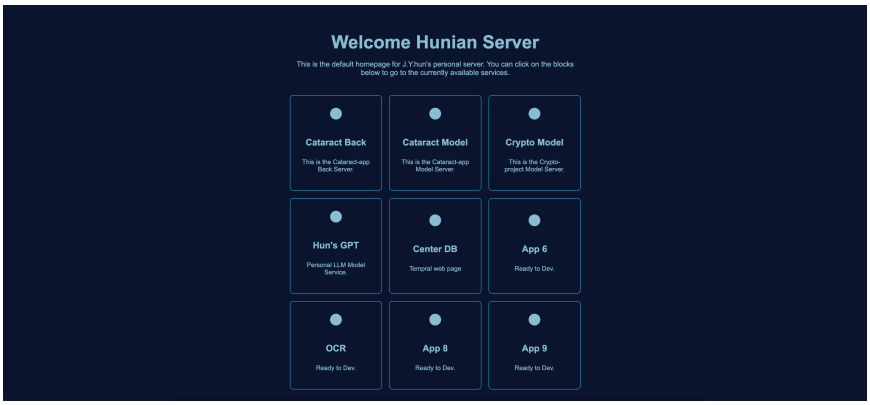
Environmental Check

우선 기숙사 공유기 관리 페이지에 접속할 수 있는지 확인했습니다. 다행히 192.168.0.1 (Private IP address)로 공유기 관리 페이지에 접속할 수 있어, 포트 포워딩이 가능하겠다는 판단이 되어 컴퓨터를 바로 구매했습니다. 구매한 PC는 i5, HDD 1TB, ssd 512GB, RAM 16GB로 AWS 프리티어 EC2에 비하면 훨씬 좋은 사양입니다.

OS Setting

개인적으로 Linux 명령어가 편하기도 하고, 서버로만 쓸거라 Ubuntu 24.04로 교체하기로 했습니다. Mac을 사용중이라, BalenaEtcher를 이용해 USB를 포멧하고 Ubuntu iso로 부팅용 USB를 생성했습니다. 만든 USB로 Ubuntu를 설치하고 Window를 삭제했습니다. 여기서 USB부팅 순서때문에 한참 헤맸었는데, BIOS에서 부팅 순서를 바꿔주니 금방 해결됐습니다.

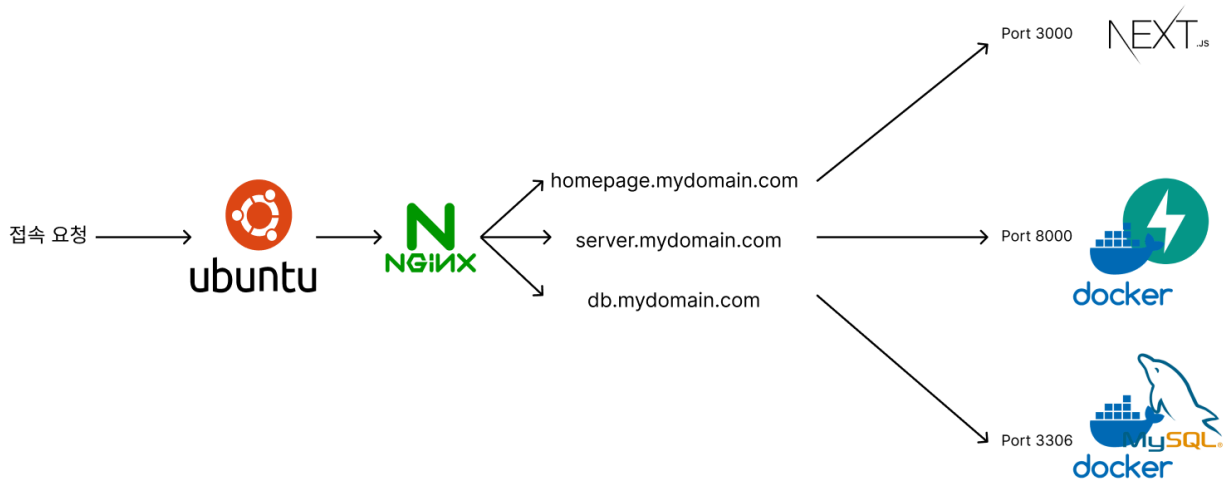
NGINX & DNS



다른 기능보다, 리버시 프록시를 위해 NGINX를 사용했습니다. 구축한 서버에는 여러 웹사이트와 백엔드 서버, 모델 서버와 DB까지 모두 서빙할 계획이라, 도메인 네임을 구매하고 특정 도메인으로 들어오면 내부 특정 포트로 포워딩하기 위해 NGINX를 이용했습니다. Gabia에서 도메인을 하나 구매하고 A tag는 자유롭게 생성할 수 있기에 제가 구매한 도메인의 앞의 A 테그를 보고 NGINX를 이용해 특정 포트로 포워딩해줬습니다.

추가로 nginx의 default index html을 수정해서 제 도메인으로 접속했을 때, 가이드 페이지를 간단히 제작해봤습니다.

System Architecture



PostMeeting - 학교인증기반 소개팅 사이트

Info

Github : https://github.com/hun9008/post_meeting/tree/main/front

Date : 2023.09 ~ 2023.11

Teck Stack : React⭐, FastAPI, MongoDB, AWS S3⭐, AWS CloudFront⭐, AWS EC2, AWS Route53⭐

Motivation

학교 축제에서 포스트잇을 활용한 미팅 이벤트가 열렸는데, 이 행사에서 영감을 받아 룸메이트와 함께 이를 웹 서비스로 구현해보면 재미있겠다는 생각이 들었습니다. 단순히 오프라인 이벤트에 그치는 것이 아니라, 온라인에서도 비슷한 방식으로 학교 인증 기반 미팅 플랫폼을 만들고자 했습니다.

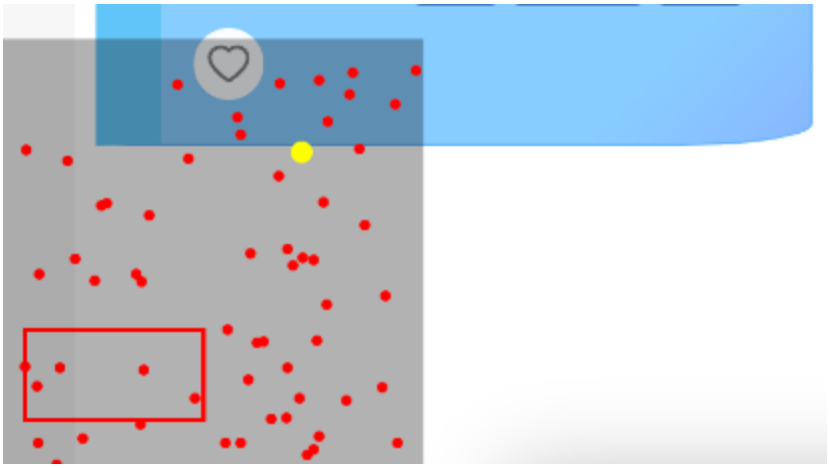
Service Key Features

- 포스트잇 게재: 넓은 화이트 보드에 자신의 MBTI, 성격 등 간단한 정보를 적은 포스트잇을 게재하여 자신을 소개할 수 있습니다. 이를 통해 관심 있는 상대방에게 첫인상을 전달할 수 있습니다.
- 마음에 드는 상대와 채팅: 마음에 드는 포스트잇을 발견하면 해당 상대와 익명 채팅을 시작할 수 있어, 서로의 관심사를 더 깊이 알아가는 시간을 가질 수 있습니다.
- 좋아요 받은 상대 정보 확인: 나를 좋아요한 상대의 정보를 확인하여 관심 있는 상대와의 연결 가능성을 높이고, 서로의 호감을 바탕으로 더 발전된 대화를 이어갈 수 있습니다.

What did I do

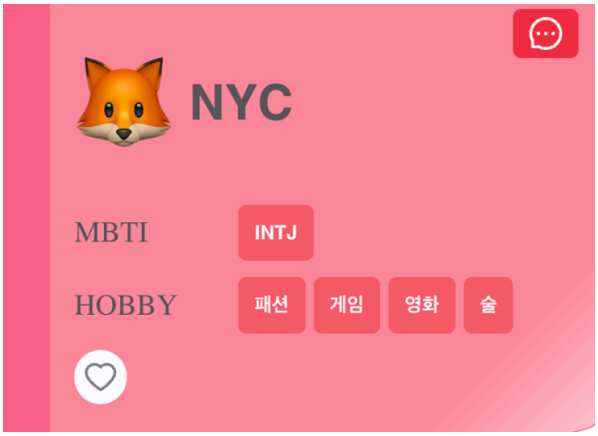
- 채팅 동기화.
  - 처음에는 채팅방을 이동할 때 이전 채팅 내역이 사라지는 문제가 있었는데, 이를 해결하기 위해 채팅방별로 이전 메시지를 유지하고 로드하는 방식을 도입.
  - chatRoomList 상태에는 각 채팅방의 room_name과 해당 방의 chat_list가 포함.
  - 새로운 채팅방을 생성하거나 기존 채팅방을 로드할 때, 각 채팅방의 이전 메시지 내역(chat_list)을 유지할 수 있도록 구현.
  - 사용자가 다른 채팅방으로 이동할 때 changeReceiverId 함수를 통해 receiver_id를 변경하고, chatRoomList에서 해당 room_name을 가진 채팅방의 chat_list를 찾아 response 상태로 설정.

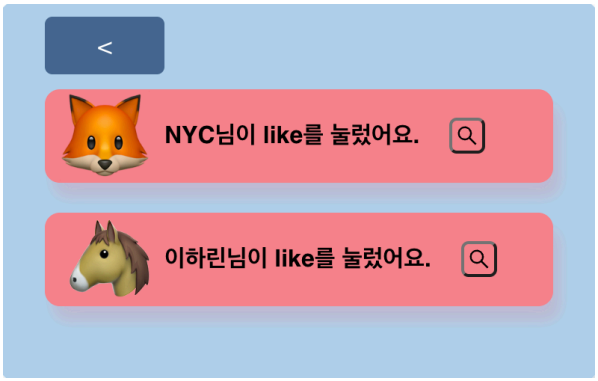
- 자신의 위치를 확인하는 Minimap 구현.



  - 처음에는 미니맵의 위치와 실제 포스트잇의 위치가 일치하지 않는 문제가 있었는데, 이를 해결하기 위해 포스트잇의 좌표를 미니맵의 축소 비율에 맞춰 변환하는 방식을 도입.
  - renderPostitPoints 함수에서 포스트잇의 x, y 좌표를 전체 스크롤 영역의 크기로 나눈 후, 미니맵 크기에 맞게 비율을 조정하여 위치가 정확히 매핑되도록 구현.

- UI / UX 디자인 (SCSS)





System Architecture