DEVELOPER PORTFOLIO

학습과 **성장**을 즐기는 개발자

이창현
changhyu953@gmail.com
apple141410@naver.com

SKILL

C (Proficient)

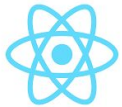
C 언어는 컴퓨터 과학의 핵심 기초입니다. **Minishell 프로젝트를 통해 Bash 쉘 작동 방식 구현**하며 시스템 프로그래밍 역량을 키웠습니다. **메모리 관리, 프로세스 관리** 등 핵심 시스템 레벨 이해도를 높였습니다.

C++ (Intermediate)

C++은 객체 지향 프로그래밍의 강력한 언어입니다. **Socket 통신 기반 HTTP 서버 직접 구현**하며 C++ 객체 지향 설계 및 저수준 시스템 프로그래밍 역량을 키웠습니다.

Django (Intermediate)

Django는 Python 기반 웹 프레임워크입니다. **Django와 Socket 통신을 활용**하여 **실시간 유저 상호작용** 웹 서비스 개발 경험이 있습니다. MTV 패턴 기반 백엔드 개발 및 API 개발, 데이터베이스 연동 등 웹 개발 핵심 역량을 갖추었습니다.



React (Intermediate)

React는 컴포넌트 기반 프론트엔드 라이브러리입니다. **Redux를 활용한 상태 관리** 경험을 통해 **클라이언트 중심 상태 관리** 웹 앱 개발 능력을 키웠습니다. 재사용 가능한 UI 컴포넌트 개발 및 효율적인 UI 구성 역량을 갖추었습니다.

Docker (Intermediate)

Docker는 컨테이너 기반 가상화 플랫폼입니다. **서비스 도커화**를 통해 프로젝트 배포 및 관리 효율성을 높인 경험이 있습니다. Docker 컨테이너 관리, Docker Compose 활용 등 컨테이너 기반 개발 환경 구축 역량을 갖추었습니다.

Node.js (Basic)

Node.js를 **활발히 학습하며 백엔드 개발 영역을 넓혀가고 있습니다. Django ASGI 기반 비동기 서버 개발 경험**을 바탕으로, Node.js 환경에서도 **기본적인 API 서버 구현**이 가능합니다. **다양한 백엔드 기술을 학습하고 활용하여, 폭넓은 역량을 갖춘 백엔드 개발자로 성장**하는 것을 목표합니다.

01 42_Transcendence

- **Django** 기반 실시간 핑퐁 웹 게임 사이트 개발 (사용자 간 즉각적인 반응형 게임 경험 제공)
- **Docker** 기반 환경 격리 및 간편 배포 구현 (개발/배포 효율성 향상)
- **REST API** 설계 및 개발 (프론트엔드-백엔드 효율적인 연동)
- **WebSocket** 기반 실시간 유저 상호작용 기능 구현 (몰입감 있는 멀티 플레이어 게임 환경 구축)

02 42_Webserv

- 외부 라이브러리 없이 순수 **C++ Socket** 통신 기반 **HTTP** 서버 직접 구현 (**C++** 저수준 프로그래밍 및 네트워크 프로토콜 심층 이해)
- 저수준 소켓통신 방식 구현 (메모리 관리, 성능 최적화 등 시스템 프로그래밍 핵심 역량 강화)

03 InsideOut

- **AI** 기반 감정 **TTS/클라우드** 시스템 활용 접근성 높은 **CAPTCHA 개발** (기존 텍스트/이미지 기반 **CAPTCHA**의 접근성 문제 해결)
- **Tacotron 2** 파인튜닝을 통한 감정 표현 **TTS** 생성 (악성 봇의 **CAPTCHA** 회피율 감소 및 보안 강화)
- **AWS CDK** 기반 인프라 자동화 구축 및 배포 (클라우드 환경 배포 및 관리 효율성 증대)

42_TRANSCENDENCE

🗓️ (24.12 - 25.02)

SKILL

Django

백엔드 프레임워크로 사용하여 웹 애플리케이션의 구조를 설계하고, 데이터베이스 모델을 정의하고, REST API를 개발했습니다. Django의 ORM을 사용하여 데이터베이스와 효율적으로 상호작용했습니다.

Django REST FrameWork

REST API를 효율적으로 설계하고 개발하기 위해 사용했습니다. Serializer를 사용하여 데이터 모델을 JSON으로 변환하고 사용하였습니다.

React

사용자 인터페이스를 개발하기 위해 사용했습니다. 컴포넌트 기반 아키텍처를 사용하여 UI를 모듈화하고, 상태 관리를 효율적으로 처리했습니다.

WebSocket

실시간 사용자 상호작용을 구현하기 위해 사용했습니다. 채널 레이어를 사용하여 WebSocket 연결을 관리하고, 메시지를 효율적으로 송수신했습니다.

Docker

개발 환경과 배포 환경을 일치시키고, 환경 의존성 문제를 해결하기 위해 사용했습니다. Dockerfile을 사용하여 애플리케이션 이미지를 빌드하고, Docker Compose를 사용하여 컨테이너를 관리했습니다.

프로젝트 소개

이 프로젝트는 Django, REST API, WebSocket을 활용하여 실시간 웹 Ping Pong 게임 사이트를 개발하는 것을 목표로 했습니다. 42 경산 학생들을 대상으로 하며, 간단하고 재미있는 실시간 온라인 탁구 게임 경험을 제공하는 데 집중했습니다.

저는 이 프로젝트에서 Django 백엔드 개발을 담당하며, 특히 WebSocket을 이용한 실시간 게임 로직 구현에 집중했습니다.

실시간 사용자 상호작용을 구현하는 과정에서 WebSocket 연결 관리 및 메시지 송수신 문제를 겪었지만, 채널 레이어를 깊이 있게 학습하고 적용하여 안정적인 연결을 구축할 수 있었습니다. 또한, REST API 설계를 통해 효율적인 데이터 관리를 가능하게 했으며, Django ORM을 활용하여 데이터베이스와 효율적으로 상호작용했습니다.

팀원

- ✔CHyuni (Django 백엔드 개발, Django REST Framework를 사용한 API 개발, React 기반 프론트엔드 연동 및 Websocket을 이용한 실시간 통신 기능 구현)
- 1107c (Django 백엔드 개발, ELK 스택 기반 로그 분석 시스템 구축 및 모니터링 환경 구축, Websocket을 이용한 실시간 통신 기능 구현)
- joejaeyoung (Solidity 기반 스마트 컨트랙트 개발 및 배포)
- skyshr (React 기반 사용자 인터페이스 개발 및 유지보수)

42_TRANSCENDENCE

프로젝트 소개

LOGIN

로그인은 42 ID 및 Google 계정을 이용한 OAuth 2.0 인증을 통해 이루어집니다.

- 인증: 로그인 필수 사용자 프로필 API를 제공합니다.
  - 미인증 시: 401 Unauthorized 반환
  - 인증 시: 200 OK 및 사용자 프로필 반환
- 로그인 상태 관리: 반환된 프로필 정보를 활용합니다. 미로그인 시 홈 화면에 로그인 페이지를 표시합니다.

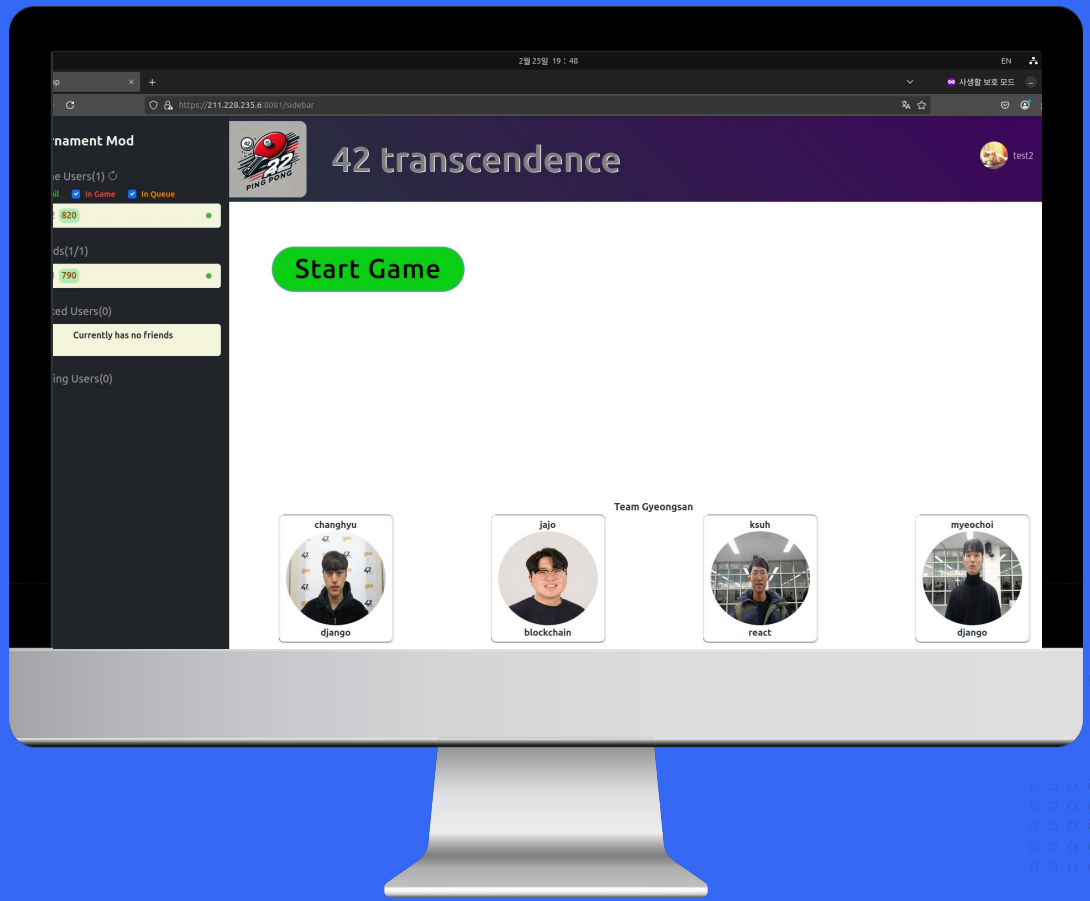
42_TRANSCENDENCE

프로젝트 소개

HOME

- 홈 화면은 사용자가 게임을 시작하고, 친구 목록을 확인하고, 프로필을 관리할 수 있는 중앙 허브 역할을 합니다. 모든 상호작용은 WebSocket을 통해서 이루어져 실시간성을 보장합니다.

- 로그인 시 Django ASGI 서버의 Consumer를 이용하여 관리되는 WebSocket 채널에 접속하여 실시간으로 유저, 친구 및 차단 목록, 매치메이킹 정보를 업데이트합니다.



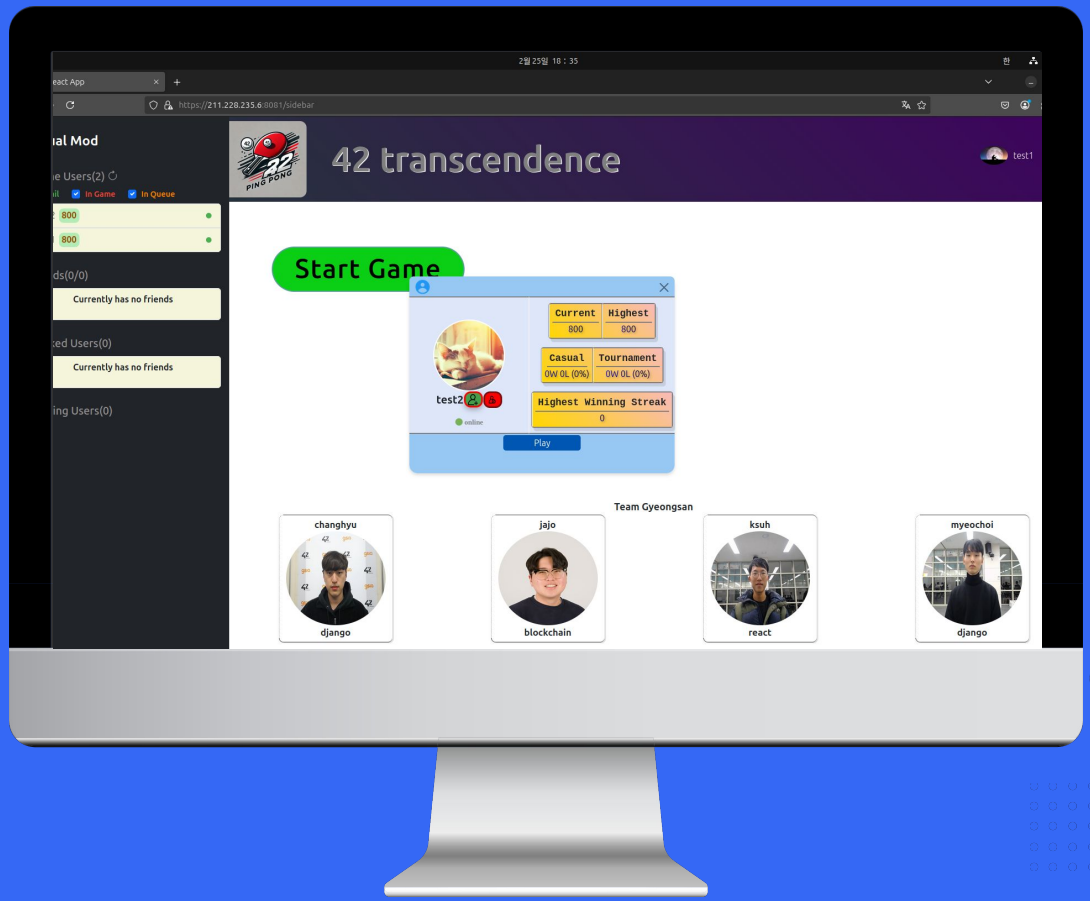
42_TRANSCENDENCE

프로젝트 소개

PROFILE

선택한 유저의 레이팅, 승패, 온라인 상태 등을 확인할 수 있는 프로필 화면입니다. 이 화면에서 다음과 같은 상호작용이 가능합니다.

- 커스텀 게임 요청: 상대방에게 직접 게임을 요청할 수 있습니다.
- 친구 추가: 상대방을 친구로 추가하여 실시간 대화 및 게임 초대를 할 수 있습니다.
- 차단: 원치 않는 상대방을 차단하여 상호작용을 제한할 수 있습니다.
- 채팅: 친구로 추가된 상대방과 실시간으로 대화할 수 있습니다.
- 이러한 상호작용은 모두 WebSocket을 통해 실시간으로 이루어지며, 사용자 경험을 향상시킵니다.



42_TRANSCENDENCE

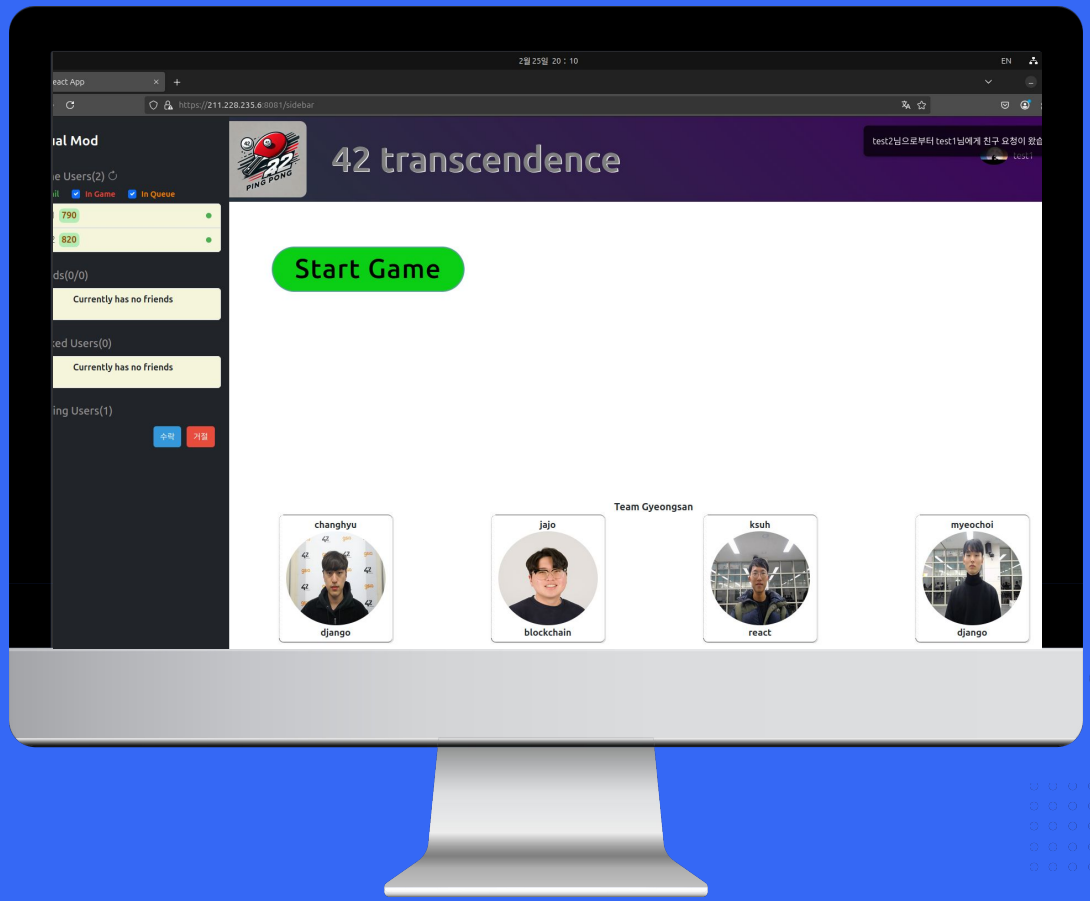
프로젝트 소개

SEND FRIEND REQUEST

친구 추가 요청 시, 상대방 유저에게 ToastMessage 알림이 전송되고, Pending Users 목록에 요청이 생성됩니다.

- 수락: 상대방이 요청을 수락하면, 해당 유저가 Friends 목록에 추가됩니다.
- 거절: 상대방이 요청을 거절하면, 요청이 상대방 및 해당 유저의 Pending Users 목록에서 제거됩니다. (별도의 알림은 제공되지 않습니다.)

해당 알림 전송, 요청 생성 등은 서버 공지 WebSocket을 통해 상호작용한 대상에게 실시간으로 안내 됩니다.



42_TRANSCENDENCE

프로젝트 소개

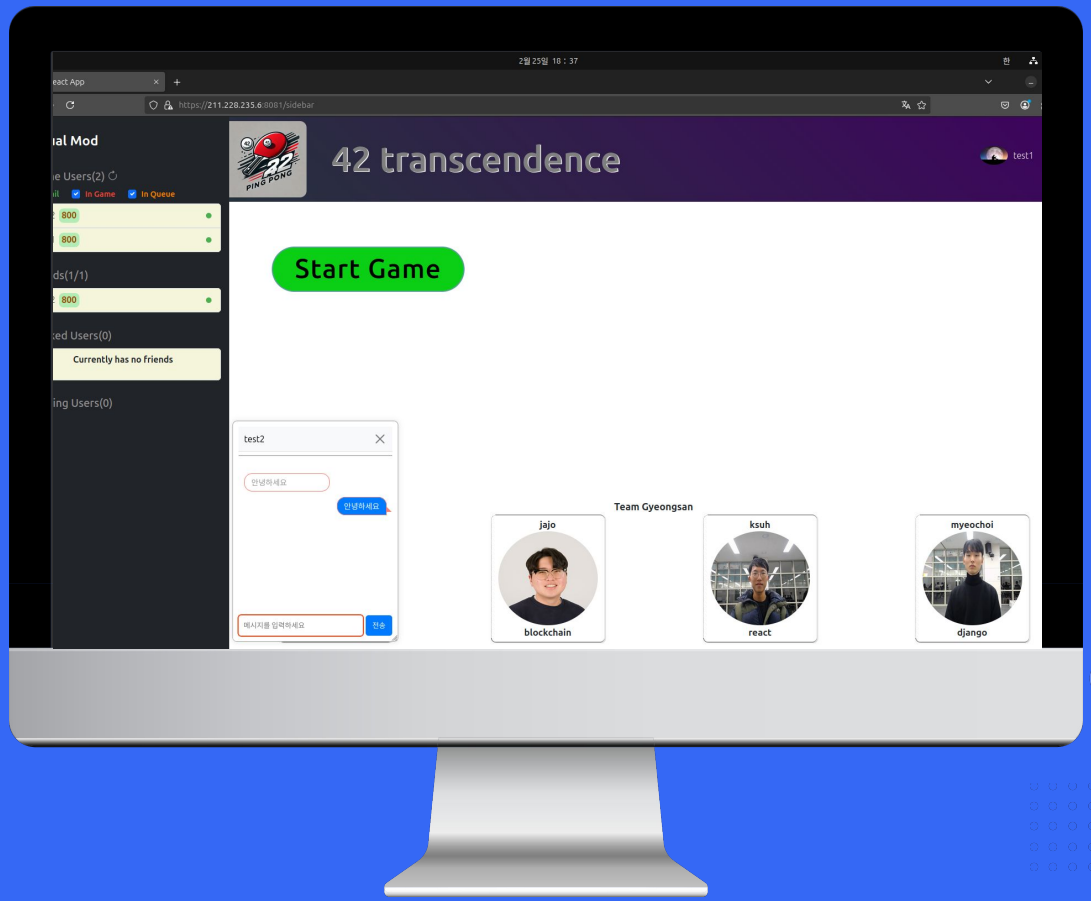
DIRECT MESSAGE

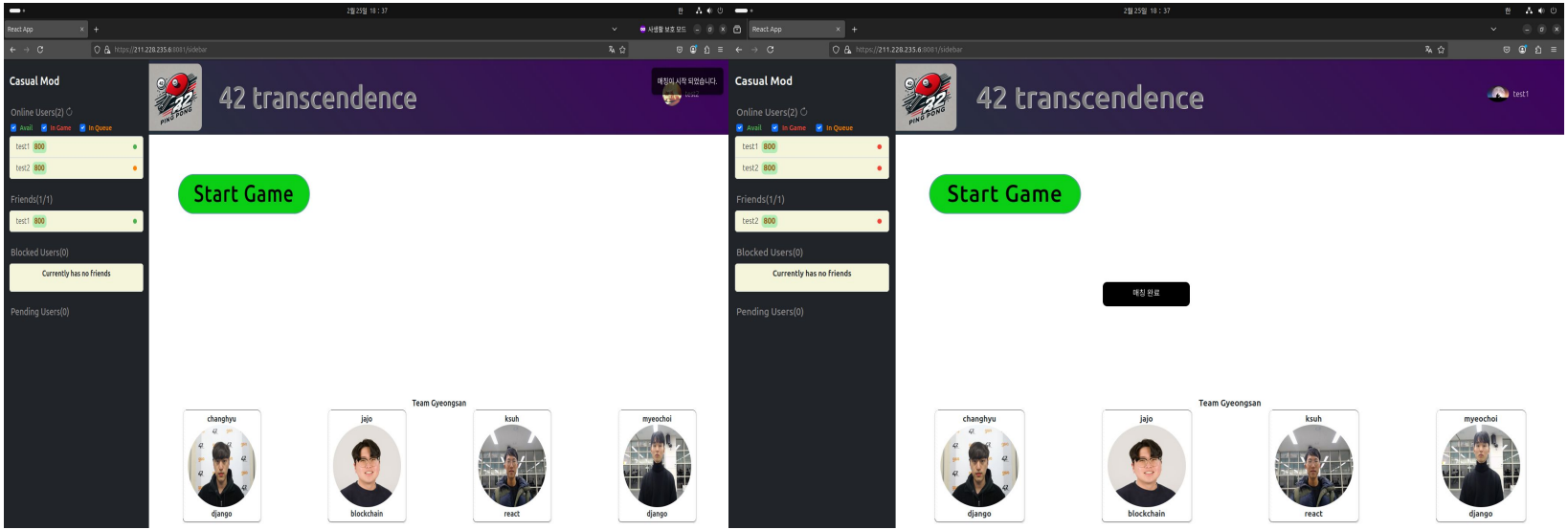
친구 관계인 상대와 실시간으로 대화할 수 있는 시스템입니다.

- 친구 관계: 상대방과 친구 관계인 경우, 프로필 화면의 왼쪽 상단 이미지 클릭 시 드롭다운 메뉴를 통해 Chat 기능을 활성화할 수 있습니다.

- 친구 추가: 상대방과 친구 관계가 아닌 경우, Chat 기능 대신 친구 추가 메뉴가 활성화됩니다.

채팅 기능 활성화 시, 본인과 상대방의 Username 조합을 통해 채팅 소켓 채널을 동적으로 생성합니다. 생성된 채널 이름은 DB에 저장하여 채팅방을 구성하고, 이전 대화 내용을 보관합니다. 이를 통해 사용자는 언제든지 이전 대화 기록을 확인하고 이어서 대화를 진행할 수 있습니다.





MatchMaking

홈 화면의 Start Game 버튼 클릭 시 게임 진행을 위한 매칭 시스템이 시작됩니다.

- 매칭 방식: 현재 레이팅을 참조하지 않고, 데이터베이스를 이용하여 매칭 상태를 업데이트합니다. 매칭 테이블에 먼저 들어온 유저 순서대로 매칭이 완료됩니다. (향후 레이팅 기반 매칭으로 개선 예정)

매칭이 완료되면 ToastMessage를 이용하여 매칭 완료 알림을 보내고 게임을 시작합니다.

42_TRANSCENDENCE

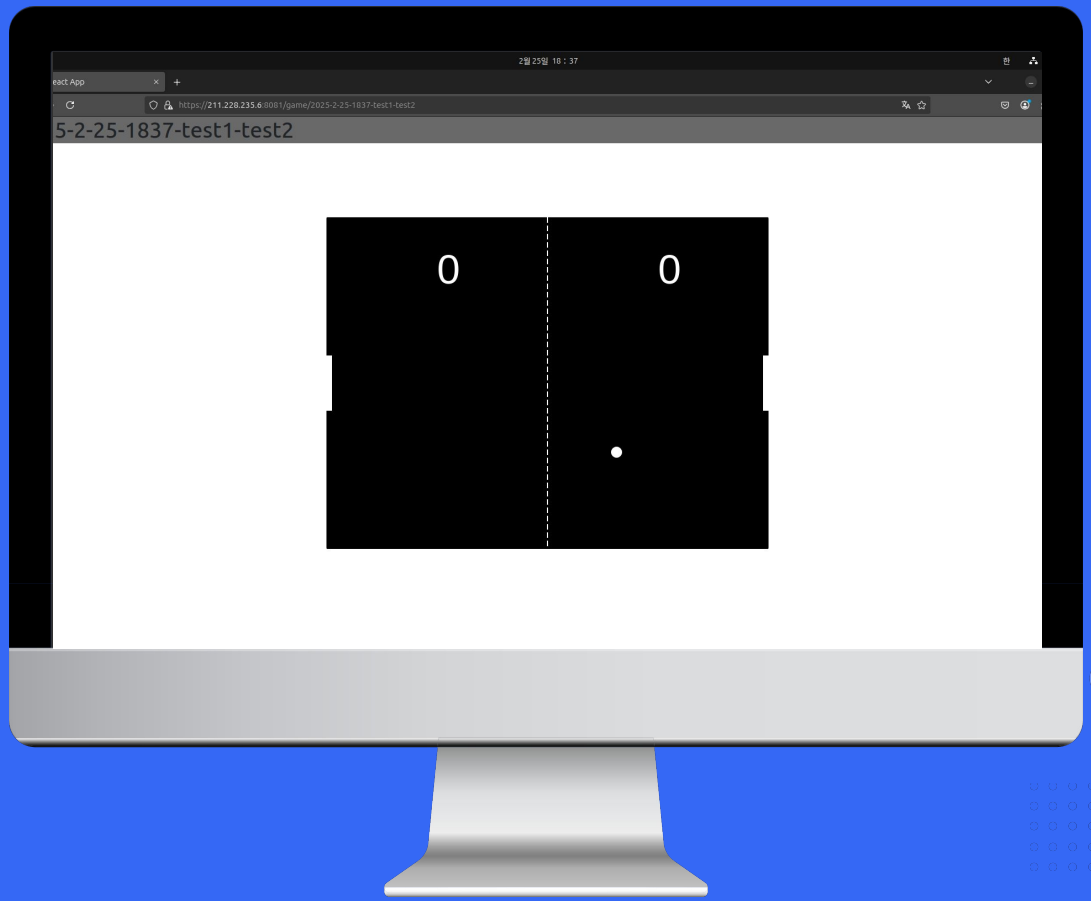
프로젝트 소개

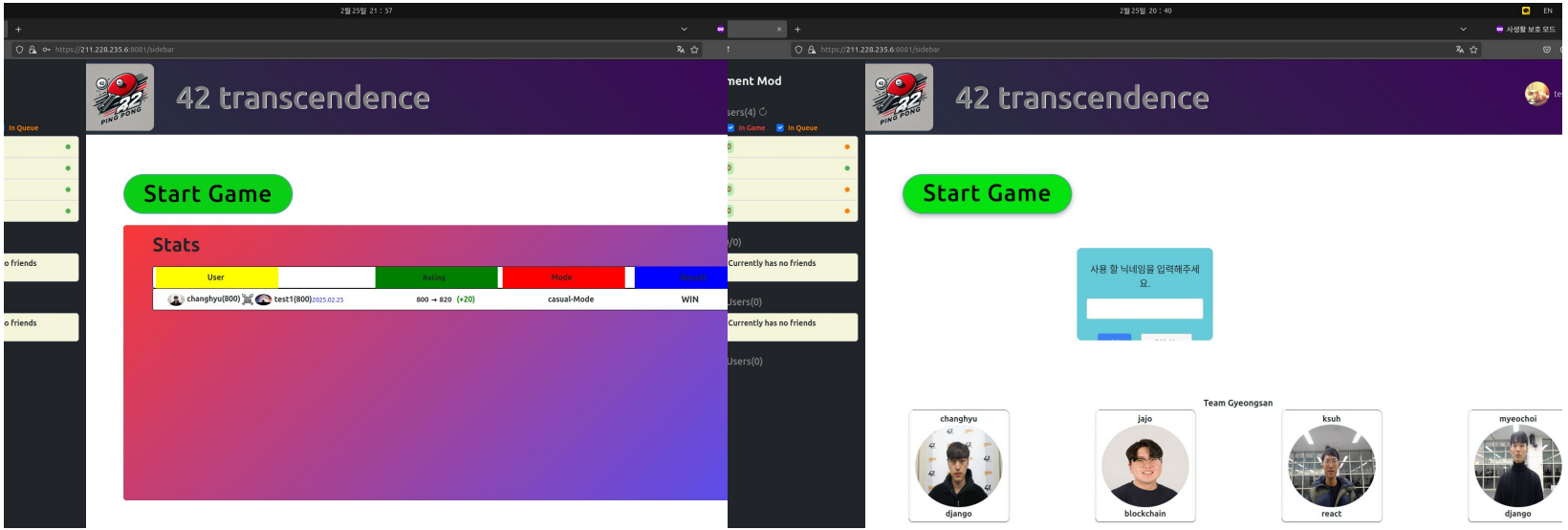
REAL-TIME GAME

게임 시작시 상대방과 새로운 Game Socket으로 연결되어 게임을 진행합니다.

- WebSocket 연결: 왼쪽 상단에 표시된 현재 날짜 및 시간, 상대방과 나의 Username을 이용하여 WebSocket 주소를 생성하고, 게임 진행 유저들이 접속합니다.

- 서버 사이드 관리: 클라이언트의 입력, 게임 결과 등 모든 정보는 서버에서 관리됩니다. 이를 통해 클라이언트에서의 데이터 변조를 방지합니다.





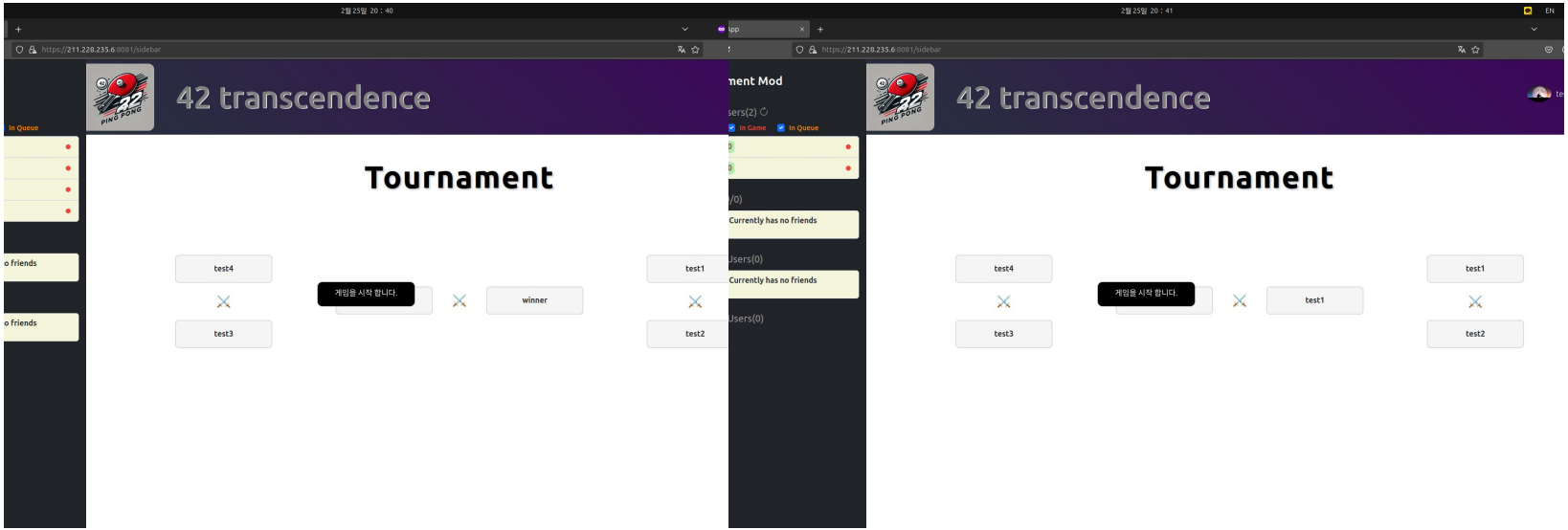
Game History

상단 우측 Profile image 클릭 시 나타나는 드롭다운 메뉴에서 Stat을 클릭하면 이전 게임의 결과를 확인할 수 있습니다.

Tournament

홈 화면 좌측 상단의 Casual Mod, Tournament Mod 클릭 시 게임 모드를 변경할 수 있습니다. Tournament Mod로 게임을 시작하면, 해당 토너먼트에서 사용할 닉네임을 입력할 수 있습니다.

- 닉네임: 닉네임을 입력하지 않으면 기본 Username이 사용됩니다. 닉네임은 해당 토너먼트에서만 표시되며, 통계 등 기록에는 해당 유저의 본 ID가 나타납니다.



Tournament Game

토너먼트 시작 시 ToastMessage를 이용하여 게임 시작을 알리고 대진표를 표시합니다.

1라운드 승리 시 기존 Winner 대진표 란에 승자가 표시되고, 승자 간 게임 대결이 시작됩니다. 패배한 유저는 Home으로 돌아가게 되고, 승자 대결 종료 후 모든 유저가 Home으로 복귀합니다.

프로젝트 결론

이 프로젝트를 통해 Django, REST API, WebSocket 등 다양한 기술을 경험하고, 실시간 웹 애플리케이션 개발 능력을 크게 향상시킬 수 있었습니다.

특히, WebSocket 연결 관리 및 메시지 송수신 문제를 해결하면서 실시간 통신 시스템에 대한 깊이 있는 이해를 얻을 수 있었습니다.

42_WEBSERV

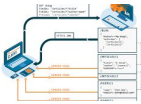
🗓️ (24.10 - 24.12)

C++

웹 서버의 기반 언어로서 사용했습니다. 객체 지향 프로그래밍 (OOP) 설계를 통해 코드의 **유지보수성**과 **확장성**을 높였으며, **C++ 표준 라이브러리 (STL)** 를 적극 활용하여 개발 생산성을 향상시켰습니다. **메모리 관리**를 직접 제어하여 **성능 최적화** 및 **안정적인 서버 운영**을 가능하게 했습니다.

Epoll

높은 동시성을 처리하기 위해 **Non-blocking I/O** 모델과 함께 사용했습니다. **많은 클라이언트의 요청을 동시에 처리해야 하는 웹 서버의 특성상**, 효율적인 I/O multiplexing 기술이 필수적입니다. **epoll 시스템 콜**을 활용하여 **파일 디스크립터 (소켓)** 의 상태 변화를 감지하고, 이벤트 기반으로 동작하는 웹 서버를 구현했습니다.



HTTP Parsing

클라이언트가 웹 서버에 보내는 요청을 이해하고 처리하기 위해 필수적인 HTTP 요청 파싱 을 구현했습니다. C++를 사용하여 **HTTP 요청 메시지** 를 **parsing** 하여 **요청 라인, 헤더, 바디**를 분석하고, 각 구성 요소에서 필요한 정보를 추출했습니다. **요청 메소드, 요청 URI, HTTP 버전, 헤더 필드** 등을 정확하게 파싱하여 웹 서버가 클라이언트의 **요청 의도를 파악**하고 적절한 처리를 수행하도록 했습니다. **파싱 과정에서의 오류 처리**를 통해 **안정적인 서버 동작**을 보장하고, **잘못된 요청에 대한 보안** 을 강화했습니다.

HTTP

웹 브라우저와의 통신을 위해 필수적인 HTTP/1.1 프로토콜 을 구현했습니다. **웹 서버와 클라이언트 간의 요청-응답** 과정을 정의하는 **HTTP 프로토콜 스펙**을 정확하게 이해하고, **요청 메시지 파싱, 응답 메시지 생성, HTTP 헤더 처리, 다양한 HTTP 메소드 지원, 상태 코드 관리** 등 웹 서버의 핵심 기능을 구현했습니다. **Keep-Alive** 기능을 구현하여 **네트워크 효율성**을 높이고, **웹 페이지 로딩 속도**를 개선했습니다. **웹 표준 프로토콜 준수** 를 통해 다양한 웹 브라우저 및 클라이언트와의 **호환성**을 확보했습니다.

이 프로젝트는 HTTP 프로토콜의 동작 방식과 클라이언트-서버 간 통신 과정을 깊이 있게 이해하고, 실제 웹 서버를 직접 구현하여 네트워크 프로그래밍 능력을 향상시키는 것을 목표로 진행되었습니다.

C++ 언어와 Epoll 기반의 I/O 멀티플렉싱 기술을 활용하여 고성능 웹 서버를 구현하고, HTTP 요청을 직접 파싱하여 응답을 생성하는 과정을 경험했습니다.

저수준 소켓 통신 과정에서 발생하는 다양한 문제점들을 해결하고, 멀티스레딩 환경에서의 동기화 문제를 해결하기 위해 노력했습니다. 본 프로젝트를 통해 HTTP 프로토콜, TCP/IP 소켓 통신, Epoll 기반의 I/O 멀티플렉싱, 그리고 멀티스레딩 프로그래밍에 대한 깊이 있는 이해를 얻을 수 있었습니다.

팀원

- ✔CHyuni (C++, Epoll, HTTP Parsing)
- 1107c (C++, Epoll, HTTP Parsing)

42_WEBSERV

프로젝트 소개

HOME

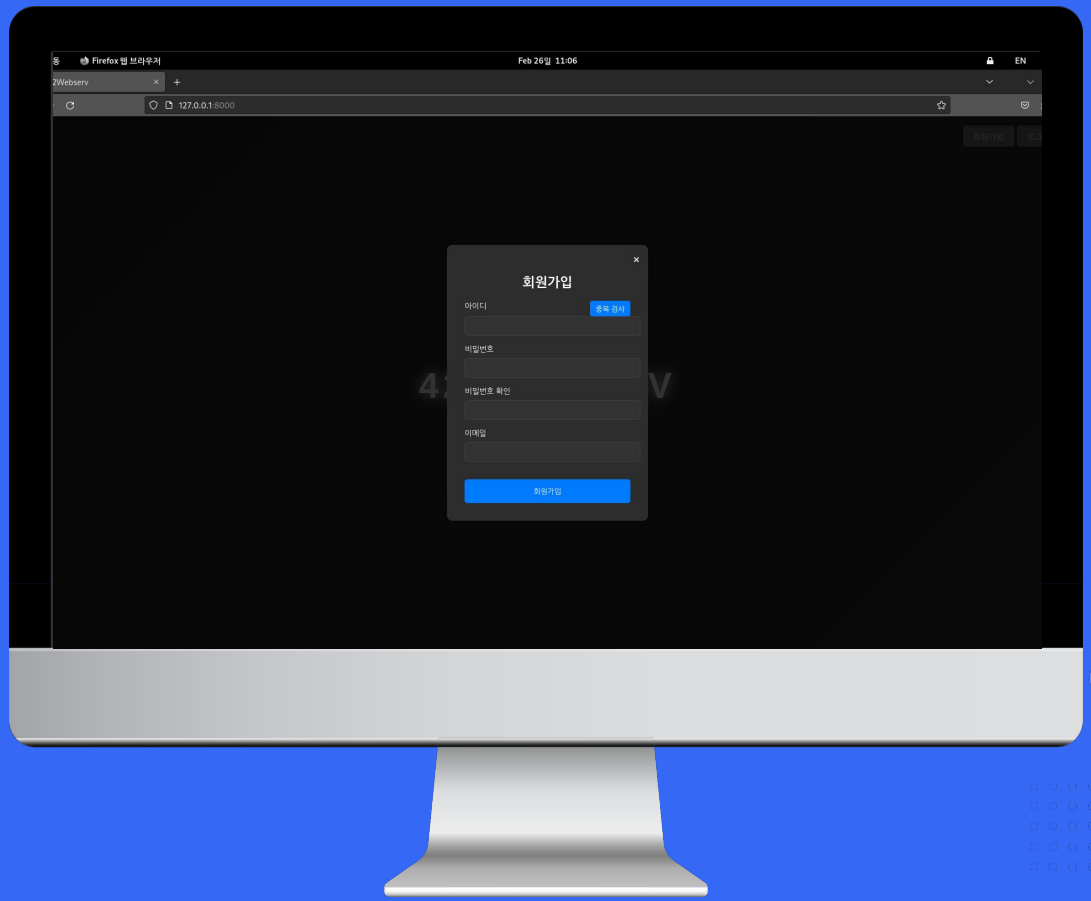
- Epoll 기반의 비동기 소켓 통신을 통해 고성능 정적 파일 서빙 기능을 구현했습니다. Epoll을 사용하여 다수의 클라이언트 요청을 효율적으로 처리하고, non-blocking I/O를 통해 응답 시간을 최소화했습니다. HTTP/1.1 프로토콜을 준수하며, keep-alive 연결을 지원하여 클라이언트와의 연결을 유지하고 재사용합니다.

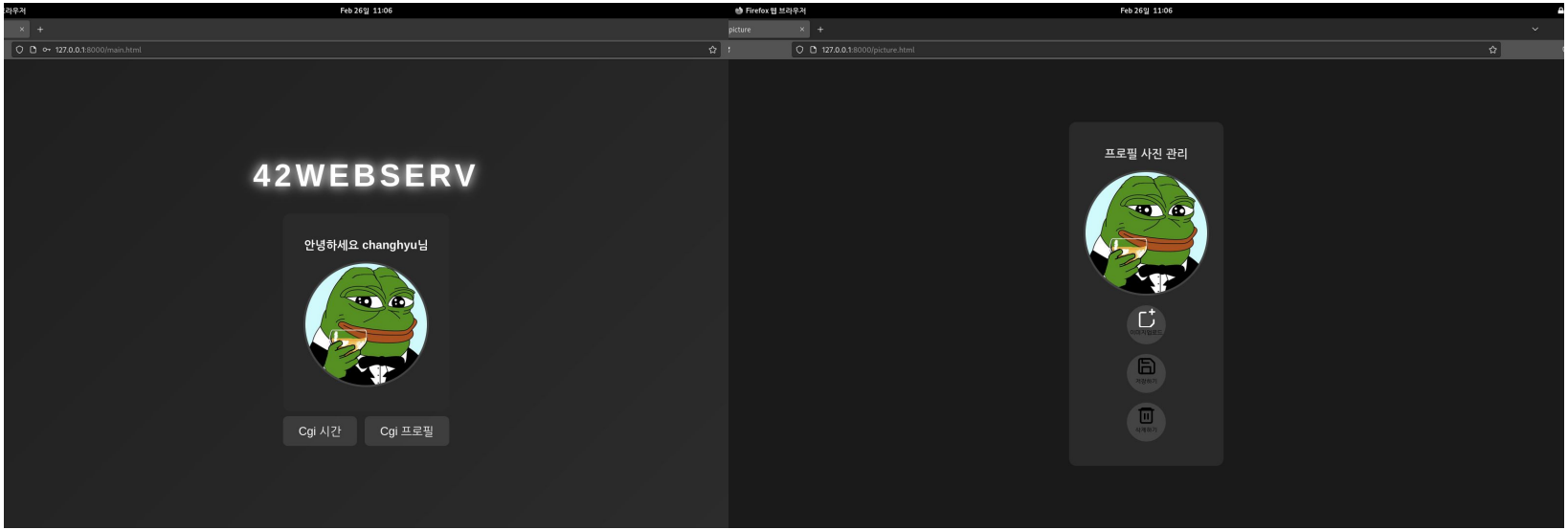
42_WEBSERV

프로젝트 소개

SIGN UP

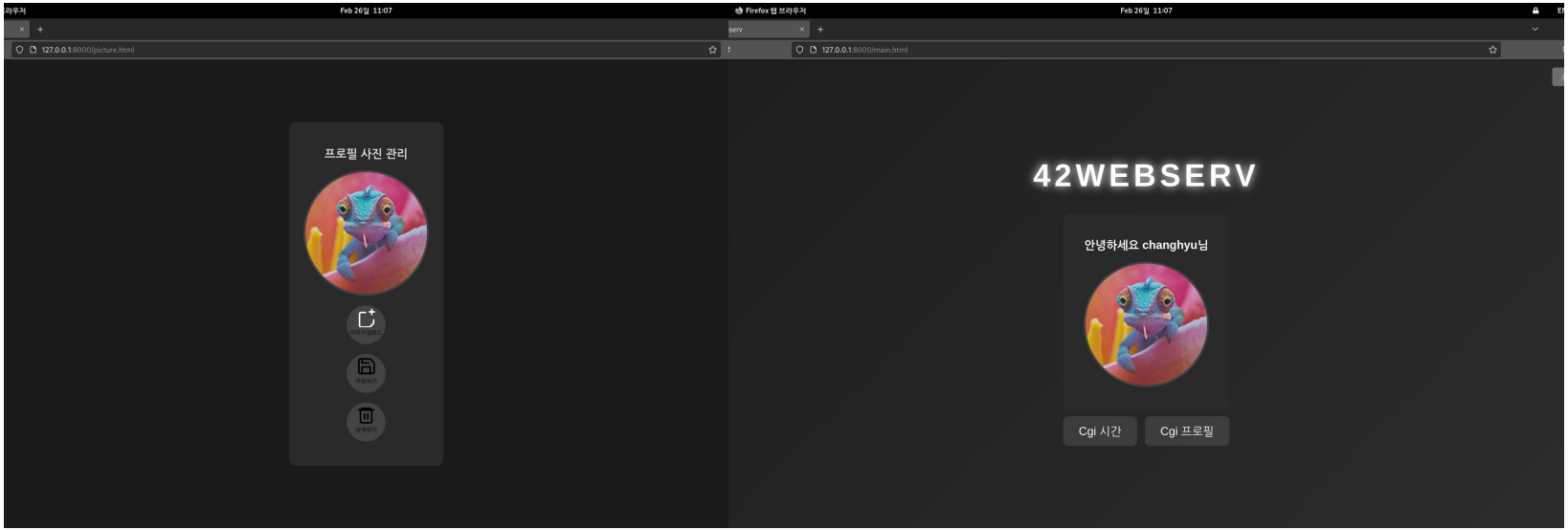
- 외부 라이브러리 및 데이터베이스 없이 JSON 파일을 사용하여 회원 정보를 관리합니다. 회원 가입, 로그인, 중복 검사 등의 기능은 CGI 스크립트(Python)를 통해 구현했습니다. 보안 강화를 위해, 회원 가입 시 각 사용자에게 고유한 Salt 값을 생성하고, 패스워드를 해시화하여 저장합니다. Salt 값을 해시된 비밀번호에 추가하여 레인보우 테이블 공격에 대한 보안성을 높였습니다. JSON 파일 파싱 및 데이터 검증은 직접 구현했습니다.





Login

로그인 성공 시, 사용자에게 기본 프로필 이미지를 표시하고, 이미지 업로드, 현재 시간 확인 (CGI), 로그아웃 기능을 제공합니다. 사용자 인증은 세션 기반으로 구현되었으며, 로그인 시 생성된 세션 ID는 쿠키를 통해 관리됩니다.



Profile Image

프로필 사진 업데이트 기능은 multipart/form-data 형식의 HTTP 요청을 통해 구현되었습니다. 클라이언트는 이미지를 선택하고, 서버는 전송된 데이터를 바운더리를 기준으로 파싱하여 파일 데이터를 추출합니다.

추출된 이미지 파일은 파일 확장자 및 크기 제한 등의 유효성 검사를 거친 후, 사용자 ID를 기반으로 생성된 디렉토리에 저장됩니다.

업로드 성공 시, 사용자 프로필 정보를 업데이트하고, 업데이트된 이미지를 화면에 표시합니다. 사용자는 저장 또는 삭제 버튼을 통해 프로필 이미지 변경을 확정하거나, 기본 이미지로 복원할 수 있습니다.

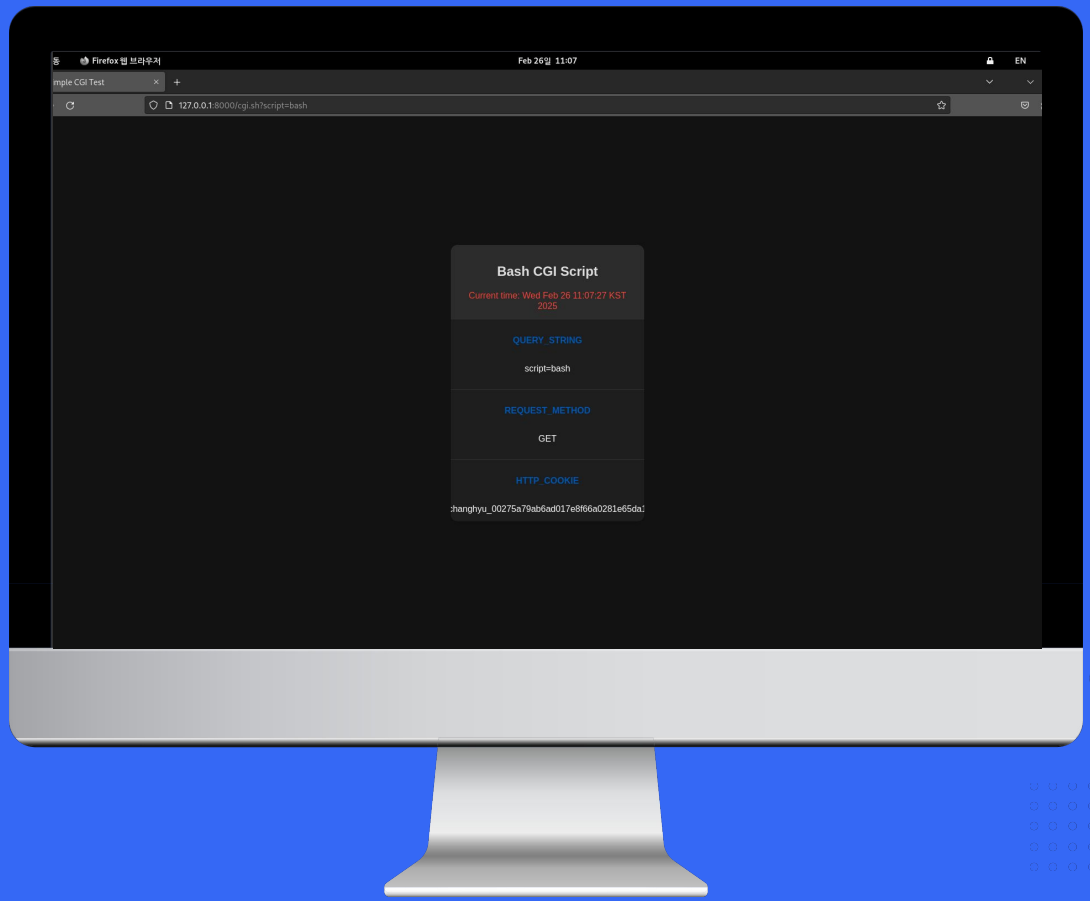
42_WEBSERV

프로젝트 소개

CGI TEST

현재 시간 확인 기능은 CGI (Common Gateway Interface)를 통해 구현되었습니다. 클라이언트는 HTTP GET 요청을 통해 CGI 스크립트를 실행하고, 서버는 스크립트 실행 결과를 HTTP 응답으로 반환합니다. CGI 스크립트는 현재 시간을 HTML 형식으로 출력합니다.

- Query String: CGI 스크립트 실행 시 전달되는 파라미터를 확인할 수 있습니다.
- Request Method: HTTP 요청 메소드 (GET, POST 등)를 표시합니다.
- HTTP_COOKIE: 클라이언트의 세션 ID를 확인할 수 있습니다. 세션 ID를 통해 사용자 인증 및 권한 관리를 수행합니다.



본 프로젝트를 통해 C++ 언어를 사용하여 저수준 소켓 통신부터 HTTP 프로토콜 구현, 그리고 웹 서버의 핵심 기능들을 직접 개발하는 과정을 경험하며 네트워크 프로그래밍에 대한 깊이 있는 이해를 얻을 수 있었습니다.

Epoll 기반의 I/O 멀티플렉싱을 통해 높은 동시성을 확보하고, HTTP 요청을 직접 파싱하여 응답을 생성하는 과정을 통해 웹 서버의 동작 원리를 명확하게 이해할 수 있었습니다. 특히, 데이터베이스 없이 JSON 파일 기반으로 회원 정보를 관리하고, CGI를 통해 동적인 콘텐츠를 생성하는 과정에서 시스템 아키텍처 설계 능력을 향상시킬 수 있었습니다.

다만, 본 프로젝트는 학습 목적으로 개발되었기 때문에, 상용 웹 서버에 비해 기능, 성능, 보안 등 여러 면에서 부족한 점이 있습니다. 향후에는 WebSocket을 구현하여 실시간 통신 기능을 추가하고, HTTPS를 지원하여 보안성을 강화하며, 데이터베이스 연동을 통해 데이터 관리 효율성을 높일 계획입니다.

INSIDE-OUT

🗓️ (24.06 - 24.07)

AWS Cloud Development Kit

Python을 사용하여 AWS 인프라를 코드로 정의하고 프로비저닝하는 데 사용했습니다. IaC (Infrastructure as Code)를 구현하여 인프라 관리 자동화를 구축했습니다.

AWS S3 Bucket

오디오 파일 및 기타 정적 파일을 저장하기 위해 사용했습니다.

AWS DynamoDB

오디오 파일 정보 (파일명, 라벨 등)를 저장하기 위한 NoSQL 데이터베이스로 사용했습니다.

AWS Lambda

서버리스 함수를 실행하기 위해 사용했습니다. CAPTCHA 처리, 더미 데이터 생성, 데이터베이스 업데이트, 학습 데이터 이동 등의 백엔드 로직을 구현했습니다.

AWS API GateWay

Lambda 함수를 호출하기 위한 REST API 엔드포인트를 생성하고 관리하는 데 사용했습니다.

AWS Amplify

웹 애플리케이션을 호스팅하고 배포하는 데 사용했습니다. GitHub 저장소와 연동하여 자동 배포 파이프라인 (Continuous Delivery)을 구축했습니다. 현재는 Main 브랜치에 푸시될 때마다 자동으로 배포되도록 설정되어 있으며, 향후 CI (Continuous Integration) 단계를 추가하여 코드 품질 검사 및 테스트 자동화를 구현할 계획입니다.

Python

AWS CDK 코드 및 Lambda 함수를 작성하는 데 사용했습니다.

INSIDE-OUT

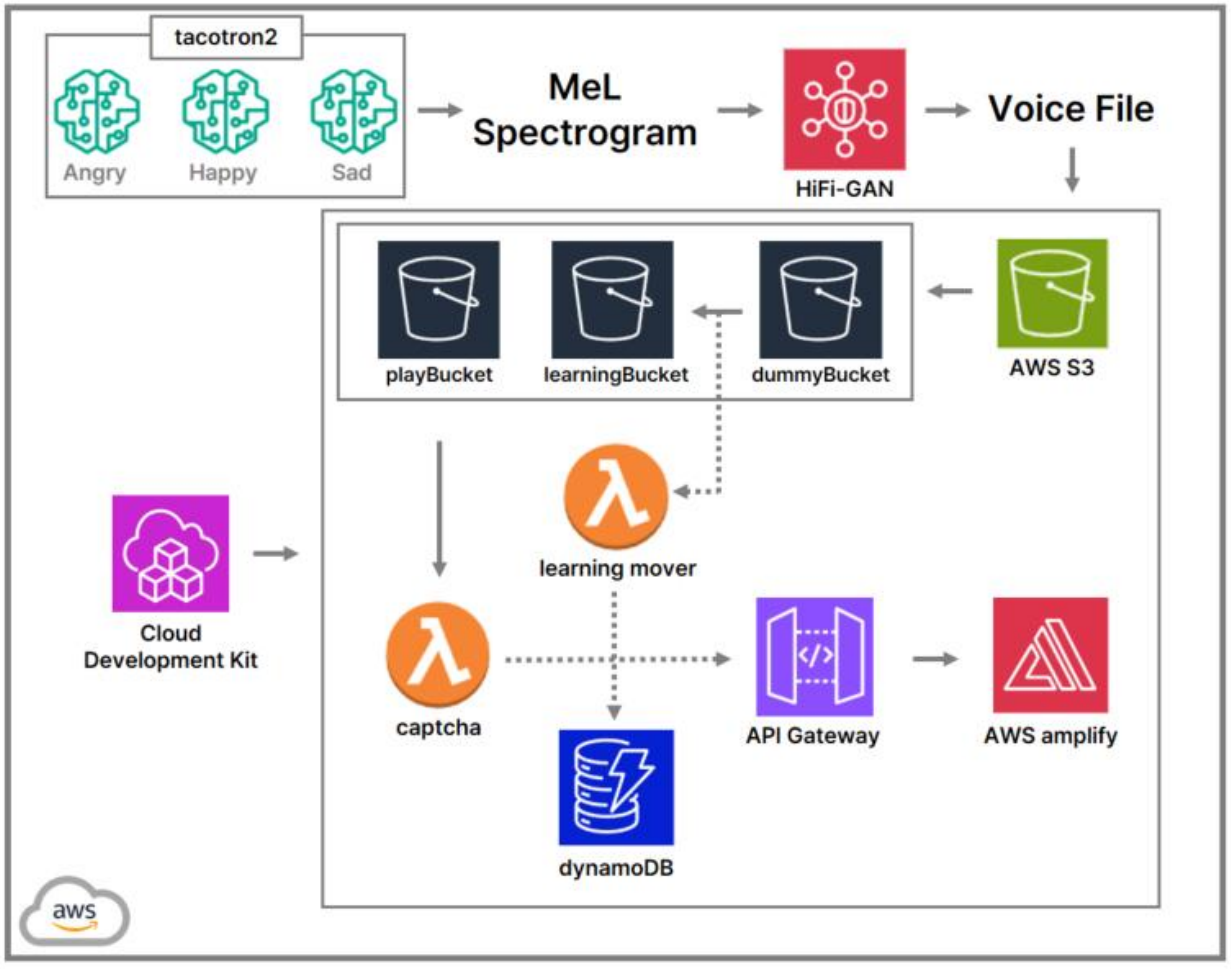
프로젝트 소개

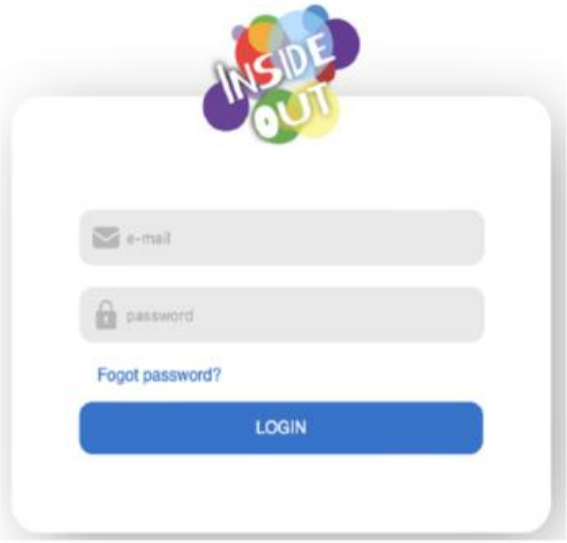
본 프로젝트는 기존 텍스트 및 이미지 기반 CAPTCHA 시스템의 낮은 방어율을 개선하기 위해, AI 기반 음성 CAPTCHA 시스템을 개발했습니다. 사용자는 감정이 담긴 음성 데이터를 듣고, 해당 감정을 올바르게 선택해야 CAPTCHA를 통과할 수 있습니다.

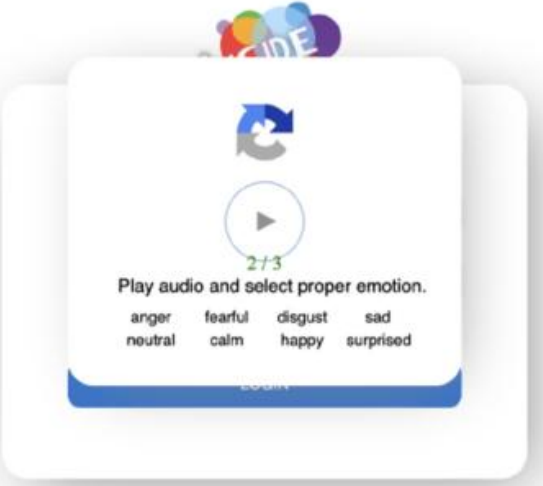
- 음성 데이터 활용: 감정이 담긴 음성 데이터를 활용하여 CAPTCHA의 난이도를 높이고, 사용자에게 더욱 자연스러운 경험을 제공하고자 했습니다.
- TTS (Text-to-Speech) 모델 파인튜닝: 봇이 쉽게 해독할 수 없는 다양한 음성 데이터를 생성하기 위해, TTS 모델을 특정 감정에 맞춰 파인튜닝했습니다.
- 더미 데이터 활용: AI 모델 학습을 위한 충분한 양의 데이터를 확보하기 위해, 긍정, 부정, 중립 등 다양한 감정을 표현하는 더미 데이터를 생성하고 활용했습니다.
- 데이터 라벨링 및 검증: 생성된 음성 데이터는 수동으로 라벨링하여 각 음성 파일에 감정 정보를 부여했습니다. 사용자가 선택한 감정과 미리 라벨링된 정답 감정을 비교하여 CAPTCHA 통과 여부를 결정합니다.

팀원

- ✔CHyuni (AWS CDK, AWS Amplify, AWS Lambda, API Gateway)
- 1107c (AWS CDK, AWS Lambda, AWS DynamoDB, API Gateway)
- 최성현 (AI모델 파인튜닝)


tacotron2
Angry
Happy
Sad
MeL Spectrogram
HiFi-GAN
Voice File
playBucket
learningBucket
dummyBucket
AWS S3
Cloud Development Kit
learning mover
captcha
API Gateway
AWS amplify
dynamoDB
aws


INSIDE OUT
e-mail
password
Fogot password?
LOGIN


2 / 3
Play audio and select proper emotion.
anger
fearful
disgust
sad
neutral
calm
happy
surprised

INSIDE-OUT

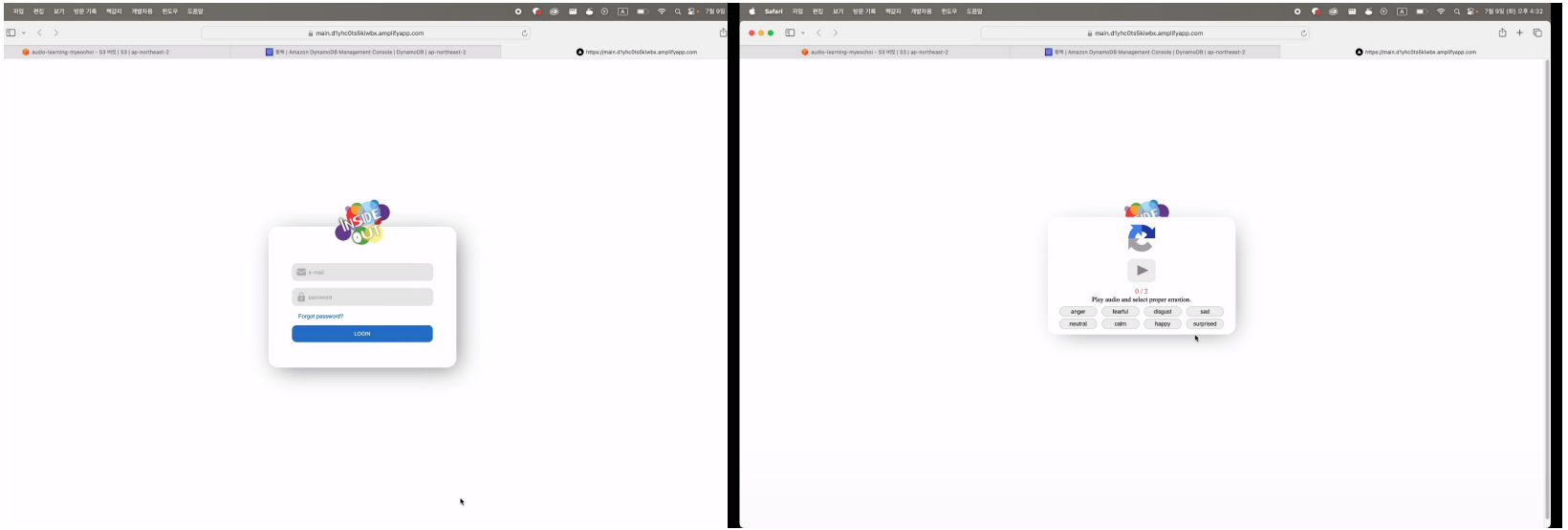
아키텍처 구성도

1. 감정 음성 생성 파이프라인:
   - 1.1. 파인튜닝 된 Tacotron2 모델을 활용하여 다양한 감정(행복, 슬픔, 분노 등)이 담긴 음성을 생성합니다.
   - 1.2. 생성된 음성은 MeL Spectrogram으로 변환됩니다.
   - 1.3. HiFi-GAN을 통해 고품질 음성 파일로 변환됩니다.
   - 1.4. 최종 음성 파일은 AWS S3에 저장됩니다.

2. 데이터 관리 시스템:
   - 2.1. dummyBucket (S3): 라벨링이 필요한 검증용 음성 파일 저장
   - 2.2. learningBucket (S3): 검증 완료되어 학습에 사용될 음성 파일 저장
   - 2.3. playBucket (S3): CAPTCHA 실제 인증에 사용되는 음성 파일 저장
   - 2.4. Lambda 기반의 'learning mover' 함수가 일정 조건을 만족한 데이터를 dummyBucket에서 learningBucket으로 자동 이동시킵니다.

3. 인프라 구축 및 관리:
   - 3.1. AWS Cloud Development Kit(CDK)를 사용하여 전체 인프라를 코드로 관리합니다.
   - 3.2. DynamoDB에 오디오 파일 메타데이터 및 사용자 응답 데이터를 저장합니다.

4. 사용자 인터페이스 및 인증 흐름:
   - 4.1. AWS Amplify를 통해 웹 애플리케이션을 호스팅합니다.
   - 4.2. 사용자가 로그인을 시도하면 CAPTCHA 인증 화면이 표시됩니다.
   - 4.3. 재생 버튼을 통해 음성을 듣고 8가지 감정(화남, 두려움, 혐오, 슬픔, 중립, 차분함, 행복, 놀람) 중 적절한 감정을 선택합니다.
   - 4.4. API Gateway를 통해 Lambda 함수와 통신하여 사용자 선택을 검증합니다.



Login

웹 페이지에 접속하면 먼저 홈 화면이 표시됩니다.

로그인을 시도하면 음성 CAPTCHA가 나타납니다. 사용자는 재생 버튼을 눌러 감정이 담긴 음성 텍스트를 듣고, 해당 감정을 8가지 보기 중에서 선택해야 합니다. 오답을 선택할 경우, 총 3번의 기회가 주어지며, 3번 모두 실패할 경우 봇으로 판단합니다.

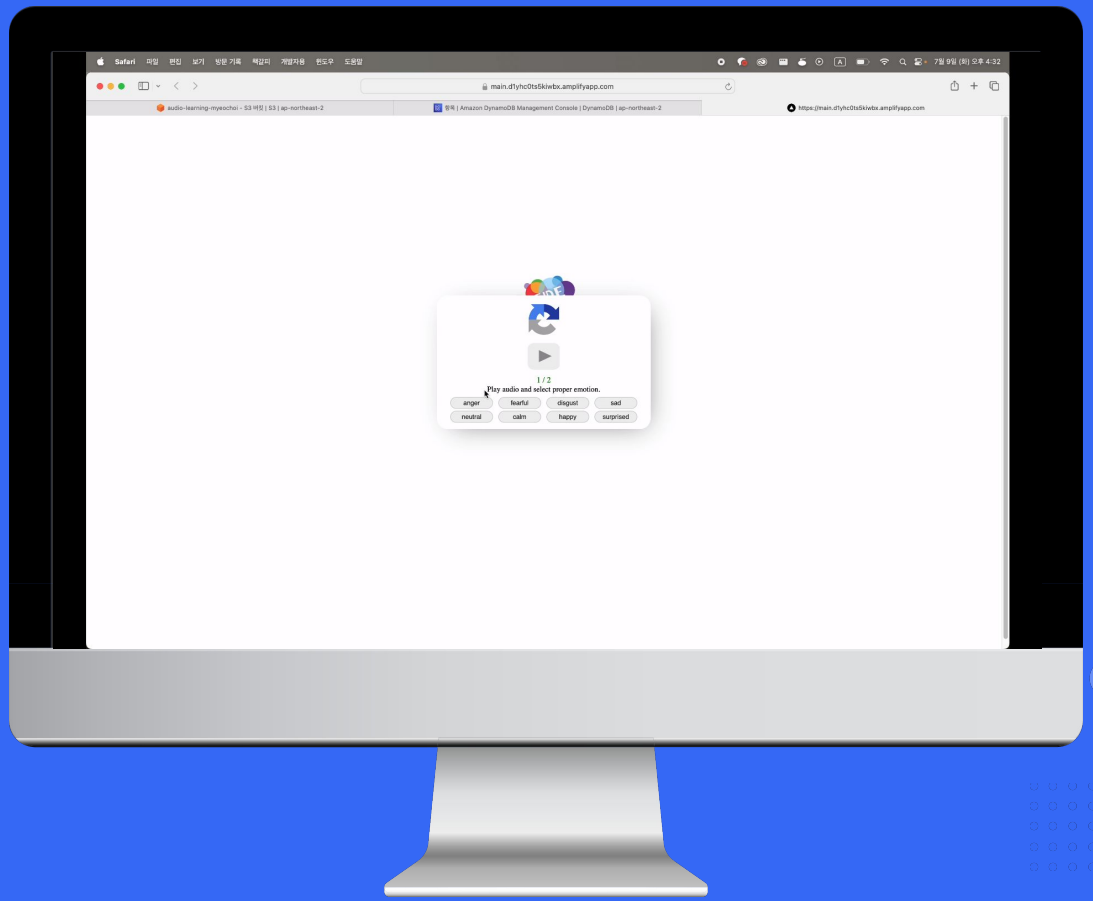
INSIDE-OUT

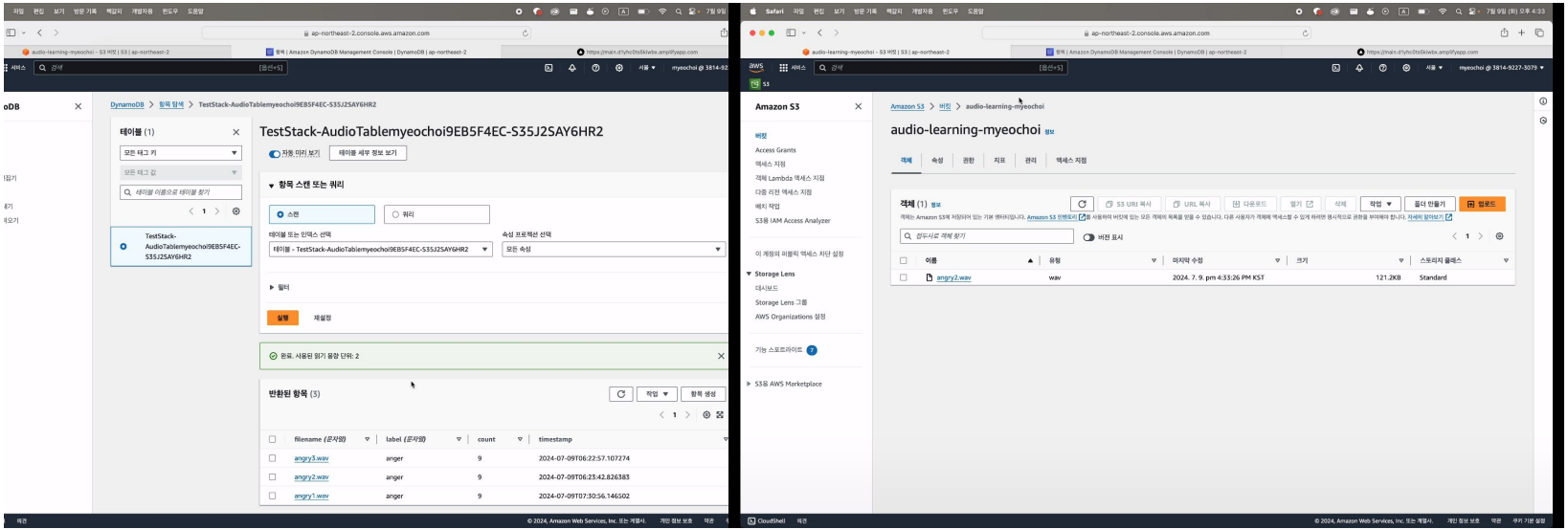
프로젝트 소개

DUMMY-DATA

- 2번째 문제에서는 검증 데이터를 사용하여 난이도를 높입니다. 사용자는 해당 음성이 어떤 감정을 담고 있는지 라벨링하도록 유도됩니다.

- 해당 데이터의 라벨링 유도를 통해 추후 학습데이터를 수집할 수 있습니다.





Dummy-data

검증 데이터 DB (DynamoDB)에는 각 음성 파일이 특정 감정으로 라벨링된 횟수가 저장됩니다.

하나의 음성 파일에 대해 특정 감정 라벨의 수가 다른 감정 라벨 수보다 현저히 높을 경우 (예: 10회 이상), 해당 음성 파일은 해당 감정으로 라벨링된 것으로 판단하고, 학습 예정 데이터 DB로 이동됩니다. 학습 예정 DB에 일정량의 데이터가 쌓이면, TTS 모델 파인튜닝을 진행하고, 해당 데이터는 삭제됩니다. 이를 통해 사용자의 참여를 통해 데이터를 수집하고, CAPTCHA 시스템의 정확도를 높일 수 있습니다.

본 프로젝트는 기존 CAPTCHA 시스템의 한계를 극복하고자 음성 기반 CAPTCHA 시스템을 개발하고 AWS 클라우드 환경에 구축하는 과정을 성공적으로 수행했습니다. AI 모델 파인튜닝, IaC 기반 인프라 구축, 자동 배포 파이프라인 구현 등 다양한 기술적 도전을 통해 이론적인 지식을 실제 서비스에 적용하는 경험을 얻었습니다.

특히, 다음 사항들을 통해 프로젝트의 가치를 더욱 높일 수 있었습니다. AI 모델 활용 및 AWS 클라우드 서비스 연동: TTS 모델 파인튜닝을 통해 생성된 음성 데이터를 AWS S3 버킷에 저장하고, Lambda 함수와 API Gateway를 통해 CAPTCHA 인증 로직을 구현함으로써, AI 모델의 결과물을 클라우드 서비스와 연동하여 활용했습니다.

사용자 참여형 데이터 수집 시스템 구축: CAPTCHA를 통해 수집된 사용자 라벨링 데이터를 활용하여 AI 모델을 지속적으로 개선하는 선순환 구조를 구축했습니다. 이는 CAPTCHA 시스템의 정확도를 높이고, 장기적인 유지보수 및 확장에 기여할 것입니다.

IaC 기반 인프라 구축 및 자동 배포 파이프라인 구현: AWS CDK를 사용하여 인프라를 코드로 관리하고, AWS Amplify를 통해 **자동 배포 파이프라인 (Continuous Delivery)**을 구축함으로써, 인프라 관리 효율성을 높이고, 배포 프로세스를 자동화했습니다.

이번 프로젝트를 통해 얻은 경험과 기술적 노하우는 앞으로 클라우드 기반 AI 서비스 개발 및 운영에 있어 큰 자산이 될 것입니다. 향후 CI/CD 파이프라인 구축, 보안 강화, 사용자 인터페이스 개선, 악성 봇 방지를 위한 음성 데이터에 미세한 변조와 같은 퍼즐 추가 등 지속적인 개선을 통해 시스템 완성도를 높여나갈 계획입니다.

THANKS !

changhyu953@gmail.com
apple141410@naver.com
Github PortFolio