기술문서

Github: <https://github.com/leh60245/recognize-piano-pitch-sound>

# 코드 구조

프로젝트 디렉토리 구조와 각 디렉토리의 역할.

App을 구성하는 기능들은 src/routes 폴더 안에 있습니다.

src  
 ┣ routes  
 ┃ ┣ AudioStreamer.jsx  
 ┃ ┣ setting.jsx  
 ┃ ┣ sheet-image-path.json  
 ┃ ┗ SheetMusicSelectorPage.jsx

## AudioStreamer.jsx

유저가 메인 화면에서 “악보 연습” 버튼을 클릭하면 보게 되는 기능들의 코드이다. 피아노 음표를 소리로 인식하고 백엔드로 데이터를 보내준다. 백엔드에서 분석된 음표 데이터가 오면 현제 음표와 비교하여 동일한지 비교하여 유저에게 화면으로 보여준다.

## Setting.jsx

개인 설정 기능을 추가할 파일이다. 현재 배경화면 설정 기능이 있다.

## Sheet-image-path.json

이미지 파일 이름 저장

## SheetMusicSelectiorPage.jsx

악보 선택 매뉴

# 주요 클래스 및 함수:

  useEffect(() => {

    if (waveformRef.current && !wavesurfer.current) {

…

      });

    }

    return () => {

…

    };

  }, []);

WaveSurfer library를 사용하여 실시간으로 음정을 인식되고 있음을 화면에 띄웁니다.

  useEffect(() => {

    ws.current = new WebSocket('ws://localhost:5000');

…

  }, []);

WebSocket을 열어 백엔드와 소리 데이터를 주고 받습니다.

  const startCountdown = (resume = false) => {

    setIsCountingDown(true);

    setCountdown(4);

…

  };

소리 인식을 시작하거나 다시 시작할 때 대기시간을 적용합니다. setCountdown 함수를 사용해 대기시간을 정할 수 있습니다.

  const beginRecording = async () => {

    setIsRecording(true);

    setIsPaused(false);

    setShowRepeatPrompt(false); // 숨기기

    setIncorrectNotes({}); // 잘못된 노트 초기화

…

  };

소리 인식을 시작하는 함수입니다. 함수 내부에는 마이크를 통해 인식된 소리 데이터를 벡엔드로 보내는 기능이 있습니다.

  const detectSound = (audioData) => {

    const threshold = 0.01; // 소리 인식 임계값

    isSoundDetected.current = audioData.some(sample => Math.abs(sample) > threshold);

  };

  const sendAudioData = (audioData) => {

    if (ws.current && ws.current.readyState === WebSocket.OPEN && audioData.length > 0 && !isWaiting.current && isSoundDetected.current && !showRepeatPrompt) {

      const float32Buffer = new Float32Array(audioData);

      ws.current.send(float32Buffer.buffer);

    }

  };

소리 데이터의 임계값을 설정해, 그 이상의 소리를 반환하는 함수와, 소리 데이터를 Websocket을 통해 백엔드로 전송하는 함수입니다.

  const pauseRecording = () => {

…

  };

  const resumeRecording = () => {

…

  };

  const stopRecording = () => {

…

  };

녹음 버튼을 구현한 함수로 일시정지, 다시 시작, 정지가 있습니다.

  const drawNote = () => {

…

};

악보 위에 요소들을 그리는 함수입니다. 주어진 악보 이미지 위에 canvas 기능을 사용하여 그릴 수 있습니다. 내부에 구현된 기능들은 백엔드로부터 분석된 음표가 현재 음표와 일치하는지 아닌지 구분하여 주어진 음표의 좌표에 빨간 표시를 하고 해당 음표를 저장하거나 파란 표시를 하고 다음 음표로 넘어갑니다.

  const handleRepeat = () => {

…

  };

악보 연습을 다시 시작할 때 해당 함수가 호출이 되어 저장된 데이터를 초기화 합니다.

# 알고리즘 설명

주요 알고리즘은 크게 세가지 있습니다.

## 녹음 시작

const beginRecording = async () => { … }

해당 함수는 녹음을 시작하거나 다시 시작할 때 호출되는 함수로 내부에는 “navigator.mediaDevices.getUserMedia”로 소리 데이터를 가져와서 데이터를 처리하고, 버퍼 형식으로 데이터를 백엔드로 전송합니다.

## 음표 비교

const compareNotes = (backendNote, note) => { … }

벡엔드에서 온 음표 데이터와 현제 음표의 데이터를 비교해 true나 false를 제공합니다.

## 악보 위에 요소 그리기

const drawNote = () => { … }

주어진 악보 이미지 위에 canvas 기능으로 필요한 요소들을 그립니다. 백엔드에서 받은 음표 데이터와 비교해 같다면 파란색 불투명 박스를, 다르면 붉은색 불투명 박스를 canvas에 그립니다.

# 확장/유지보수 지침

1. 악보에 그려진 음표 데이터가 현재 json 형식으로 코드 내부에 구현되어 있습니다. 또한 각 음표 데이터는 음정과 좌표 데이터로 구성되어 있는데, 이는 추후에 음표 데이터를 통일하여 따로 json 파일 형식으로 저장을 해두어야 합니다.
2. 모든 함수들이 하나의 파일에 구현되어 있어 확장성이 떨어집니다. 이후 추가될 기능들은 파일들을 구분하여 추가해야 합니다.