**시작**

안녕하세요 6조 중간발표를 맡은 임채윤입니다. 발표 시작하겠습니다.

**목차**

발표 순서는 목차와 같이 프로젝트 개요, 팀원 별 역할, Front, Back, AI 별로 프로젝트 진행 현황, 최종 목표 및 향후 계획 순으로 진행할 예정입니다.

**프로젝트 개요**

저희 6조는 ‘기분어때’라는 이름의 ‘AI 기분인식 감정 캘린더 APP’ 을 ReactNative와 Spring, Flask 환경에서 개발하고 있습니다.

저희 앱은 감정을 포함한 일기를 작성하고자 하는 사용자라면 남녀노소 사용이 가능한데, 감정일기를 작성하는 것이 스트레스를 관리하는 데에 큰 도움이 되는 만큼, 스트레스가 많은 현대인들에게 많은 도움을 줄 수 있으리라 생각합니다.

**구현목표 디자인**

Figma로 디자인한 ‘기분어때 APP‘의 완성 목표 모습입니다.

순서대로 앱 로고, 얼굴인식 페이지, 기분 확정, 명언 페이지 입니다. 얼굴인식 페이지의 해당 중앙부분은 사용자의 전면카메라가 들어갑니다.

**(클릭)** 메인 페이지의 디자인입니다. 아래에는 ‘홈, 음악 추천, 채팅방’ 세개의 탭이 존재합니다. 우선 홈 탭을 살펴보면 캘린더와 함께 사용자가 작성한 일기를 보여줍니다. 일기를 작성할 수도 있고, 수정 및 삭제도 가능합니다.

또한 캘린더 상단의 버튼에는 기분을 재촬영할 수 있는 ‘기분어때’ 버튼과 일주일 단위의 일기요약 기능인 ‘AI일기요약’ 버튼이 존재합니다. 일기가 요약되면 해당 일기의 요약과 함께 AI의 한줄평을 얻을 수 있습니다.

**(클릭)** 음악 추천, 채팅창 탭입니다. 음악 추천 탭에서는 감정에 어울리는 음악 장르를 선택할 수 있습니다. 장르를 선택하면, 해당 장르에 맞는 음악을 AI가 추천해줍니다.

채팅창 탭에서는 같은 감정을 가진 사용자들이 함께 대화할 수 있습니다.

**팀원 별 역할**

다음은 '기분어때'APP에서의 팀원 별 역할입니다.  
진정준 학생은 Frontend 역할을 맡았고, => 글귀 페이지와 앱 초기 로딩화면, 캘린더와 일기가 있는 메인 탭을 구현하였습니다.  
저는 FullStack 역할을 맡아 Frontend와 Backend를 함께 수행하였는데요, => Figma, Notion을 통한 프로젝트 관리, AI서버에 전달할 얼굴 인식 페이지 및 분석 결과 페이지(Front)를 구현하였고, OpenAI 프롬프트 엔지니어링을 활용한 AI일기요약, AI일기한줄평 기능을 구현하였습니다. 그리고 OpenAI와 자바 Jsoup 크롤링을 활용한 음악 추천탭 구현(Front&Back), 유저 및 댓글 관리를 포함한 채팅창 탭 구현(Front&Back)을 맡았습니다.  
마지막으로 최민석 학생은 AI 역할을 맡았고, => 감정 분석을 위한 모델 선정, 데이터 수집 및 전처리, 모델 파인튜닝 및 평가, AI서버 구축 및 배포, 지속적 모델 개선을 맡았습니다.

우선 Frontend 진행현황에 대해서 말씀드리겠습니다.

**Expo Go, VS Code 연결**

저희 팀 프론트엔드들은 휴대폰의 Expo Go 앱과 노트북의 VS Code를 연결하여 개발을 진행했습니다. 우선 개발 전 백엔드 서버와 연결을 테스트하기 위해

**(클릭)** axios 기능을 활용해 서버에 POST 요청을 넣어보고,

**(클릭)** 정상적으로 값이 전송되는 것을 확인한 후 개발을 시작하였습니다.

**글귀 페이지**   
저희 앱은 얼굴인식이 완료된 후, 해당하는 감정에 맞는 글귀를 보여줍니다.  
(클릭)좌측 사진과 같이 코드를 작성하여 우측에 있는 글귀 페이지를 만들었습니다. 저희 앱은 이미 오늘의 감정을 기록한 상태여도 사용자가 원한다면 기분을 재촬영 할 수 있는데요, 이 때 이미 오늘의 감정이 등록되어 있다면  
(클릭) 다음과 같은 모달 창을 띄워 감정을 업데이트해서 저장할 것인지, 그렇지 않을 것인지 선택할 수 있도록 하였습니다. 하단의 Show Modal 버튼은 테스트용 버튼으로 추후 삭제할 예정입니다.

**앱 초기 로딩화면**이 부분은 앱이 실행될 때의 초기 로딩화면입니다. 아직 명확한 로고 디자인을 한 것은 아니라서, 크게 틀만 만들어둔 상태입니다.

**메인 홈 화면  
(클릭 두번)**  
다음은 메인 홈 화면을 구성하는 코드입니다. 해당 코드들을 통해  
**(클릭)** 다음과 같은 화면을 구성했습니다. 윗 공간은 캘린더가 들어갈 자리이고, 아래 부분은 해당 날짜에 대한 일기를 보여줄 공간입니다. 위의 캘린더에서 선택한 날짜에 일기가 작성되어있지 않을 경우 저렇게 ‘일기 쓰기’라는 버튼이 보이게 됩니다. 지금 보시는 화면은 ios 환경이고  
**(클릭)** 해당 화면은 안드로이드 환경입니다. 이처럼 iOS, Android 두 플랫폼 모두 잘 반영되는 것을 보실 수 있습니다. 또한 일기 쓰기 버튼을 클릭할 경우  
**(클릭)** 이렇게 일기를 작성할 수 있는 화면이 띄워지고, 일기를 작성한 뒤 작성 완료 버튼을 눌러 일기를 작성할 수 있습니다.  
**(클릭)** 일기 작성이 완료된 후에는 이와 같이 사용자가 작성한 일기의 내용을 보여줍니다.  
**메인 홈 화면 – 캘린더  
(클릭 천천히 5번 하면서)**  
지금 보시는 코드는 메인 탭의 캘린더를 구현하기 위한 코드입니다. 리액트 네이티브에서 제공하는 캘린더 라이브러리가 있지만 커스텀 디자인에는 제한이 따른다는 단점이 존재했고,   
**(클릭)** 따라서 저희는 이렇게 캘린더에 해당 일자의 기분을 이모지로 나타내려 하기 때문에 캘린더를 새롭게 제작하였습니다. 현재까지는 캘린더에 이모지를 넣을 수 있는지 까지 구현한 상태입니다.

**얼굴 인식 페이지**위 사진은 실제 Frontend에서 구현한 얼굴인식페이지 입니다. 우선 카메라 권한에 동의하면 사용자 기기의 카메라를 사용할 수 있습니다. AI가 감정을 잘 분류하려면, 어느정도 일관적인 사진을 전송할 필요가 있었는데요, 따라서 위와 같은 가이드라인을 통해 사용자가 어느정도 일관된 위치에서 촬영할 수 있도록 유도하였습니다.  
얼굴인식을 위해 사용자의 얼굴을 캡쳐해야 했는데, 사용자가 버튼을 눌러 촬영하는 ‘직접촬영’ 방식과 얼굴을 인식하여 자동으로 촬영해주는 ‘자동촬영’ 방식 중 저희 6조는 자동촬영 방식을 선택했습니다.  
자동 촬영을 위해서는 사용자의 얼굴 위치를 파악하고, 해당 가이드라인에서 벗어나는지 아닌지를 파악해야 했는데요,  
ExpoCamera 라이브러리에 있는 onFacesDetected라는 이벤트 속성이 있는데, 해당 이벤트에서 사용자의 얼굴부위 벡터값을 반환하는 것을 보고, 그 값을 이용해 가이드라인 내에 얼굴이 위치하는지 boolean여부를 반환했습니다.  
먼저, 얼굴 벡터가 존재하는지 확인하고, 세가지 조건을 만족하는지 확인했습니다.  
첫 번째, 얼굴 크기를 측정하는 기준인 얼굴 높이가 200~280 사이인 경우  
두 번째, 얼굴이 가로로 중앙부에 존재하는지 확인하기 위해 코의 x위치가 140~230 사이인지 확인하고,  
세 번째, 얼굴이 세로로 중앙부에 존재하는지 확인하기 위해 코의 y위치가 180~310 사이인지 확인했습니다.  
사람마다 얼굴부위가 모두 다른 위치에 존재할 수 있기에, 가장 공통적으로 적용되는 얼굴 높이, 코 위치를 통해 가이드라인 내에 얼굴이 존재하는지 확인했습니다.

**얼굴인식페이지 (다양한 실패케이스**위 사진들의 경우엔 다양한 실패 케이스인데요, 첫번째는 너무 얼굴을 멀리한 경우, 두번째는 너무 가까이한 경우, 세번째, 네번째, 다섯번째는 가이드라인을 벗어난 상태로 촬영을 하려는 경우입니다. 이전 슬라이드에서 설명한 조건에 부합하지 않는 경우 이와같이 촬영이 되지 않는것을 볼 수 있습니다.

**얼굴인식페이지 (성공 케이스)**다음은 실제로 카운트다운이 성공하여 촬영을 완료한 경우입니다. 위 예시는 차례대로 무표정, 놀람, 행복, 통곡, 격노 라는 감정들이고 해당 감정이 본인의 현재 감정이 맞으면 ‘맞아요‘ 버튼을 눌러서 감정에 따른 글귀 페이지로 이동할 수 있고, 아니라면 ‘아니에요’ 버튼을 눌러 다시 얼굴 인식 페이지로 돌아가서 재촬영할 수 있습니다.  
**(클릭)** 실제로 어떤 방식으로 작동하는지 영상을 찍어서 준비했습니다. 카메라에 얼굴이 인식되면 3, 2, 1과 같은 카운트 다운이 자동으로 나타납니다. 또한 중간에 가이드라인을 벗어나거나 너무 멀리서 촬영하는 등, 촬영조건을 만족하지 못한다면 영상과 같이 카운트다운이 사라지며 초기화 됩니다. 해당 영상은 얼굴 인식을 통해 무표정을 감지할 수 있었습니다.  
**AI 음악추천**  
다음 소개할 페이지는 AI음악추천 기능입니다. 현재 사용자의 감정이 ‘행복한’ 이라는 감정일 때의 경우로 테스트하였습니다. AI 음악추천을 받기 전, 사용자의 추천받을 음악에 대한 키워드를 직접 선택할 수가 있는데요, 슬픈 감정일 때 기운이 솟는 노래도 좋지만 때로는 슬픈 노래를 듣는걸 선호하는 사용자도 있기에 이처럼 사용자에게 선택지를 주어 더욱 맞춤형 음악을 추천 받을 수 있도록 구현하였습니다.  
**(클릭)** 실제 어떤 방식으로 동작하는지 영상을 준비했습니다.  
**감정 채팅방**  
다음 소개할 페이지는 감정 채팅방입니다. 해당 채팅창은 같은 감정을 가지고 있는 사용자들끼리 대화할 수 있는 공간을 제공합니다. 사용자는 첫 접속시 닉네임을 등록합니다. 닉네임 등록이 성공적으로 이루어졌다면 다른 사용자와 대화할 수 있게됩니다. 또한 본인의 댓글을 꾹 눌러 삭제할 수도 있습니다.  
해당 채팅창은 기능은 페이징 기능을 통해 너무 많은 양의 댓글이 불러와지지 않도록 하였고, 실시간 조회를 위해 화면에 나타날때 Listiner를 통해 subscribe하고, 채팅창을 벗어나면 unsubscribe하는 방식을 통해 실시간으로 댓글을 조회할 수 있도록 하였습니다.

다음은 Backend 진행현황에 대해서 말씀드리겠습니다.  
**OpenAI API**  
저희 6조는 Backend언어로 스프링을 사용하였습니다. 백엔드에서 크게 구현해야할 사항은 일기요약, 일기한줄평, 음악추천에 사용되는 OpenAI API 활용 기능, 채팅창을 위한 사용자 관리, 추천 음악 크롤링 기능, 채팅창 관리 이렇게 크게 네가지였습니다. 그 중에서 먼저 OpenAI api 구현에 대해서 말씀드리겠습니다. OpenAI를 사용하기 위해 토큰을 결제한 뒤, Organization을 생성하고 API키를 발급받았습니다.   
그리고 해당 값들을 스프링 빈에 등록한 뒤, OpenAiConfig라는 것을 통해 RestTemplate을 정의하여 OpenAI 사용 준비를 마쳤습니다.Backend 기능에 활용한 LLM 모델은 우리에게 가장 익숙한 모델인 ‘gpt-3.5-turbo’ 모델을 사용했습니다. 이렇게 가져온 LLM모델을 솔루션을 안내하여 원하는 결과를 생성하는 프로세스 과정인 “Prompt Engineering”을 통해 원하는 값을 가져오도록 설정하였습니다.   
또한 OpenAI의 경우에 temperature이라는 값을 조절하여 AI의 창의력을 조절할 수 있는데요, 저는 이 값을 0으로 설정하여 주어진 명령에 맞는 결과만 반환하도록 초기설정을 진행했습니다.  
**AI일기 요약**  
이제, OpenAI를 활용해 구현한 백엔드 기능들에 대해 설명해드리겠습니다.  
일기요약은 일주일 단위로 이루어지는데, 일기가 존재하지 않거나 너무 적은 양의 일기, 예를들면 3개 미만의 일기가 입력되면 실패메세지를 반환하도록 설정했습니다.  
Frontend에게 건네받은 일기 배열을 Dto를 통해 AI에게 전달해주면, Prompt Engineering에 의해 주요사건을 중심으로 최소 70자의 요약메세지를 반환합니다. POSTMAN을 통해 테스트한 결과, 다음과 같이 성공적으로 요약된 데이터를 반환하는것을 확인할 수 있었습니다.

**AI일기 한줄평**  
일기 한줄평 기능은 바로 이전 슬라이드에 받았던 일기요약 메시지를 인자값으로 받아 진행하게 되는 기능입니다. 요약된 일기를 입력하면, Prompt Engineering에 의해 AI가 30자 정도의 상냥한 말투로 한줄평을 남겨줍니다. 마지막에 이모티콘도 넣어주도록 설정했습니다. POSTMAN을 통해 테스트한 결과, 다음과 같이 성공적으로 한줄평으로 답장하는 것을 확인할 수 있었습니다.  
**AI음악 추천**  
다음은 AI에게서 음악을 추천받은 뒤, Crawling을 시도하여 유튜브 썸네일과 유튜브 링크를 반환해주는 서비스입니다.  
**(클릭)** Java의 Jsoup 라이브러리를 활용해 코드를 작성하였고, Youtube 검색결과는 크롤링을 막는 기능이 포함되어있어, 낮은 성공확률을 보였기에  
이를 해결하고자 고민한 결과 Google 동영상 검색결과를 활용하여 Crawling 결과를 얻어냈습니다.  
테스트한 결과 우측 POSTMAN과 같이 각 노래의 title, url, image에 대한 정보를 얻어낼 수 있었고, 이를 Frontend가 열 수 있는 Object형태로 담아서 답변해주었습니다.  
**감성 채팅창**  
다음은 감정채팅창을 관리하는 기능을 구현한 화면입니다. Controller를 통해 채팅창에 댓글을 작성, 조회, 삭제할 수 있도록 구현하였고,  
**(클릭)** 해당 기능의 동작을 Service에 이와 같이 구현되어 있습니다. 댓글을 작성했을 때, 본인이 작성한 댓글을 확인할 수 있어야했기에 닉네임 등록을 활용하여 간단한 유저등록을 진행했습니다. User와 Comment에 대한 테이블이 MySQL을 통해 데이터베이스에 저장되어있고, 여기서 정의한 기능들을 통해 채팅창의 댓글을 관리할 수 있습니다. POSTMAN 테스트 결과,  
닉네임 등록 **(클릭)** 닉네임 수정 **(클릭)** 방명록 조회 **(클릭)** 댓글 작성   
**(클릭)** 댓글 삭제와 같이 모든 기능이 잘 작동하는것을 확인할 수 있었습니다.

마지막으로 AI 진행현황에 대해서 말씀드리겠습니다.

**Ai 모델 선정**컴퓨터 비전에서 이미지 처리를 하는 모델은 CNN와 VIT를 떠올리곤 하는데요,  
저희 6조는 VIT 모델을 선택했습니다. 이미지 데이터를 기준으로 하는 경우와, 사전 학습한 모델을 가지고 파인튜닝 하는 과정이 훨씬 유리했기 때문입니다.  
다음 사진은 구글에서 제공하는 vit모델로 1,400만 이미지와 12,843개의 클래스로 사전학습이 된 모델입니다.

**필요한 데이터 유형 조사**  
저희 6조는 감정의 종류를 나타낼 때 Pluchick이라는 학자가 제시한 원형 정서모델을 활용했습니다. 어떤 감정을 라벨링에 사용할지 논의한 결과 기쁨, 슬픔, 분노, 놀람, 무표정 5가지가 서로 뚜렷하게 구분되며 감정이 격해짐에 따라 기쁨은 행복, 분노는 격노, 슬픔은 통곡 같은 감정을 추가하여 8가지로 라벨을 부여하여 훈련했습니다. 훈련 데이터 셋의 경우엔, 동양인은 ai 허브에 있는 데이터 셋을 사용했고, 외국인 같은 경우는 캐글의 데이터를 사용했습니다.

**데이터 확인 (전처리)**  
이미지 데이터는 충분했으나 슬픈 표정과 무표정의 경계가 애매해서, 이미지를 재분류하는 과정을 겪었습니다. 슬픈 표정에서의 분류기준은 좌측 사진들을 참고 할 수 있습니다.

**데이터 확인 (전처리)**무표정의 경우도 재분류 한 예시 사진들입니다. 무표정과 슬픈 표정을 각 3천장 씩 분류했는데요, 기쁨, 화남, 놀람은 명확한 결과가 나타나기에 기존 데이터를 재 분류하지 않고 데이터 확인만 진행했습니다

**데이터 확인 (전처리)**다음 전처리 과정에서는, 배경을 포함한 원천 데이터를 얼굴만 보이도록 코드를 작성하여 이미지를 다시 저장했습니다. 코드의 주요내용은 분류기 모델을 가져와서 눈과 얼굴을 인식하고, 사각형 근처의 부분을 크롭하여 이미지를 저장하는 것입니다. 이와 같은 방법을 통해 다른 노이즈 요소를 배재하여 훈련데이터를 더욱 정확하게 만들 수 있었습니다.

**데이터 확인 및 학습 계획**데이터 확인은 다음과 같이 훈련 데이터와 평가 데이터를 나눠서 사용했습니다.  
학습은 다음과 같이 진행합니다. 먼저 동양인과 외국인의 사진 비율을 2:1로 훈련합니다. 원하는 수치가 나오지 않으면 추가로 학습을 진행하고 그래도 나오지 않는다면 파라미터를 수정하여 최선의 하이퍼 파라미터를 찾을 계획이었습니다. 마지막으로 학습 도중 결과값인 f1-score가 0.8이 넘으면 종료합니다.

**모델 파인 튜닝 코드**이미지 경로와 이미지 라벨링 값을 csv파일로 저장했고, 이를 통해 이후 훈련을 시킬 때 있어서 더욱 빠르게 이미지를 읽을 수 있도록 하였습니다.  
주요 파라미터와 훈련 과정을 살펴보자면, 먼저 주요 파라미터는 오른쪽 표를 통해 확인하실 수 있습니다. 총 15,000장을 훈련하였고, 모든 데이터셋을 한번 학습완료 할 때마다 모델을 저장하였습니다.

**모델 평가**실험 계획 시 평가 기준으로 삼은 목표 f1-score는 0.8이었고, 해당 모델의 결과를 도출하니, 오른쪽 표와 같은 평가 결과가 나왔습니다.   
무표정, 슬픈 표정에서는 미묘하게 평가 기준에 미치지 못했는데요, 아마 이런 결과를 보인 것은 사람마다 입꼬리, 눈 모양에 따라서 비슷한 표정 처럼 보이기 때문이라고 생각했습니다.

**모델 결과 코딩**플라스크를 이용하여 서버를 열었고,  
학습한 모델을 가져와서 기쁨, 분노, 슬픔에 대해서 예측확률이 0.4가 넘는다면 행복, 격노, 통곡 으로 바꾸어 전달했습니다. 이렇게 총 무표정과 놀람을 포함하여 8가지의 감정이 도출됩니다.

**POSTMAN 테스트**포스트맨으로 이미지에 대한 결과를 확인했는데요,  
사진을 넣어서 확인한 결과 무표정으로 성공적으로 분류되었습니다.

**남은 구현사항**다음은 저희 6조가 남은 캡스톤디자인 기간 내에 구현해야할 사항입니다.  
Frontend는 현재 로딩페이지, 캘린더페이지, AI일기요약페이지의 디자인을 우선 구현할 예정입니다. 남은 모든 페이지의 디자인을 마친 후엔, 캘린더의 감정이모지 및 일기 관리 기능을 연결하고, Backend에서 구현한 AI일기요약 및 한줄평 기능을 연결할 예정입니다. 그리고 모든 기능이 연결되었다면 마지막으로 '오늘의 감정'에 따른 반응형 서비스를 제공하여 저희 6조의 최종목표인 감정에 따른 서비스를 구현할 예정입니다.  
Backend는 현재 APP에 필요한 모든 기능들은 구현 완료된 상태입니다. 따라서 추가적인 문제가 발생하지 않는지 Frontend와의 Feedback을 통해 오류를 개선하고, 그리고 APP 배포를 위한 Spring 및 Flask 서버를 구축할 예정입니다.  
AI는 현재 감정 인식 모델을 구현하고, Backend에게 모델을 넘겨 서버를 구축한 상태입니다. 하지만 아직까지는 처음에 의도했던 방식처럼 기분 감정을 분류하여 해당 감정의 확률값이 0.4 이하면 화남, 기쁨, 슬픔으로 분류하고 해당 감정이 크게 적중하면 격노, 행복, 통곡이라는 감정으로 변환할 예정이었으나,  
초기 의도와 다르게 현재 파인 튜닝의 결과로 학습이 매우 잘 된 탓에, 확률값이 0.85에서 0.98 사이의 값으로 반환되어 화남, 기쁨, 슬픔에 대한 결과가 반환되지 않고 있습니다.   
따라서 이제는 남은 기간동안 모델이 화남, 기쁨, 슬픔과 같은 감정에 대한 결과를 얻게하기 위해 지속적으로 모델을 개선하여 최종적으로는 8가지의 감정 모두를 원활히 표현할 수 있도록 할 예정입니다.

**최종 목표**저희 앱은 최종적으로는 앞에서 Figma 툴을 사용해 디자인 한 기능들을 모두 구현하는 것을 목표로 하고 있습니다.  
또한 현재 AI 모델을 더욱 개선하여 보다 정확한 결과를 도출할 예정이고,  
서버 구축을 성공적으로 마친다면 '기분어때'앱을 배포까지 계획하고 있습니다.

**향후 계획**  
다음은 간트차트로 나타낸 주차별 계획입니다. 현재 5월 셋째주가 시작되었고,  
이번주를 포함하여 다음주까지 일차적인 개발을 모두 마칠 계획입니다. 이후 요구사항 및 기능점검을 모두 마친다면 공식 배포하기 위한 서버 구축을 할 예정이고, 계획이 순조롭게 풀린다면 최종발표 이전에 안드로이드 Google PlayStore에 앱을 배포할 수 있으리라 기대하고 있습니다.

감사합니다 이상으로 6조 중간발표를 마치겠습니다.