데이터 시각화를 활용한
사용자 감정 분석 플랫폼 구현

최종 보고서

팀명	소요
팀장	성도범
팀원	송시우 성민기 김세엽

목차

1. 배경

- 본 프로젝트는 사용자들이 자신의 감정을 자유롭게 표현하고, 이를 바탕으로 감정 분석 및 소통이 가능한 플랫폼을 개발하는 것을 목표로 하였습니다.
- 위 플랫폼은 게시글 작성, 실시간 채팅, 감정 분석, 사용자 인증 등 다양한 기능을 제공합니다.

1.1 팀 소개 및 역할 상세화

- 성도범(조장):
  - 총괄
  - 프론트엔드 채팅 페이지 개발
  - 프론트엔드 채팅 로직 구현
- 성민기(조원):
  - 프론트엔드 게시글 및 세부 페이지 개발
  - 감정 관리 시스템 페이지 개발
- 송시우(조원):
  - 인공지능 모델 개발 및 학습
  - 인공지능 <-> 백엔드 연동 구현
- 김세엽(조원):
  - 전체 백엔드 구현
  - 프론트엔드 <-> 백엔드 연동 구현

2. 플랫폼 개요 및 구현 내용 소개

2.1 플랫폼 개요

JustFeeling 플랫폼은 모던 웹 기술을 활용하여, SNS 의 기본적인 기능(회원, 피드, 채팅, 프로필 등)을 풀스택으로 구현한 예제 플랫폼입니다.

JustFeeling 은 사용자가 자신의 감정이나 생각을 자유롭게 공유하고, 다른 사용자와

소통할 수 있는 소셜 네트워크 SNS 플랫폼입니다. 주요 기능으로는 회원가입/로그인, 피드(게시글) 작성 및 조회, 실시간 DM(Direct Message)채팅, 프로필 관리 등이 있습니다.

프론트엔드는 React(타입스크립트) 기반, 백엔드는 Spring Boot(Java)기반으로 구성되어 있으며, RESTful API 를 통해 양쪽 통신하는 구조로 구현했습니다.

2.2 폴더 구조 기반 구현 내용(백엔드)

2.2.1 SpringBoot 기반의 Rest API 서버로, 사용자 인증, 게시글 관리, 채팅 등 핵심 비즈니스 로직을 담당합니다.

2.2.2 주요 패키지 및 역할

- Controller: 클라이언트 요청을 처리하는 REST API 엔드포인트 제공(회원가입, 게시글, 채팅, 인증)
- Service: 실제 비즈니스 로직 구현(JWT 인증, 게시글/사용자/채팅 처리)
- Repository: JPA 기반 데이터베이스 접근 계층(사용자, 게시글, 채팅방, 메시지)
- Entity: 데이터베이스 테이블과 매핑되는 도메인 모델
- Dto: 데이터 전송 객체(요청/응답용)
- Config: 보안(JWT), 예외처리, OpenAPI(Swagger)등 설정
- 데이터베이스: H2 in-memory DB 사용(개발/테스트용)
- API 문서: Swagger UI 제공
- 실행: Maven 기반 빌드 및 실행

2.3 프론트엔드

2.3.1 React + TypeScript 기반의 SPA 로 사용자 인터페이스와 클라이언트 로직을 담당합니다.

2.3.2 주요 컴포넌트 및 역할

- Pages: 각 주요 화면(홈, DM, 프로필, 로그인, 게시글 작성 등)구현
- Components: 공통 UI, 홈화면, UI 요소 등 재사용 컴포넌트 분리
  - Common: 로그인 화면, 네비게이션 바
  - Home: 피드, 배너 등 홈 관련 컴포넌트

    - Ui: 카드 등 UI 요소
- Services: API 통신 로직(axios 관련 함수 처리)
- Stores: 전역 변수 상태 관리
- Hooks: 커스텀 훅
- Constants: 테마, 상수 등
- Styles: 전역/공통 스타일
- Assets: 이미지 아이콘 등 정적 리소스

2.3.3 주요 구현 기능

- 회원가입/로그인(유효성 검사, JWT 기반 인증)
- 피드(게시글) 목록 및 작성
- 실시간 DM 채팅방
- 프로필 조회 및 수정
- 반응형 UI 및 UX 개선

3. 기술 스택 및 특징

- 프론트엔드: React, TypeScript, Vite, Zustand, Emotion, Axios API
- 백엔드: Spring Boot, Spring Security(JWT), JPA, H2 Database, Swagger API
- API 통신 : RESTful 방식, JWT 토큰 기반 인증/인가

계층	기술	버전	용도
언어	Java	17	메인 프로그래밍 언어
프레임워크	Spring Boot	3.2.0	애플리케이션 프레임워크
보안	Spring Security	3.2.0	인증 및 권한 관리
데이터	Spring Data JPA	3.2.0	ORM 및 데이터 접근
DB	H2	2.2.224	인메모리 DB (개발용)
통신	WebSocket	3.2.0	실시간 메시징
인증	JWT	0.11.5	토큰 기반 인증
문서화	OpenAPI/Swagger	2.2.0	API 문서화

계층	기술	버전	용도
빌드 도구	Maven	3.8+	프로젝트 관리 및 빌드

```
just_feeling/
├── backend/                # Spring Boot 기반 백엔드 서버
│   ├── src/
│   │   └── main/
│   │       ├── java/
│   │       │   └── com/justfeeling/
│   │       │       ├── controller/   # REST API 컨트롤러
│   │       │       ├── service/      # 비즈니스 로직 서비스
│   │       │       ├── repository/   # JPA 리포지토리
│   │       │       ├── entity/       # DB 엔티티 클래스
│   │       │       ├── dto/          # 데이터 전송 객체
│   │       │       ├── config/       # 설정 및 예외 처리
│   │       │       └── JustFeelingApplication.java # 메인 클래스
│   │       └── resources/            # 리소스(설정, 정적파일 등)
│   ├── pom.xml                       # Maven 빌드 설정
│   └── ...                           # 기타 빌드/설정 파일
├── src/                    # React 기반 프론트엔드 소스
│   ├── pages/              # 주요 페이지 컴포넌트
│   ├── components/         # UI/공통/홈 컴포넌트
│   ├── services/           # API 통신 등 서비스
│   ├── stores/             # 상태관리(store)
│   ├── hooks/              # 커스텀 훅
│   ├── constants/          # 상수/테마
│   ├── styles/             # 스타일 파일
│   ├── assets/             # 이미지/아이콘 등 에셋
│   ├── App.tsx, main.tsx   # 엔트리포인트
│   └── ...
├── public/                 # 정적 파일
├── package.json            # 프론트엔드 의존성/스크립트
├── README.md               # 프론트엔드/전체 설명
├── README_Backend.md       # 백엔드 설명
└── ...
```

4. 디렉토리 구조

- **회원가입 / 로그인**
  - AuthController, AuthService, JWTService 를 통해 JWT 기반 인증 구현
- **프로필 관리**
  - UserController, UserRepository 등
- 게시글 작성/조회/삭제
  - PostController, PostService, PostRepository
- **감정 태그/분류**
  - 게시글에 감정 카테고리 지정 가능
- **DM(Direct Message)**
  - ChatController, ChatRoom, ChatMessage 등 엔티티 및 컨트롤러

- **채팅방 관리**
    - 채팅방 생성, 메시지 저장 등
- **감정 분석**

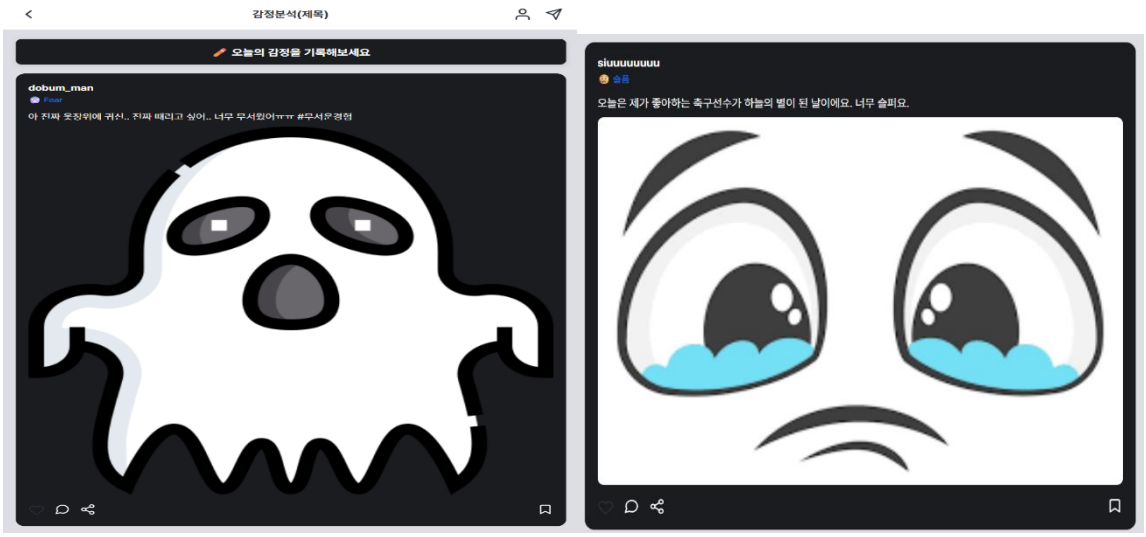
```
sent-api/
├── README_API.md # FastAPI 앱의 설명 문서
├── requriements.txt # 필요한 패키지 목록
├── serve.py # FastAPI 앱 진입점 (메인 서버 코드)
├── structure.txt # 디렉토리 구조 설명 텍스트

├── emotion_model/ # 학습된 감정 분류 모델 관련 파일
│   ├── config.json # 모델 설정 정보 (hidden size 등)
│   ├── label2id.json # 감정 라벨 → 정수 ID 매핑
│   ├── model.safetensors # 학습된 모델 파라미터 (PyTorch)
│   ├── special_tokens_map.json # 특수 토큰 매핑 정보
│   ├── tokenization_kobert.py # KoBERT용 토크나이저 구현
│   ├── tokenizer_78b3253a26.model# SentencePiece 토크나이저 모델
│   ├── tokenizer_config.json # 토크나이저 설정 파일
│   └── vocab.txt # 토크나이저의 단어 사전

└── pycache/ # Python 캐시 디렉토리
```

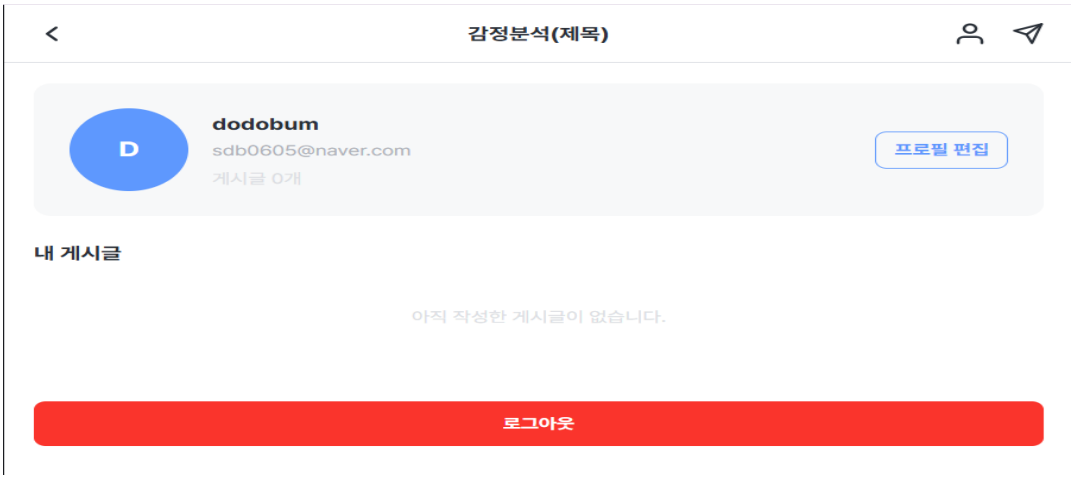
    - Client 요청 시 모델 추론 결과를 텍스트(감정)으로 반환하는 FastAPI 기반 구현 디렉토리.

5. 프론트엔드 (UI 설명)

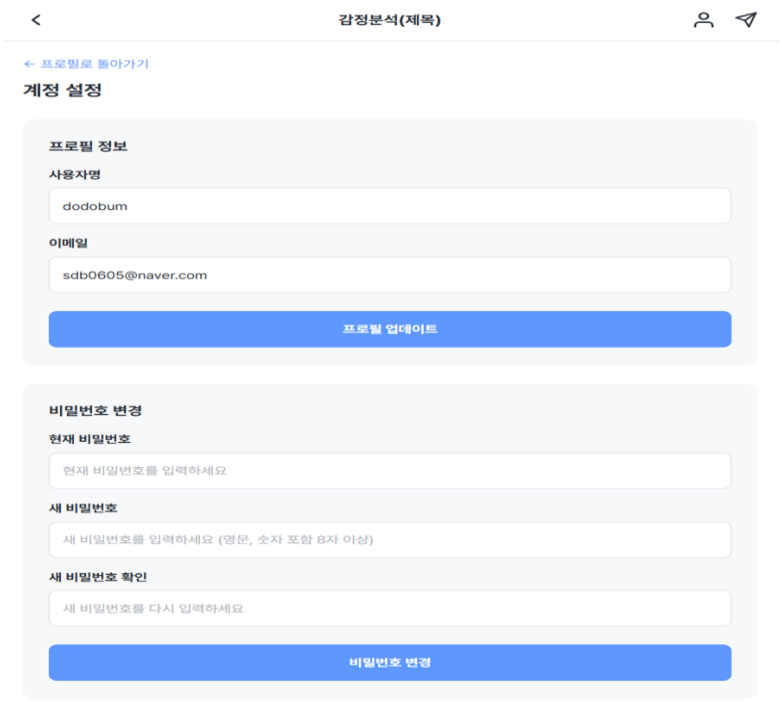


- 메인 게시글을 출력하는 화면

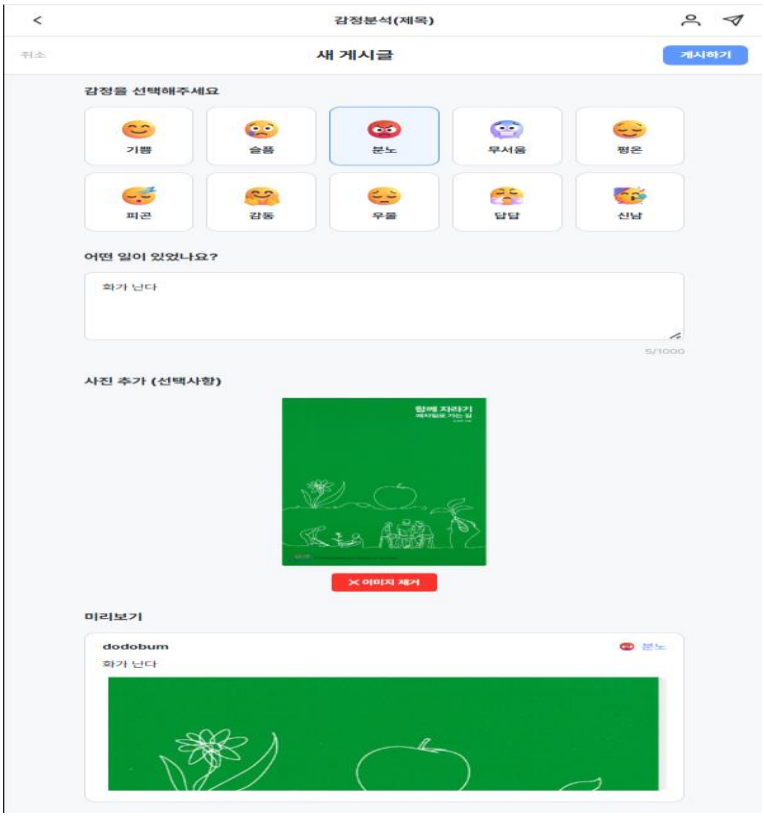
- Dobum_man 님이 작성한 게시글의 제목을 AI 가 분석해서 #Fear 라는 감정을 분석함
- 해당 게시글의 "공감"버튼을 클릭하거나 "댓글"을 작성하면, 작성한 내용을 AI 가 분석해서 감정을 추론한 다음, 해당 감정 점수를 증가시켜 사용자의 감정 점수를 계산을 수행함



- 로그인 페이지
- 본인이 로그인한 정보를 출력해줌



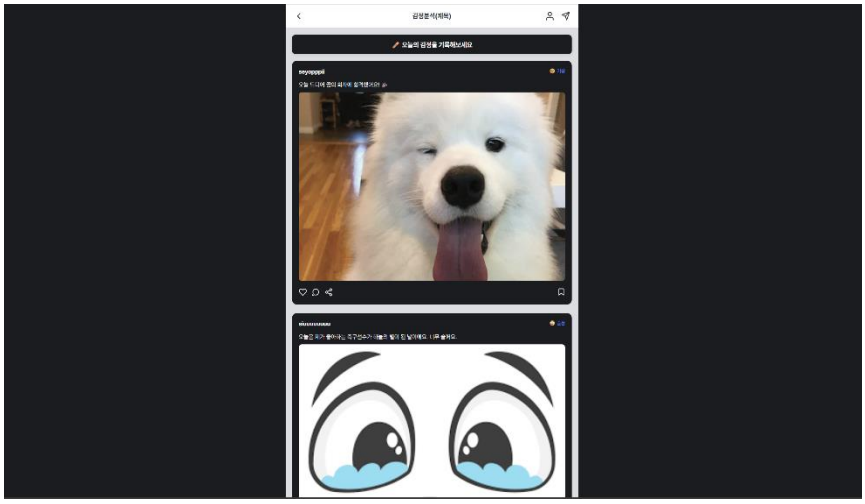
- 프로필 편집 페이지



- 게시글 작성 페이지

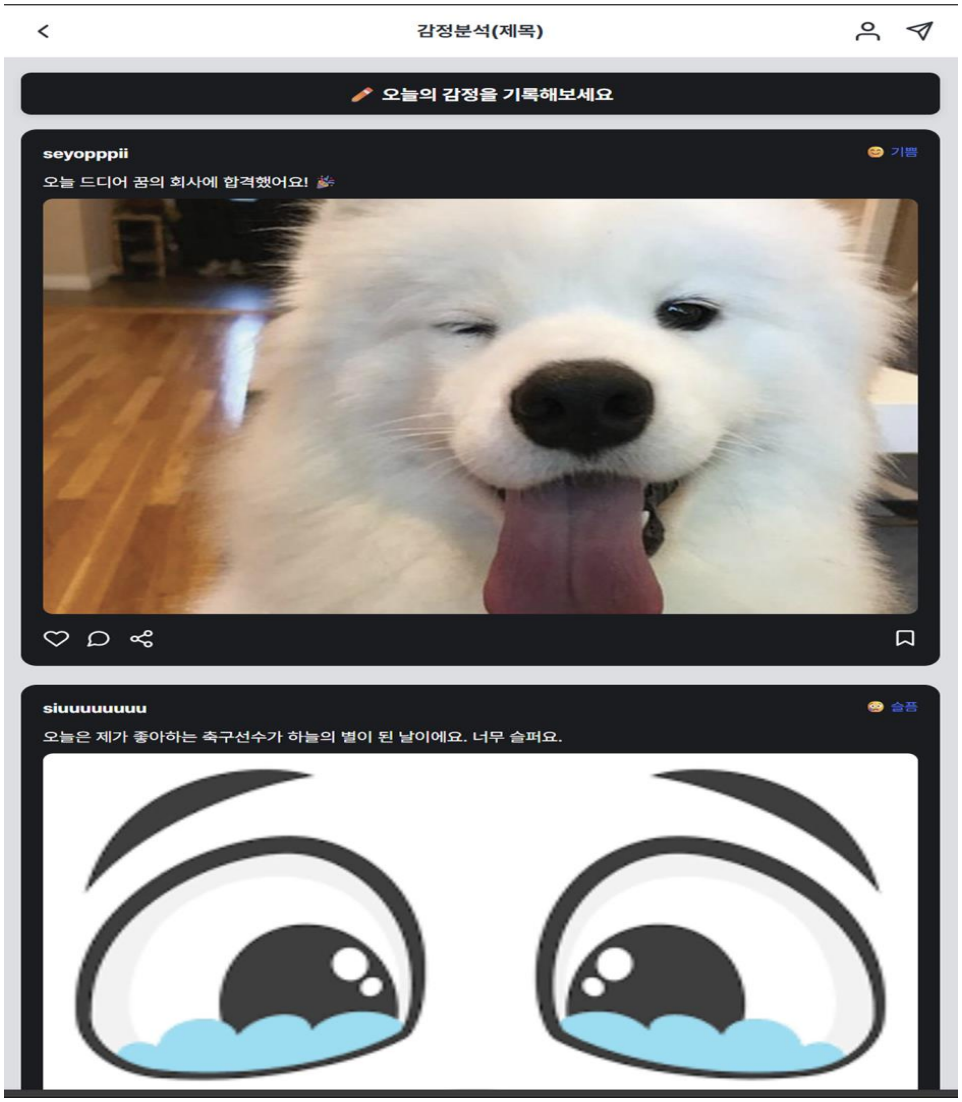
- 본인의 감정과 텍스트를 기반으로 AI 가 감정을 분석해서 사용자의 감정 점수를 반영함

5.1 공통사항



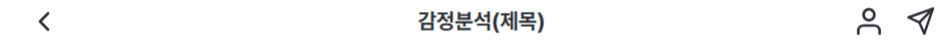
- 720PX 규격으로 설계하여 웹/앱 호환 가능한 서비스를 제공함
- 심플하면서도 세련된 UI 도입

5.2 **홈 화면 UI – 게시글 열람**



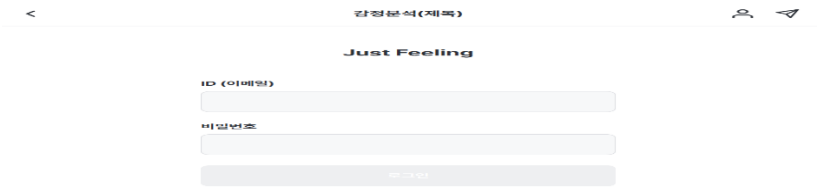
- 게시글 작성을 쉽게 할 수 있도록 상단에 "오늘의 감정을 기록해보세요" 버튼 배치
- 작성된 피드들이 작성자, 감정, 본문, 사진 등의 정보를 담아서 보여지도록 배치
- 감정분석 결과에 기반하여 사용자에게 맞춤형 피드를 제공함
- 게시글은 공감 버튼, 댓글 달기, 공유하기, 저장하기 등의 기능을 내포
- 공감 버튼을 클릭, 댓글 작성, 공유, 저장 등의 기능을 사용자가 사용할 경우, 피드를 AI 가 분석하여 감정을 추론하고 해당 감정 점수를 업데이트하여 계산을 진행하는 구조

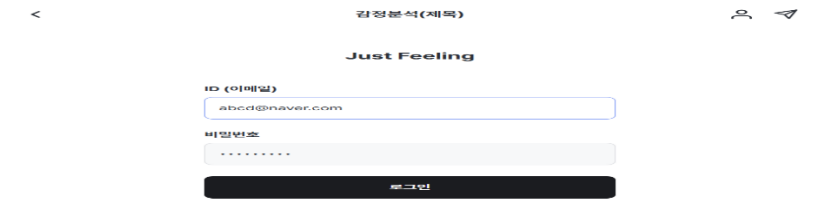
5.3 네비게이션 바 UI



- 간단하면서도 필수적인 요소를 담음
- 뒤로가기, 홈 접근 (제목), 마이페이지, DM 기능
- 이를 통해, 사용자가 앱 내에서 효율적인 이동이 가능하도록 설계 및 구현

5.4 로그인 화면 UI





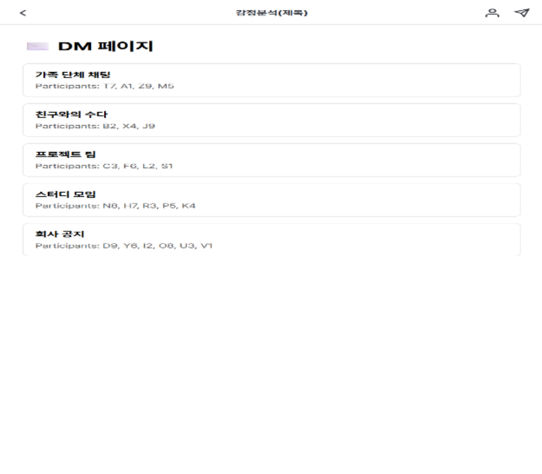
[로그인 화면 UI - 로그인 버튼 활성화]

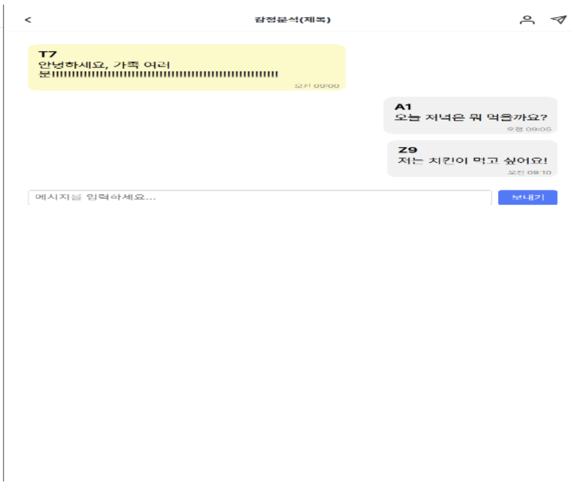
[로그인 입력 유효성 검사 UI]

- 마이페이지 접근, DM 기능 접근, 초기 화면 접근 등 앱 내에서 계정 정보를 불러와야 하는 상황에 자동으로 로그인 유도

- 이메일, 비밀번호 유효성 검사를 도입하여 올바른 입력을 필터링하여 받도록 제작 (유효한 입력이 있을 경우, 로그인 버튼 활성화)

5.5 DM 화면 UI





[DM 홈 화면 UI] **[DM 채팅 화면 UI]**

- 네비게이션 바 기준 우측 상단 아이콘을 통해 접근 가능

- DM 기능 홈에서는 채팅방 목록을 열람 가능

- 홈에서는 채팅방 제목, 참가자 등 확인 가능

- 채팅 화면에서는 채팅 작성자 이름과 내용 등을 확인 가능

6. 백 엔드

6.1 JWT 토큰 형식

```
{
  "header": {
    "alg": "HS256",
    "typ": "JWT"
  },
  "payload": {
    "sub": "user@email.com",
    "userId": 123,
    "userName": "사용자명",
    "iat": 1640995200,
    "exp": 1641081600
  }
}
```

6.2 보안 설정

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .cors(cors -> cors.configurationSource(corsConfigurationSource()))
            .csrf(csrf -> csrf.disable())
            .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .authorizeHttpRequests(authz -> authz
                .requestMatchers("/emoai/auth/**").permitAll()
                .requestMatchers("/emoai/posts/**").permitAll()
                .requestMatchers("/ws/**").permitAll()
                .requestMatchers("/v3/api-docs/**", "/swagger-ui/**").permitAll()
                .requestMatchers("/h2-console/**").permitAll()
                .anyRequest().authenticated()
            );
    }
}
```

6.3 실시간 통신

6.3.1 웹 소켓 통신

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.setApplicationDestinationPrefixes("/app");
        config.enableSimpleBroker("/topic", "/queue");
        config.setUserDestinationPrefix("/user");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/ws")
                .setAllowedOrigins("http://localhost:3000")
                .withSockJS();
    }
}
```

6.4 설정

6.4.1 CORS 설정

```java
@Bean
public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.setAllowedOriginPatterns(Arrays.asList("*"));
    configuration.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"));
    configuration.setAllowedHeaders(Arrays.asList("*"));
    configuration.setAllowCredentials(true);

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", configuration);
    return source;
}
```

6.4.2 개발환경

```yaml
server:
  port: 8080

spring:
  datasource:
    url: jdbc:h2:mem:testdb
    driver-class-name: org.h2.Driver
    username: sa
    password:

  h2:
    console:
      enabled: true
      path: /h2-console

  jpa:
    hibernate:
      ddl-auto: create-drop
    show-sql: true
    properties:
      hibernate:
        format_sql: true

logging:
  level:
    com.justfeeling: DEBUG

jwt:
  secret: justFeelingSecretKeyForJwtTokenGenerationThatMustBeLongEnoughForHS256Algorithm
  expiration: 86400000 # 24시간
```

6.5 성능

6.5.1 응답 시간

- API 응답: 95% 요청이 500ms 이내
- 데이터베이스 쿼리: 95% 쿼리가 100ms 이내

- WebSocket 메시지: 95% 메시지가 100ms 이내

6.5.2 **처리량**

- 동시 사용자: 1,000명
- 초당 요청: 1,000 RPS
- 데이터베이스 연결: 최대 50개

6.5.3 **확장성**

- 수평 확장: 무상태 설계로 인스턴스 추가 기능
- 데이터베이스: 읽기/쓰기 분리 가능
- 캐싱: Redis 데이터베이스를 도입해서 성능 향상

7. 인공지능 모델 학습 및 연동

7.1 모델 선정 및 데이터셋 선정

하드웨어상의 제약사항을 고려하여 비교적 적은 파라미터로 구동 가능한 모델 선정. 감정이 태깅된 한글 데이터셋중 감정 태깅이 정확하게 된 데이터셋이 충분하지 않아 사전이 구성되어있고 한글 기반으로 학습된 SKTBrain/KoBERT 를 사용. 데이터셋은 AI Hub 의 감정 분류를 위한 대화 음성 데이터셋의 텍스트 데이터를 사용.

wav_id	발화문	상황	1번 감정	1번 감정세기	2번 감정	2번 감정세기	3번 감정	3번 감정세기	4번 감정	4번감정세기	5번 감정	5번 감정세기
5ee1e7939aa8ea0eec53fac7	나 어제 헤어졌어	sad	Sadness	1	Sadness	1	Sadness	1	Sadness	1	Sadness	1
5ee1e7a379bf120ed2b81ba2	어쩌다 보니까 그렇게 됐네	sad	Sadness	1	Sadness	1	Neutral	0	Sadness	1	Sadness	1
5ed9793b9aa8ea0eec53f7f1	유기견 다큐멘터리를 봤는데 무책임한 사람들 때문에 너무 화가 나	disgust	Sadness	2	Sadness	2	Angry	1	Angry	1	Angry	1
5ed979582880d70f286128c3	버려지는 유기견들의 생활을 다른 다큐멘터리 없어	disgust	Sadness	1	Sadness	2	Sadness	1	Angry	1	Sadness	2
5ed9797b1dcf350eeded50b8	작년보다 5배는 더 늘어나는 것 같대 최근 들어 더 늘어나고 있고	disgust	Fear	1	Sadness	1	Sadness	1	Angry	1	Sadness	2
5ed9799ac90a530ee56b5f6b	정부보조금이랑 사람들의 기부금으로 운영되고 있어	disgust	Neutral	0	Neutral	0	Sadness	1	Sadness	1	Sadness	2
5ed979dc7e21a10eee253e90	맞아 벌어지는 대부분의 아이들이 다 큰 강아지 들였어 아직도 상처	disgust	Sadness	2	Sadness	2	Sadness	2	Sadness	2	Sadness	2
5ed97a22c90a530ee56b5f6c	어제저녁에 진짜 무서웠어	fear	Fear	2	Fear	2	Fear	1	Fear	1	Fear	2
5ed97a381dcf350eeded50bc	친구랑 약속 있어 나갔는데 밥 먹고 이따가 지진이 발생하는 거야 얼	fear	Fear	2	Fear	2	Fear	2	Fear	1	Fear	1
5ed97a5cc90a530ee56b5f6d	큰 지진은 지나갔는데 여진이 조금씩 있는 거 같기도 해	fear	Fear	2	Fear	2	Fear	1	Fear	1	Fear	1
5ed97a779aa8ea0eec53f7f5	다른 테이블에서 밥 먹던 사람들이 도망치다가 넘어지고 그 와중에	fear	Sadness	1	Fear	2	Sadness	1	Fear	1	Fear	1
5ed97a977e21a10eee253e92	그렇지 않아도 오늘 친구 데리고 병원 가기로 했어	fear	Neutral	0	Sadness	1	Sadness	1	Sadness	1	Fear	2
5ed97ac29aa8ea0eec53f7f8	룸메이트와 너무 자주 싸우게 돼	anger	Disgust	2	Angry	2	Angry	1	Angry	1	Fear	1

5명의 사람이 발화문에 대해 라벨링을 하여 구축된 데이터이며, 감정의 종류는 Neutral, Sadness, Angry, Fear, Disgust, Surprise, Happiness 총 7가지 종류로 구성되어있으며 감정의 강도는 0, 1, 2로 태깅되어있음. 데이터셋은 발화문 43991문장으로 구성되어있음. 해당 데이터셋을 기반으로 Fine-tuning 을 진행하여 감정 분석에 적합한 모델 생성

7.2 데이터 추가 라벨링 및 학습

```
개를 예쁘다고 사놓고 끝까지 키우지도 않고 버리는 사람들이 엄청 많아졌대. | Angry
지금도 그대로 있어. 치우는 사람이 없어. | Disgust
맞아. 무기력증인 것 같아. 한동안 정말 바빴었거든. | Sadness
오늘이 발표날인데 연락이 없더라고. 그래서 알아봤더니 명단에 내 이름이 없대. | Sadness
그치. 개 키우는 사람이 늘어나니까 그만큼 버리는 사람도 늘어나는 거야! | Angry
공채로 볼 수 있는 마지막 회사였어. 그래서 정말 많이 노력했거든. | Sadness
버려진 개가 워낙 많으니까 시설이 부족할 수 밖에 없고 당연히 상황이 너무 열악하지. | Angry
나 면접 또 떨어졌어. | Sadness
부모님께서 기대가 너무 크셔서 실망도 크실까봐 아직 말씀 못 드렸어. | Sadness
그렇지! 아 나 진짜 너무 싫어 그런 사람들. 대체 생명이 있는 개를 뭐라고 생각하는 거야? | Angry
지난 번에 토 해놓은 것도 관리하시는 분이 치우셨거든. 또 어떻게 말씀 드리지? | Disgust
해피가 어제 세상을 떠났어. | Sadness
오늘은 아침부터 입맛이 없어서 아무것도 못 먹고 있어. 이따가 저녁에나 먹으려고. | Sadness
```

- 재라벨링된 데이터

```
Epoch 1/20 Training: 100%|                                        | 550/550 [02:01<00:00,  4.54it/s]
Epoch 1/20 - Train loss: 1.4701
Epoch 1/20 Validation: 100%|                                      | 138/138 [00:11<00:00, 12.27it/s]
Epoch 1/20 - Validation accuracy: 0.7104
Epoch 2/20 Training: 100%|                                        | 550/550 [02:00<00:00,  4.56it/s]
Epoch 2/20 - Train loss: 0.7750
Epoch 2/20 Validation: 100%|                                      | 138/138 [00:11<00:00, 12.48it/s]
Epoch 2/20 - Validation accuracy: 0.7534
Epoch 3/20 Training: 100%|                                        | 550/550 [02:00<00:00,  4.57it/s]
Epoch 3/20 - Train loss: 0.6331
Epoch 3/20 Validation: 100%|                                      | 138/138 [00:10<00:00, 12.60it/s]
Epoch 3/20 - Validation accuracy: 0.7617
```

감정 데이터를 다시 **[발화문 | 감정]** 형식의 csv 파일로 라벨링을 진행. 5명의 감정 라벨중 가장 많이 나온 라벨이 있다면 해당 감정으로 라벨링. 동일한 개수의 감정이 둘 이상 존재한다면 감정세기의 합이 가장 큰 감정으로 라벨링함. 이마저도 동일하다면 더 비중이 많은 Angry>Disgust,Surprise , Sadness>Fear 순의 우선순위를 두었고, Neutral 이 가장 우선순위가 낮게 라벨링하여 학습 데이터셋을 구성.

적은 데이터셋을 감안하여 2e-5의 learningrate 로 학습을 진행하였고, Trainset 에 대해 과적합이 빠르게 발생하여 20epoch 학습을 진행. 그 외 parameter 은 KoBert 오픈소스 코드와 동일하게 적용함. model.save_pretrained(output_dir)메서드를 사용하여 저장.


```
{
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4",
    "5": "LABEL_5",
    "6": "LABEL_6"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4,
    "LABEL_5": 5,
    "LABEL_6": 6
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.53.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 8002
}
```


-Model parameter

7.3 모델 연동 및 감정점수 기능 추가

학습이 완료되어 safetensor 형식으로 저장된 모델을 구동할 서버를 구현. FastAPI 를 이용하여 구현하였고, 학습된 모델을 불러와 클라이언트가 보낸 텍스트에 대해 감정분석을 진행하고, 모델의 추론 index 를 감정으로 변환해 반환하게 구현되어있음.

-감정분석 테스트 결과. 서버에 "너무 기뻐 지금"이라는 텍스트와 함께 요청을 보내고 "기쁨"이라는 감정이 확률 74%로 가장 높게 나왔으며 return 됨

Api 추론 결과를 받아오는 Service 를 구현하고 사용자가 게시글 생성 시 감정분석을 진행하여 EmotionRecordRepository 에 사용자의 감정 상태를 반영하게 구현. 사용자의 게시글 감정 선택에 따라 해당 감정점수를 5점 늘리고, 모델의 분석 결과에 따라 3점을 늘려 하루의 감정 상태를 확인할 수 있게 구현함.

8. 참여후기

조장: 성도범

먼저, 한창 바쁠 시기인 학부 3학년 과정에 조원들과 현장 실습을 참여하고 있는 조원과 함께 팀 프로젝트를 참여하게 되서 일정을 소화화기 힘들 수 있다는 걱정이 들었지만, 조장의 요구사항을 기간내에 수행해준 조원들에게 너무나도 감사하다는 말을 전하고 싶습니다. 저는 여러 프론트엔드 프로젝트를 수행해봐서 경험이 있지만, 몇몇 조원들은 처음 접해보는 웹 프로젝트이기에 최대한 다양한 경험을 시켜주려고 노력했습니다. 저는 이번 프로젝트에서 경험해보지 못한 채팅 로직 구현을 담당했습니다. 백엔드가 구현한 웹 소켓 통신을 받아와서 채팅방 -> 채팅 리스트를 출력하도록 구현하는 경험을 쌓을 수 있었습니다. 추가로, 조장이라는 역할이 백엔드와 프론트엔드를 모두 잘 숙달해야만 할 수 있다고 느꼈습니다. 저희 프로젝트는 백엔드와 프론트엔드 외에 AI 모듈을 사용하고 있기 때문에, AI 에 대해서 지식이 없어서 AI 담당 조원과 소통이 힘들었습니다. 백엔드 서버와 AI 모듈 통신은 아직 경험이 없기 때문에, 다음 프로젝트를 참여할 기회가 생긴다면, AI 모듈과 백엔드 연결을 공부해서 역량을 쌓아갈 계획입니다.

조원: 송시우

웹 프로젝트가 처음이라 많이 걱정되었으나 훌륭한 조원들 덕분에 무사히 프로젝트를 끝마칠 수 있어 좋았습니다. 모델 학습부분에서는 한글 데이터셋이 제한적이기에 AI Hub 의 데이터셋을 사용할 수밖에 없었는데, 다운받아본 데이터셋들의 감정 태깅이 제대로 되지 않아 충분한 데이터셋을 확보하지 못하고 학습을 진행할 수밖에 없었고, 이 부분이 가장 아쉬웠습니다.

이번 프로젝트를 통해 웹 프로젝트의 전체적인 흐름을 이해할 수 있었고, 감정 지수 관리 작업을 하면서 Spring boot 의 책임 부담과 의존성 주입 관련 내용을 이해하는데 많은 도움이 되었습니다. 다음 프로젝트를 참여할 기회가 생긴다면 데이터셋 수집 및 환경 구축에 더 많은 신경을 쓸 계획입니다.

조원: 성민기

이번 프로젝트는 웹 개발에 대한 실무 감각을 키울 수 있는 매우 유익한 경험이었습니다.

특히 팀 프로젝트를 통해 기획부터 구현까지의 전반적인 흐름을 함께 경험하면서 협업의

중요성과 실전 감각을 체득할 수 있었습니다.

React 와 TypeScript 같은 프론트엔드 핵심 기술을 처음 접해보았는데, 컴포넌트 기반 설계나 타입 안정성 확보와 같은 개념을 직접 코드로 구현해보며 실력을 키울 수 있었습니다. 또한 emotion 을 활용한 CSS-in-JS 방식의 스타일링, Zod 와 React Hook Form 을 통한 폼 유효성 검사, axios 를 이용한 API 통신 처리 등 현업에서 자주 쓰이는 기술 스택들을 직접 적용해보며 많은 것을 배울 수 있었습니다.

머릿속으로만 상상하던 서비스가 점점 눈에 보이는 형태로 완성되어 가는 과정은 정말 뿌듯했고, 개발자로서의 동기부여가 더욱 강해졌습니다. 이번 경험을 통해 프론트엔드 개발에 대한 흥미와 자신감을 크게 얻을 수 있었습니다.

다음 프로젝트에 참가한다면 조금 더 고도화 된 프론트엔드 기술을 활용하여 더 퀄리티 좋은 코드를 작성하고, 백엔드와의 조율을 더 원만하게 진행하고자 노력하고 싶습니다.

조원: 김세엽

이 프로젝트를 통해 Spring Boot 생태계의 강력함을 체감할 수 있었습니다. 특히 Spring Data JPA 의 편리함과 Spring Security 의 보안 기능이 인상적이었습니다.

실시간 채팅 기능 구현을 통해 WebSocket 의 동작 원리를 이해하고, JWT 인증 시스템을 통해 현대적인 웹 애플리케이션의 보안 패턴을 학습할 수 있었습니다.

프론트엔드 팀과의 협업을 통해 API 설계의 중요성과 문서화의 필요성을 깨달았으며, CORS 설정이나 데이터 형식 등에서 발생할 수 있는 문제들을 해결하는 경험을 쌓을 수 있었습니다.

이 프로젝트는 단순한 CRUD 애플리케이션을 넘어서 실시간 통신, 사용자 인증, 파일 업로드 등 실제 서비스에서 필요한 다양한 기능들을 구현해볼 수 있는 좋은 기회였습니다. 특히 감정 공유라는 독특한 주제로 인해 사용자 경험을 고려한 API 설계의 중요성을 배울 수 있었습니다.

앞으로도 이러한 경험을 바탕으로 더욱 복잡하고 실용적인 백엔드 시스템을 구축해나가고 싶습니다.

프론트엔드 결과물
https://github.com/wannagola/just_feeling/tree/main/src

백엔드 결과물
https://github.com/wannagola/just_feeling/tree/main/backend

감정 분석 인공지능 모델
https://github.com/wannagola/just_feeling/tree/main/sent-api

전체 프로젝트 결과물
https://github.com/wannagola/just_feeling