Android ChatBot On Device AI

202010142 최민성
202010094 고권표
202010116 신재석

목차

1. 팀원 간 역할 분담

1. 팀원 간 역할 분담

성명	역할
최민성	안드로이드OS UI/UX 제작, Tflite 모델 & 서버 연동
신재석	모델 제작,Tflite Convert, MetaData 제작, Tokenizer
고권표	모델 제작, 디버깅, MetaData 제작, Tokenizer

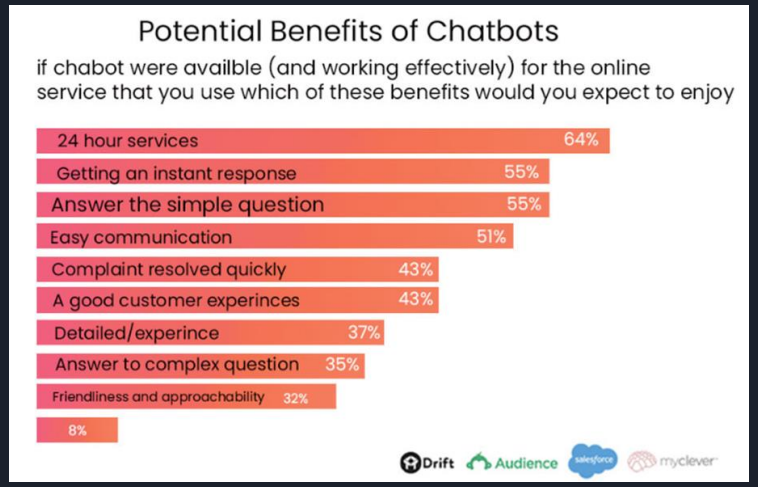
1. 팀원 간 역할 분담

S/W Android OS : 최민성
AI 모델 : 신재석 , 고권표
공동 작업

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
모델 및 S/W 설계															
S/W UI/UX															
S/W 서버 및 모델 연동															
모델 서칭 및 빌드															
모델 디버깅															
문서 작업															

2. Project의 필요성과 목적

2 . Project의 필요성과 목적

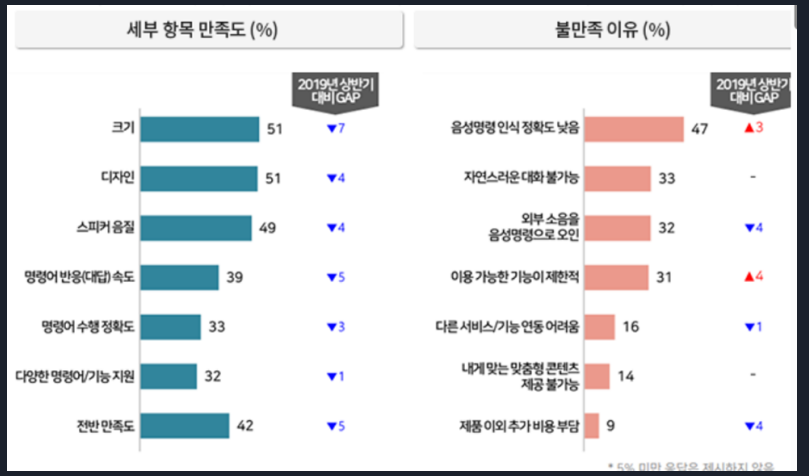


챗봇의 잠재적 수익성

Potential Benefits of Chatbots

자유 주제 대화 챗봇
이루다 개인정보 침해 및 법적 논란

Free-Talk chatbot LUDA Violation of personal information and legal issue



챗봇에 대한 만족도 조사 결과

Satisfaction survey result about chatbot

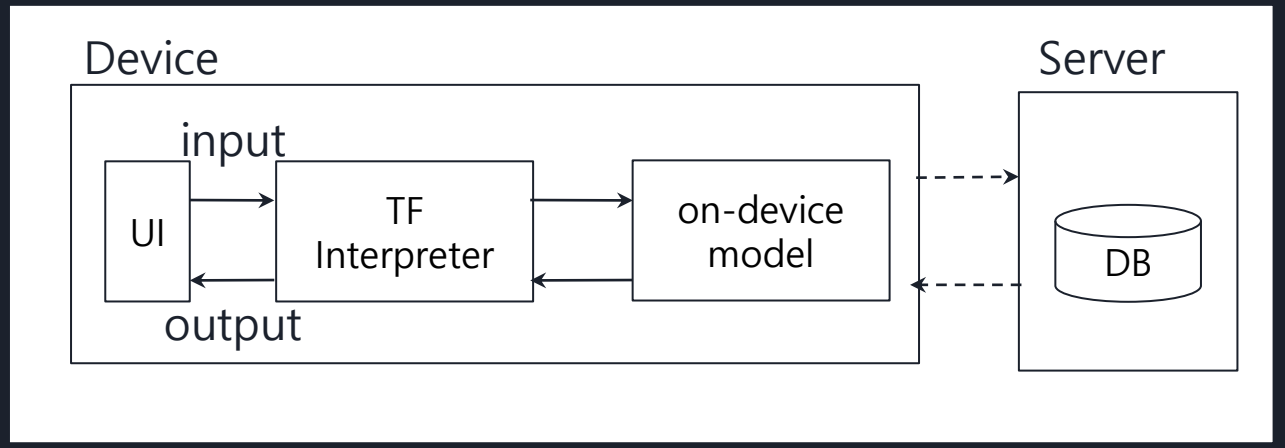
2 . Project의 필요성과 목적 - On-Device AI Model

- 파이어베이스 저장 가능 모델 용량의 한계(40MB)
- 지나치게 무거운 모델 (12 layer)

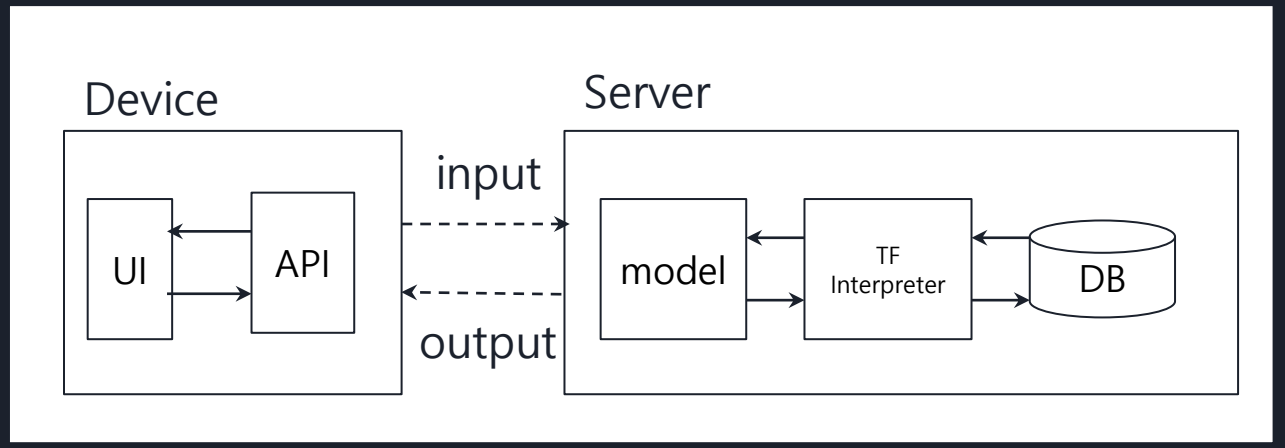
→ 모바일 장치에 직접 모델을 넣는 On-device 방식 채택

온디바이스의 장점 Advantage
- 오프라인 환경에서도 실행이 가능
- 서버에서 대화 내용에 대한 사적인 내용을 저장하지 않아 개인 정보 유출의 가능성 낮춤
- 개인별 맞춤화 모델이 형성

온디바이스의 단점 Disadvantage
- 사용자의 장치가 모델의 구동을 전담
- 일정 수준 이상의 성능과 용량이 요구됨



On-Device Architecture



Server-Based Architecture

3. Project의 내용

3. Project의 내용

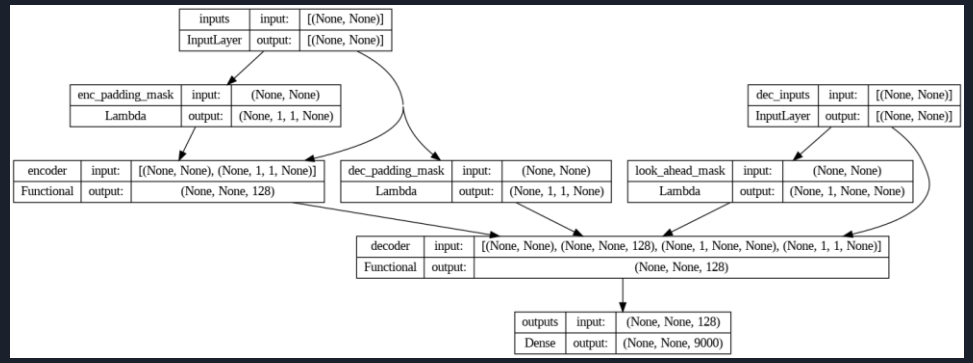
Project Content

AI Model

Android OS

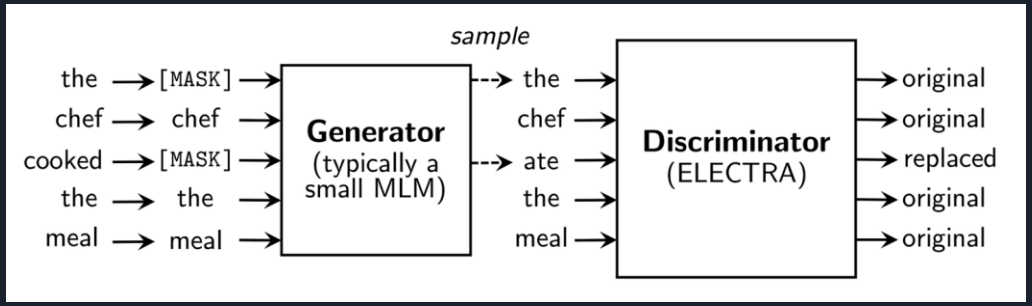
3. Project의 내용 (AI Model)

TRANSFORMERS (Week 2 - 4)



선정 근거 **Rationale**	자연어 처리의 기본적이고 단순한 모델 The basic and simple model of the NLP
특징 **Features**	단순한 아키텍처 Simple architecture
결점 **Defects**	고정된 답변 Answers are fixed 낮은 문맥 파악과 주제 관련 답변 Low content and topic comprehension

KoELECTRA + KoGPT2 (Week 5 - 6)



선정 근거 **Rationale**	KoELECTRA : 입력 문장의 토큰화 및 라벨링 Tokenizing & labeling of input sentence KoGPT2 : 라벨링 문장의 분류와 답변 생성 Classify labeled sentence and create answer
특징 **Features**	두 자연어 처리 모델의 결합 시도 Trial of two NLP method combination
결점 **Defects**	두 모델 간 라벨 전달의 어려움 Difficulty to apply label to other model 안드로이드 환경과의 연동 문제 Interwork problem with Android OS

3. Project의 내용 (AI Model) - KoGPT2

Torch GPT2

GPT2의 원형 모델

.tflite으로 바꾸는게 불가능해서 기각

TensorFlow (with 🤗 hugging face Tokenizer)

기존 Torch 모델을 텐서플로우에 맞게 적용하기 위해 변환

3. Project의 내용 (AI Model) - Tokenizer

- **토크나이저 Tokenizer**

입력 텍스트를 단어 단위로 쪼개어 학습가능한 형태로 변환시키는 모듈
SKT에서 학습한 KoGPT2 토크나이저

- **언어 모델 Language Model**

TFGPT2LNHeadModel
SKT에서 만든 TensorFlow GPT2 모델

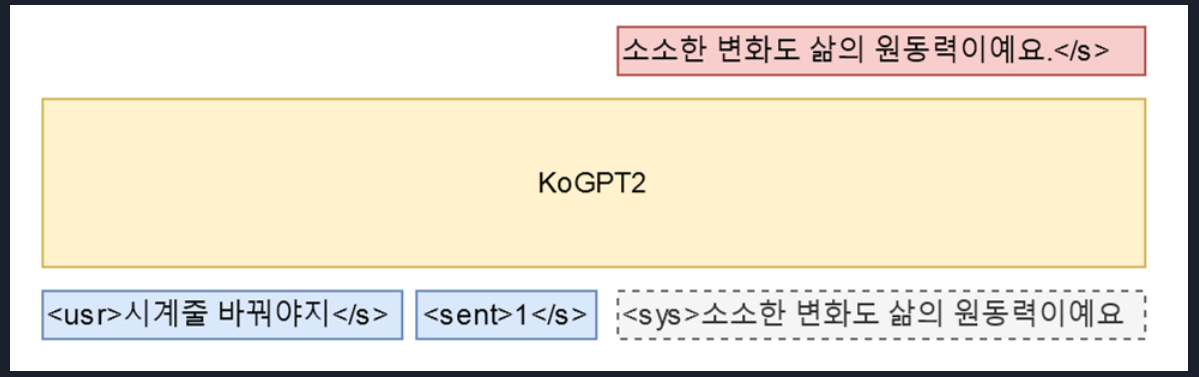
- **최적화 알고리즘 Optimizer**

손실 loss의 최소화 및 가중치 Weight 갱신
최적화 알고리즘 파라미터

➔ Adam (Adaptive Moment Estimation)
➔ Learning Rate

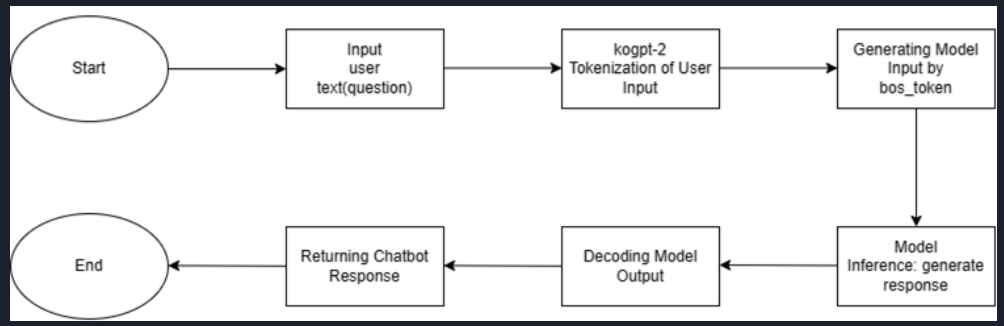
```
koGPT2_TOKENIZER = AutoTokenizer.from_pretrained('skt/kogpt2-base-v2', bos_token=BOS, eos_token=EOS, pad_token=PAD)
model = TFGPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2', from_pt=True)
adam = tf.keras.optimizers.Adam(learning_rate=3e-5, epsilon=1e-08)
```

```
from transformers import AutoTokenizer
from transformers import TFGPT2LMHeadModel
import tensorflow as tf
import pandas as pd
import tqdm
```



3. Project의 내용 (AI Model) - Class diagram

플로우 차트 Flow chart



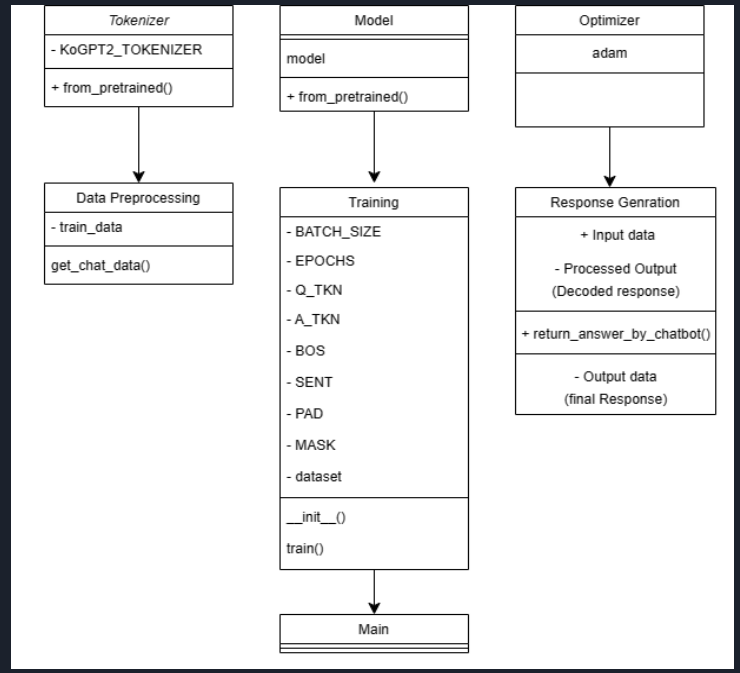
User input : 사용자 질문

Tokenization: 사용자 질문을 더 작은 단위로(문장→단어), 이후(단어→숫자)

Token: 모델의 출력을 구조화하고 정답 부분을 명시적으로 표시하는 데 사용.

Generate text: 입력된 질문 이후를 예측하여 문맥파악 후 답변 생성

클래스 다이어그램 Class Diagram



Tokenizer → Data Preprocessing Flow

: 토크나이저로 학습 데이터 사전 학습

Model → Training → Main Flow

: 사전 학습 데이터 기반 모델 훈련 및 생성

Optimizer →Response Generation Flow

: Adam 옵티마이저로 모델을 거쳐 답변 생성

3. Project의 내용 (AI Model) - 양자화 Quantization

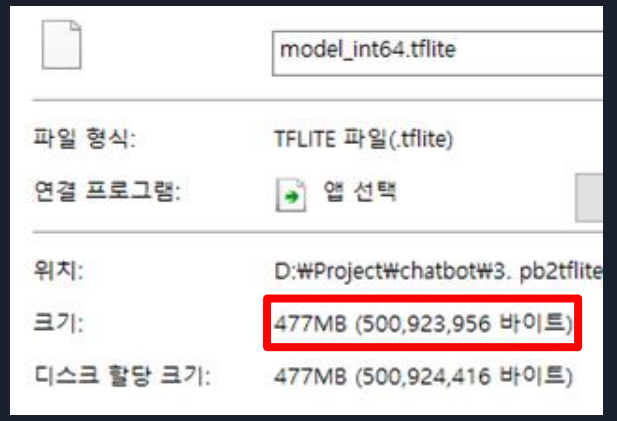
477 MB → 119MB Size 75.1 % ↓

문제점 Problem

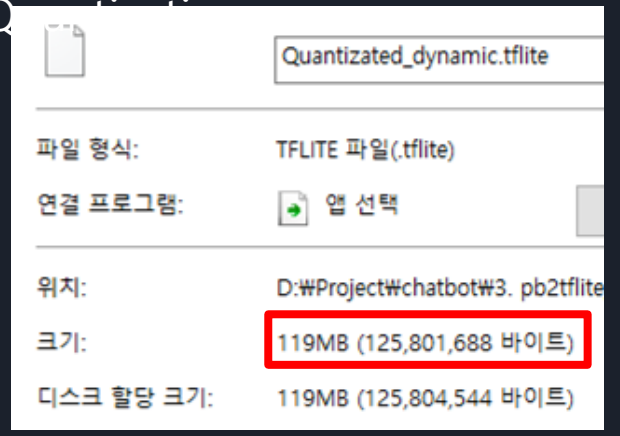
- 학습으로 인해 모델
- 모바일 환경에서의 구동 속도와 용량 문제를 야기
- 최적화를 위해 양자화를 적용

양자화 (Quantization)

- 인공신경망 모델에서 가중치와 활성화 함수를 더 작은 비트 수로 표현하도록 변환하는 기술
- 기존 모델의 크기를 줄이고, 계산 속도를 높이며, 낮은 메모리 사용 등 효율적인 모델 사용이 가능



양자화 전 / Before Q



양자화 후 / After Quantization

3. Project의 내용 (Android OS FrameWork)

Android OS

- CompileSDK 33 (안드로이드13버전)
- DefaultConfig : minSDK33, TargetSDK 33
- Java 17.0.5 Oracle OpenJDK Version
- build Gradle Ver 8.1.3

Firebase

- Realtime Database
- Storage
- Authentication

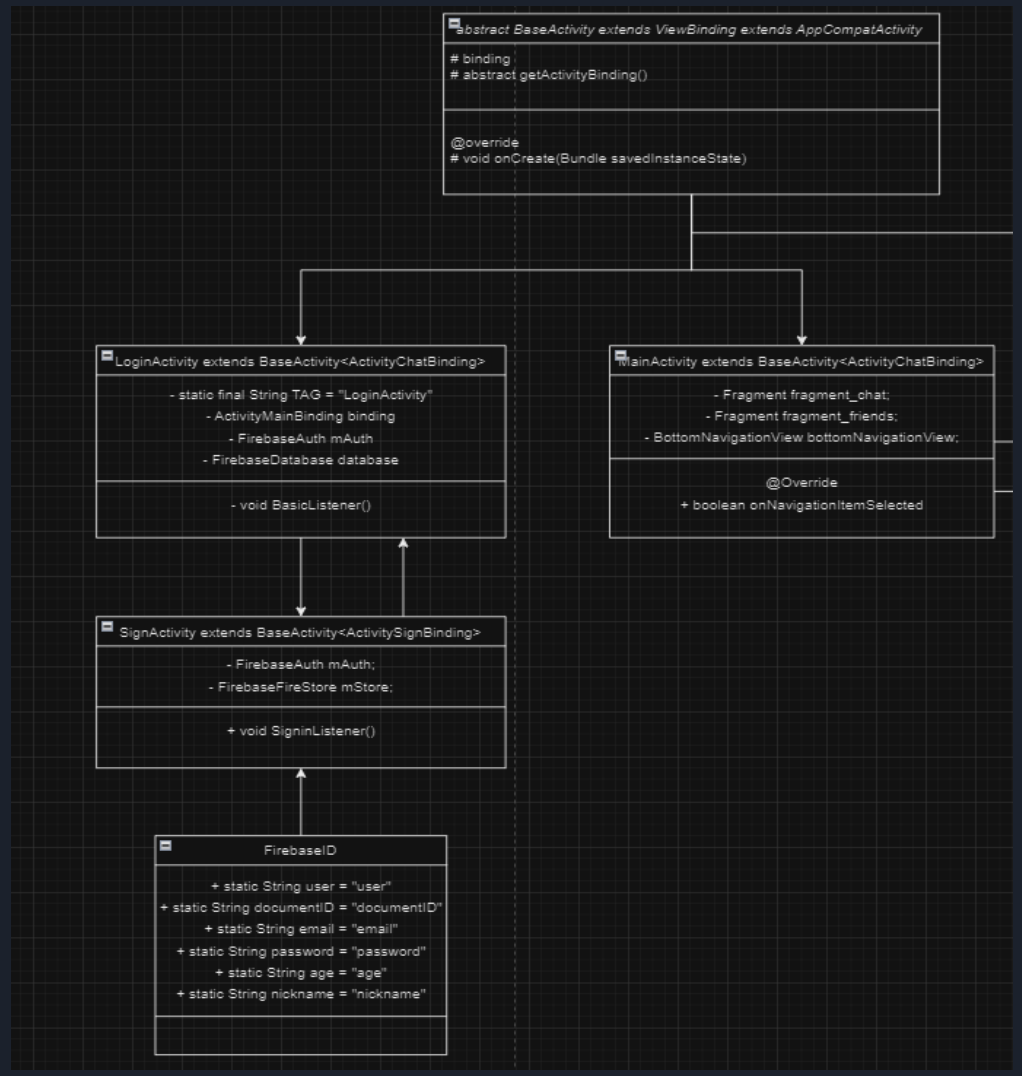
TensorFlow Lite

- Tensorflow-lite-task-text:0.3.0
- Tensorflow-lite-gpu:2.3.0

3. Project의 내용 (Android OS Class diagram)

Technology Stack

- Databinding : 레이아웃의 뷰를 앱 코드에 저장된 데이터와 연결하는 라이브러리. (MVVM 패턴)
- FirebaseAuth : Firebase 계정관리 기능
- FirebaseFireStore : Firebase 데이터베이스의 선택된 부분에 대한 엑세스를 제공하는 저장소
- MVVM 패턴 : Model, View, ViewModel 을 분리해 뷰에 모델간의 의존성을 줄여주도록 하는 기능.



3. Project의 내용 (Android OS Class diagram)

Technology Stack

- **RecyclerView & Adapter(ViewHolder)** :

  대량의 데이터 세트를 효율적으로 표시할 수 있는 라이브러리. 개발자가 데이터를 제공하고 각 항목의 모양을 정의하면 동적으로 생성할 수 있음. (Adapter를 통한 MVVM 패턴을 구현 할 수 있게 해줌.)

- **FirebaseDatabase** :

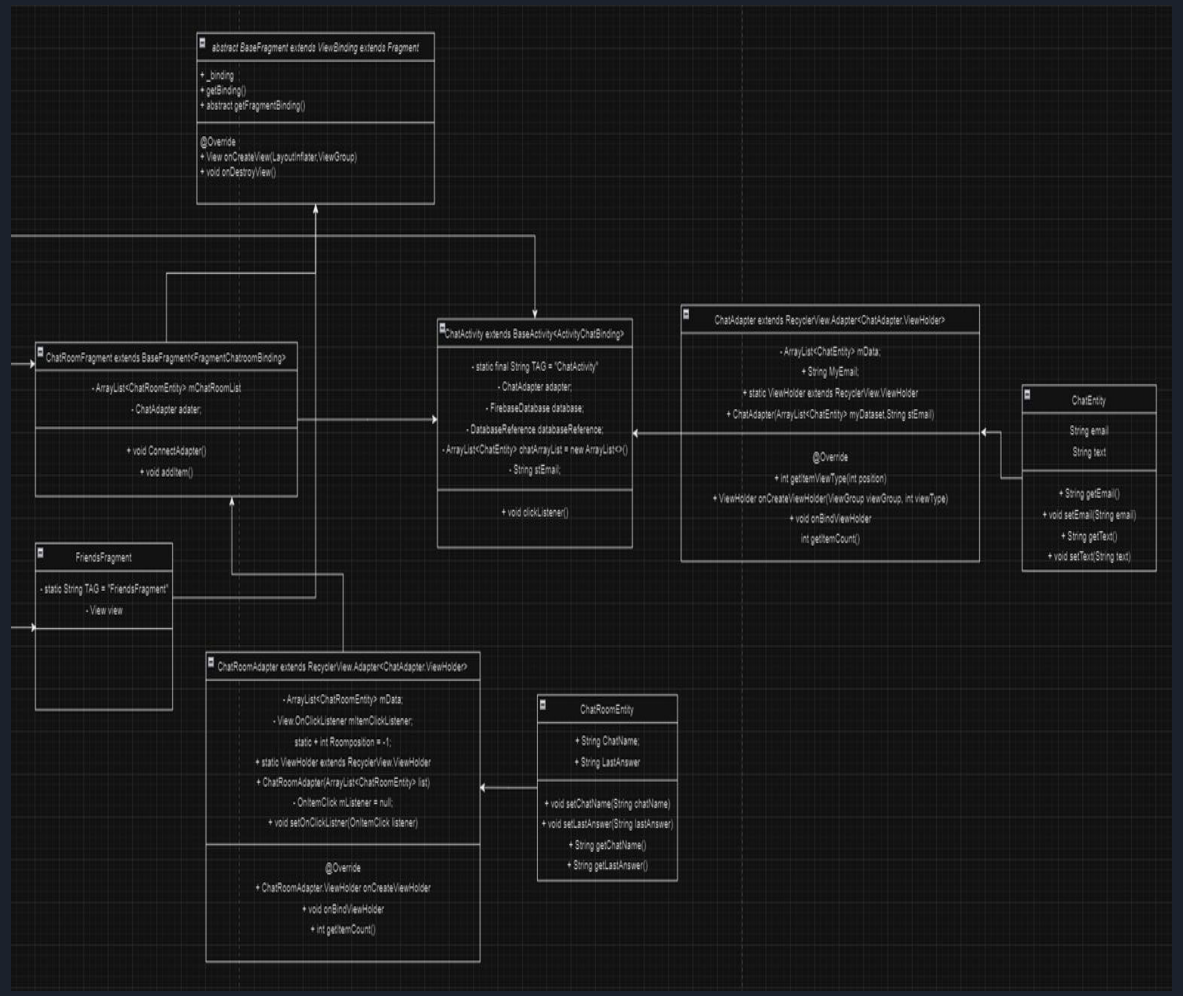
  앱에서 실시간 데이터베이스의 데이터를 읽고 쓰는 기능.

- **DatabaseReference** :

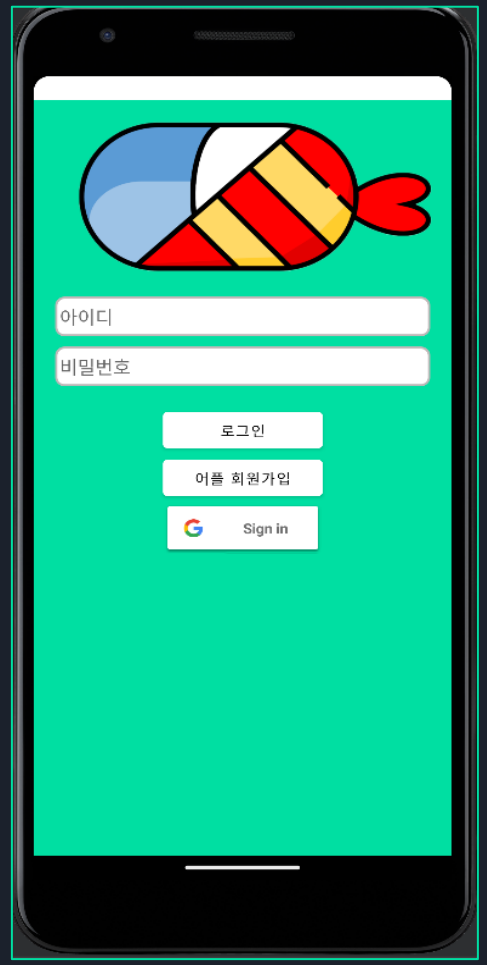
  데이터베이스의 특정위치를 나타내며 해당 데이터베이스 위치에 데이터를 읽거나 쓰는데 사용 될 수 있음.

- **Fragment** :

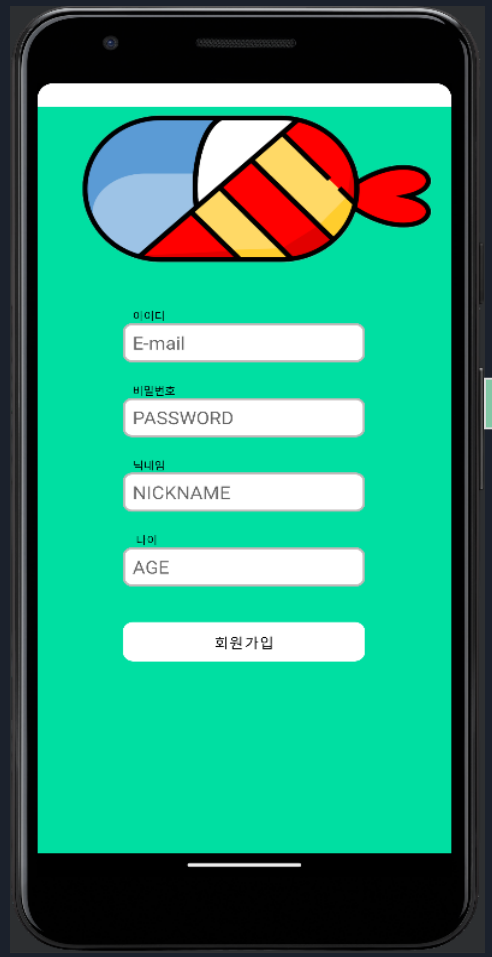
  사용자 인터페이스의 일부를 나타냄. 여러 개의 프래그먼트를 하나의 액티비티에 결합하여 창이 여러 개인 UI를 빌드 가능



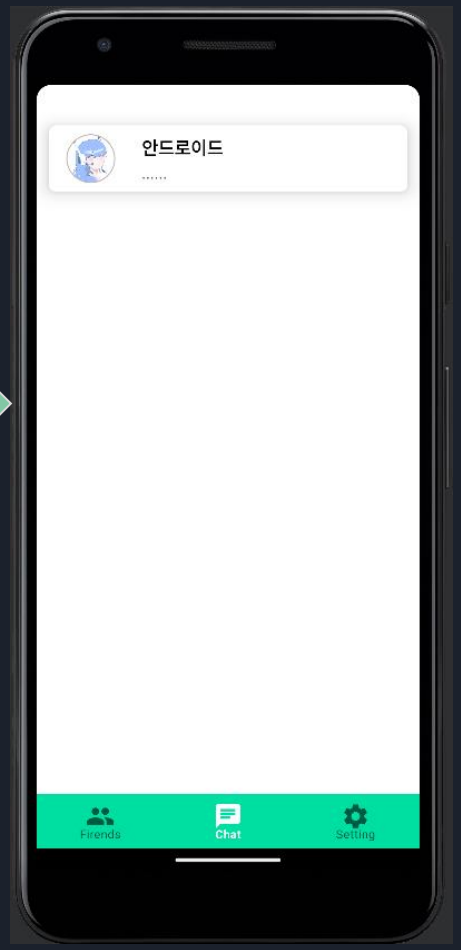
3. Project의 내용 (Android OS Scenario)



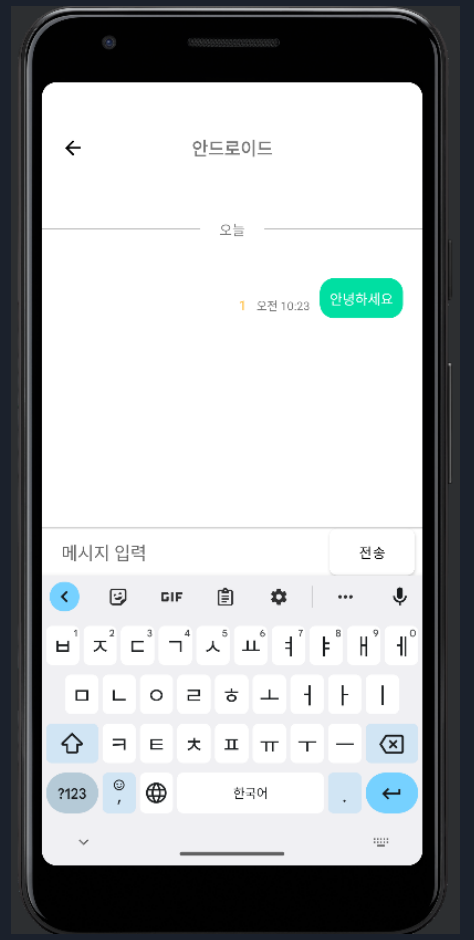
[로그인 화면]



[회원가입]



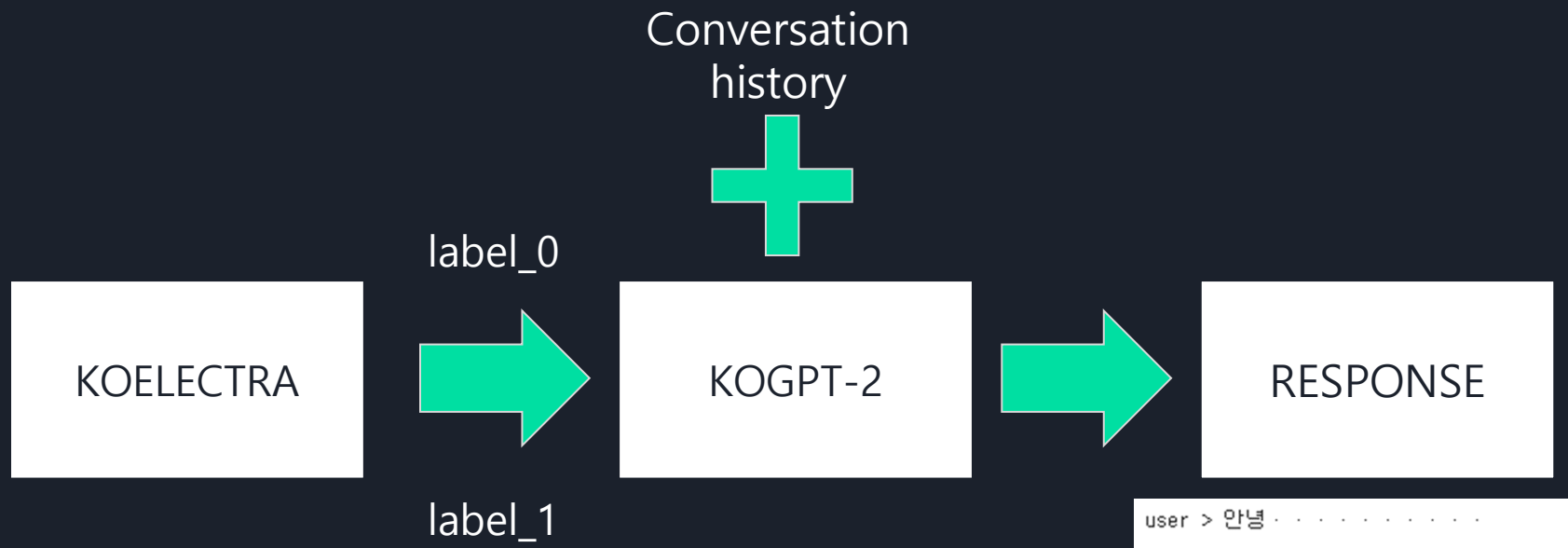
[채팅방목록]



[채팅창]

4. Project의 해결방안 및 수행과정

4. Project의 해결방안 및 수행과정 (Model Sequence)



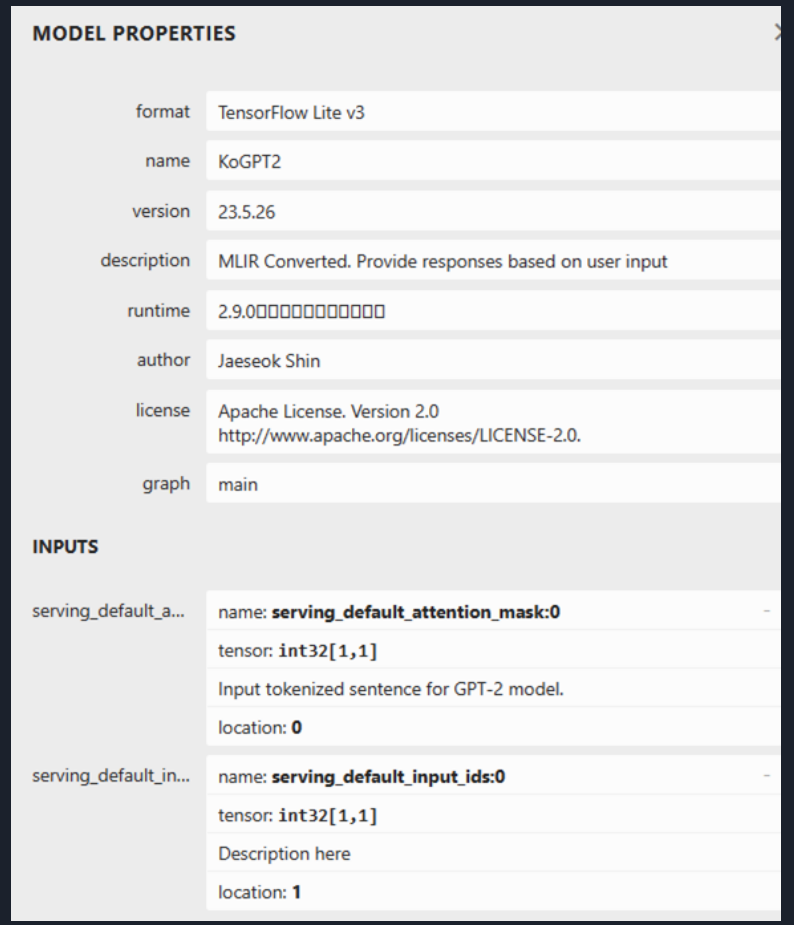
이쁘고 좋아요~~~씻기도 편하고 아이고 이쁘다고 자기방에 갖다놓고 잘써요~^^
>> {'label': '1', 'score': 0.9945501685142517}
아직 입어보진 않았지만 굉장히 가벼워요~~ 다른 리뷰처럼 어깡이 좀 되네요ㅋ 만족합니다. 엄청 빠른발송 감사드려요 :)
>> {'label': '1', 'score': 0.995430588722229}
재구매 한건데 너무너무 가성비인거 같아요!! 다음에 또 생각나면 3개째 또 살듯..ㅎㅎ
>> {'label': '1', 'score': 0.9959582686424255}
가습량이 너무 적어요. 방이 작지 않다면 무조건 큰걸로구매하세요. 물량도 조금밖에 안들어가서 쓰기도 불편함
>> {'label': '0', 'score': 0.9984619617462158}
한번입었는데 옆에 봉제선 다 풀리고 실밥도 계속 나옵니다. 마감 처리 너무 엉망 아닌가요?
>> {'label': '0', 'score': 0.9991756677627563}
따뜻하고 좋긴한데 배송이 느려요
>> {'label': '1', 'score': 0.6473883390426636}
맛은 있는데 가격이 있는 편이에요
>> {'label': '1', 'score': 0.5128092169761658}

user > 안녕
chatbudd > 안녕하세요
user > 너 이름이 뭐야
chatbudd > 위로봇
user > 내 이름은 고권표
chatbudd > 이름을 말하면 서로를 알 수 있어요
user > 내 이름
chatbudd > 고권표
user > 잘 지내니?
chatbudd > 기분이 좋아요 당신은요?

4. Project의 해결방안 및 수행과정 (메타 데이터)

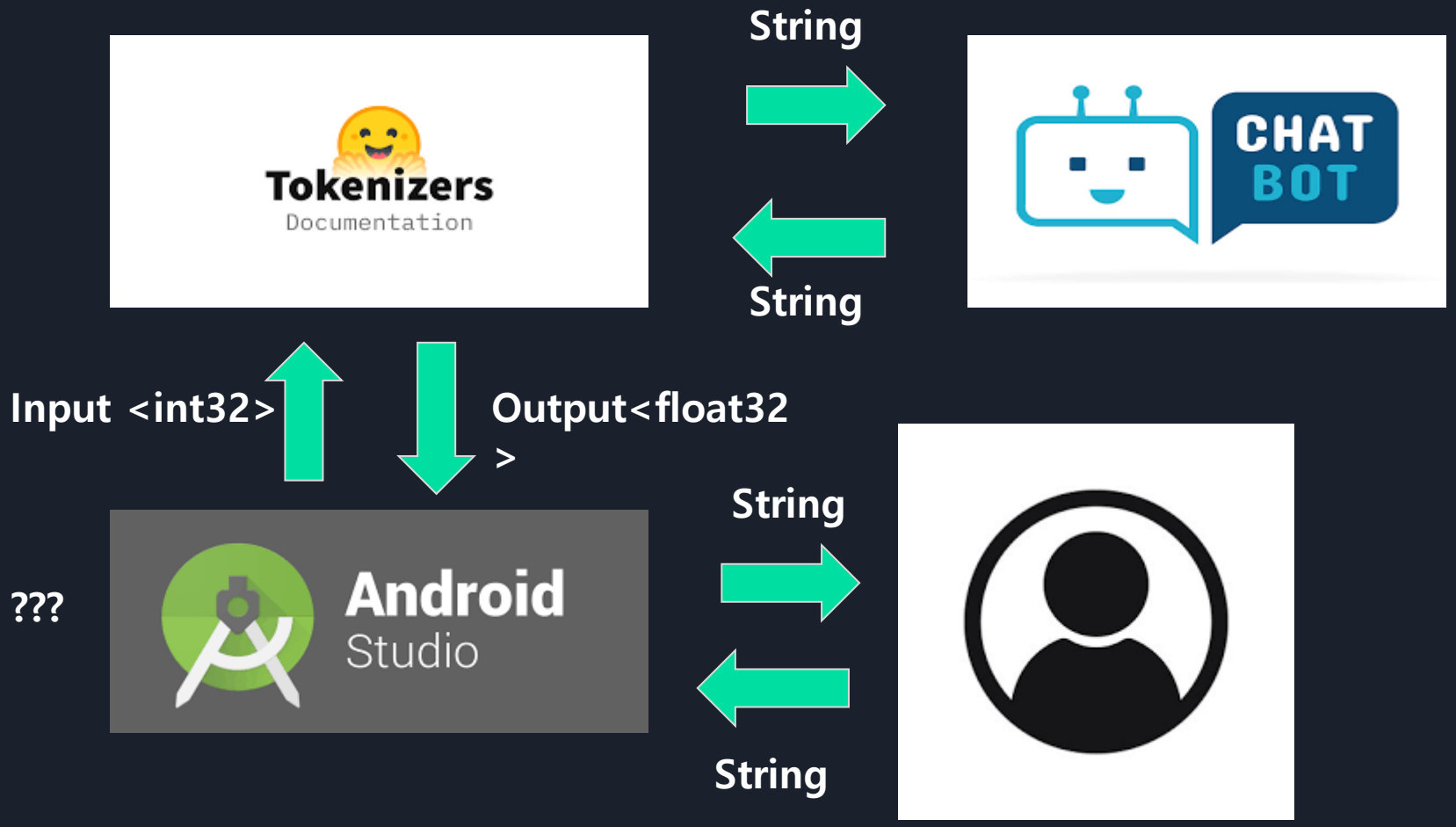
```python
#메타 데이터 시각화  ###
import os
from absl import flags

# Assuming model_path is the path to your original model file
model_path = "./Quantizated_dynamic.tflite"

# Get the base name of the model file (without directory path or extension)
model_basename = os.path.splitext(os.path.basename(model_path))[0]

displayer = _metadata.MetadataDisplayer.with_model_file(model_path)

# Specify the export directory path
export_directory = "./meta_applied_model/"


export_json_file = os.path.join(export_directory, model_basename + ".json")
json_file = displayer.get_metadata_json()

# Optional: write out the metadata as a json file
with open(export_json_file, "w") as f:
    f.write(json_file)
```

[메타 데이터 시각화 구현 코드]



[메타 데이터 시각화 결과]

4. Project의 해결방안 및 수행과정 (Android OS)



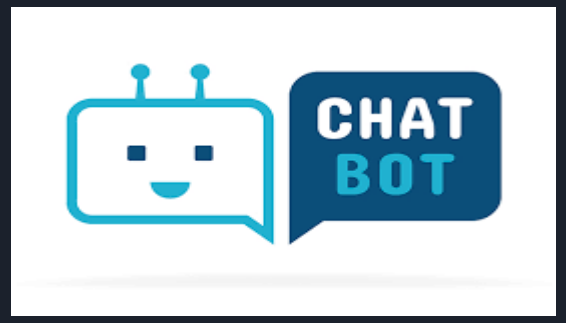
4. Project의 해결방안 및 수행과정 (Android OS)

```java
public List<Integer> encode(String text)
{
    Matcher matcher = pattern.matcher(text);
    List<String> unicodes = new ArrayList<>();
    List<Integer> bpeTokens = new ArrayList<>();

    while (matcher.find()) {
        String match = matcher.group();
        StringBuilder unicodeBuilder = new StringBuilder();
        for (byte b : match.getBytes(StandardCharsets.UTF_8)) {
            unicodeBuilder.append(this.byte2unicode.get((int)b));
        }
        unicodes.add(unicodeBuilder.toString());
    }

    for (String token : unicodes) {
        for (String bpeToken : bpe(token).split( regex: " ")) {
            bpeTokens.add(((BigInteger)encoder.get(bpeToken)).intValue());
        }
    }

    return bpeTokens;
}
```

```java
public String decode(List<Integer> tokens) {
    StringBuilder textBuilder = new StringBuilder();
    List<String> byteBufferList = new ArrayList<>();

    for (int token : tokens) {
        textBuilder.append(decoder.get(BigInteger.valueOf(token)));
    }
    String text = textBuilder.toString();

    for (int i = 0; i < text.length(); i++) {
        byteBufferList.add(byte2unicode.get((int)text.charAt(i)));
    }

    byte[] byteBuffer = new byte[byteBufferList.size()];
    for (int i = 0; i < byteBuffer.length; i++) {
        String byteString = byteBufferList.get(i);
        if (byteString == null) {
            byteString = " ";
        }
        byteBuffer[i] = (byte)byteString.charAt(0);
    }

    return Chars.asList(StandardCharsets.UTF_8
                    .decode(ByteBuffer.wrap(byteBuffer))
                    .array()) List<Character>
            .stream() Stream<Character>
            .map(String::valueOf) Stream<String>
            .collect(Collectors.joining());
}
```

5. 추후계획

5. 추후 계획



Thank you