[1p]

안녕하세요 저희는 온라인 조문 서비스 "마음길"을 구현한 ITPLE 김은지, 최윤녕입니다.

[2p]

발표는 이렇게 진행될 예정이고

[3p]

현재 병원에서는 조문글을 작성할 수 있는 시스템이 마련되어 있지만, 대부분이 단순한 텍스트 입력 방식으로 되어 있습니다. 예를 들어 화면에 보시는 것처럼, 고인과의 관계를 간단히 적고, 내용을 입력한 뒤 글을 남기는 방식입니다.

하지만 저희는 여기서 몇 가지 고민을 하게 되었습니다.

"텍스트로만 조문을 전한다면, 과연 그 사람의 진심이 제대로 전달될 수 있을까?"
"단순히 부의금을 계좌로 보내는 것으로 마음을 다 전할 수 있을까?"
"직접 찾아 뵙지 못한 미안한 마음은 어떻게 표현하고, 또 그 아쉬움을 어떻게 덜 수 있을까?"

이러한 질문들을 바탕으로, 저희는 조문이 단순한 '글 쓰기'를 넘어,
진심이 오갈 수 있는 하나의 '정서적 교류의 장'이 되어야 한다고 생각했습니다.

그래서 저희의 온라인 조문서비스 마음길은, 텍스트만으로는 한계가 있었던 조문 방식에
영상이라는 매체를 더하고, 온라인 상에서도 따뜻한 위로와 공감이 오갈 수 있는 새로운 조문 공간을 만들고자 했습니다.

[4p] 역할 분담

역할 분담은 이렇게 이루어졌고

[5p] 사용 기술 스택

저희 프로젝트는 안드로이드 기반 앱으로, **Kotlin 과 Android Studio** 를 사용해 프론트엔드를 개발했습니다.
영상 조문에 AI 기능을 적용하기 위해, **MediaPipe, OpenCV, Numpy** 등을 활용한 **On-device AI** 기능도 구현했습니다.

서버는 **AWS EC2** 위에 **Spring** 기반 백엔드를 올려서, 사용자 데이터와 조문 영상 정보를 **MySQL** 로 관리하고 있습니다.
영상 파일은 **AWS S3** 를 활용해 저장하고 불러오는 방식으로 구성했습니다.

[6p] 구현 기능 소개

저희가 구현한 주요 기능은 총 세 가지입니다.

첫 번째는 **흑백 처리된 영상 조문**입니다.
앱에서 촬영한 영상은 자동으로 조문객의 옷이 있을 아래 부분이 흑백 처리되어, 조문의 분위기를 전달할 수 있도록 했습니다.
이렇게 처리된 영상은 이후 상주에게 조문 메시지와 함께 전달됩니다.

두 번째는 **AI 기반 유해 언어 감지**입니다.
조문 글에 부적절한 표현이 포함되지 않도록, AI 가 자동으로 내용을 감지해 유해 언어를 차단합니다.

세 번째는 **부의금 송금 및 봉투 전달** 기능입니다.
카카오페이와 연동하여 부의금을 QR 코드로 간편하게 송금할 수 있고, 동시에 봉투 이미지도 함께 전달되어 실물 봉투와 같은 의미를 담을 수 있습니다.

[7p] 구현 기능 소개 –1

이 페이지는 **흑백 처리된 조문 영상 기능**의 작동 과정을 보여주는 화면입니다.

사용자가 앱에서 **촬영하기 버튼**을 누르면 영상 촬영이 시작되고,
녹화 중지 버튼을 누르면 영상 촬영이 종료됩니다.
이때 촬영된 영상은 자동으로 **흑백 필터가 적용**되며, 이후 영상은 자동으로 갤러리에 저장됩니다.

영상 처리는 OpenCV를 활용해서 구현했습니다.
녹화가 끝나면 영상에서 프레임을 추출하고, 각 프레임의 **아랫부분에만 마스크를 씌워서 흑백으로 변환**합니다. 그 후 이 프레임들을 다시 이어붙여 새로운 영상을 생성하고 저장합니다.
이렇게 하면 사용자가 영상에서 상단은 컬러 그대로 두고, 하단은 차분하게 흑백으로 강조할 수 있습니다.

저장된 영상은 앱 내에서 확인할 수 있고, **'나의 기록' 페이지**를 통해 상주에게 직접 조문 메시지를 전송할 수 있습니다. 이 과정을 통해 사용자는 단순한 텍스트가 아닌, 영상을 통해 진심이 담긴 조문을 전할 수 있도록 구성했습니다.

✅ 영상 아랫부분 흑백 처리 방식 설명

1. 전체 영상에서 프레임 추출

- `MediaMetadataRetriever`를 이용해서 일정 시간 간격으로 비디오의 **Bitmap 프레임 이미지**들을 추출함.
- 예: 1초에 30프레임 → 약 33ms 간격으로 하나씩 가져오기.

kotlin 복사 편집

```kotlin
retriever.getFrameAtTime(time * 1000L, MediaMetadataRetriever.OPTION_CLOSEST)
```

2. 프레임 이미지 처리 (OpenCV 사용)

프레임마다 아래 과정을 반복:

① RGBA → BGR 색 공간 변환

OpenCV는 BGR 형식을 기본으로 사용하기 때문에 변환 필요.

kotlin 복사 편집

```kotlin
Imgproc.cvtColor(rgbaMat, bgrMat, Imgproc.COLOR_RGBA2BGR)
```

② 원본 프레임 복사 → 흑백 버전 만들기

- 원본 프레임(BGR)에서 **흑백 이미지** 생성:

kotlin 복사 편집

```kotlin
Imgproc.cvtColor(bgrMat, grayMat, Imgproc.COLOR_BGR2GRAY)
```

- 그리고 다시 3채널로 확장 (왜냐면 영상은 3채널이어야 하니까)

kotlin 복사 편집

```kotlin
Imgproc.cvtColor(grayMat, grayBGRMat, Imgproc.COLOR_GRAY2BGR)
```

↓

③ 마스크 생성 (아랫부분만 흑백 처리하기 위해)

- 영상 높이의 절반 아래만 선택하는 흑백 마스크 생성:

kotlin 복사 편집

```kotlin
val mask = Mat(bgrMat.size(), CvType.CV_8UC1, Scalar(0.0))
mask.rowRange(height / 2, height).setTo(Scalar(255.0)) // 하단 절반만 활성화
```

④ 흑백 이미지에서 아랫부분만 추출

- `Core.bitwise_and()`로 흑백 이미지에서 **아랫부분만 분리**:

kotlin 복사 편집

```kotlin
Core.bitwise_and(grayBGRMat, grayBGRMat, maskedGray, mask)
```

⑤ 원본 + 블러 이미지 → 블렌딩

- 원본과 가우시안 블러 이미지를 70:30 비율로 섞어서 부드럽게 표현:

kotlin 복사 편집

```kotlin
Core.addWeighted(bgrMat, 0.7, blurredMat, 0.3, 0.0, blended)
```

⑥ 흑백 아랫부분을 블렌딩된 이미지에 덮어쓰기

- 마스크를 써서 아랫부분만 흑백으로 덮어쓰기:

kotlin 복사 편집

```kotlin
maskedGray.copyTo(blended, mask)
```

3. 다시 RGBA → Bitmap 으로 변환

- 최종 결과를 다시 `Bitmap`으로 변환해서 영상으로 합침.

kotlin 복사 편집

```kotlin
Utils.matToBitmap(resultRgba, resultBitmap)
```

🎤 발표 요약용 문장 예시

> "프레임마다 OpenCV를 이용해 아랫부분만 흑백으로 변환하는 마스크를 만들어 적용했습니다.
> 이렇게 하면 위쪽은 컬러를 유지하면서, 아래쪽은 흑백으로 차분하게 처리된 영상이 완성됩니다."

필요하면 실제 프레임 가공 흐름을 **간단한 도식**으로 보여줘도 좋아:

css 복사 편집

```
[컬러 프레임] → [흑백 변환] → [아래쪽 마스크 생성] → [흑백 부분만 덮어쓰기] → [최종 프레임]
```

[8p] 구현 기능 소개 - 2

이 페이지는 **유해 언어 감지 기능**을 소개하는 화면입니다.

사용자가 조문글을 작성하고 **전송 버튼을 누르면**, 해당 내용을 분석하여 욕설이나 부정적인 표현이 포함되어 있는지를 확인합니다.

사용자가 입력한 텍스트에서 욕설이나 부정적인 표현이 포함되어 있는지 확인하는 기능을 구현했는데, 욕설 단어 리스트는 JSON 파일로 관리하고, 앱 실행 시 메모리로 로드됩니다.
입력된 텍스트에 해당 단어가 포함되면 자동으로 게시가 되지 않습니다.
이를 통해 조문공간에 올라가는 메시지의 품격을 유지할 수 있도록 했습니다.

가운데 보시는 curse 안의 단어들처럼, 사전에 정의된 **비속어 리스트**와 비교해 유해한 단어가 포함되어 있으면 글은 등록되지 않고, 오른쪽 아래와 같이 "게시에 실패하였습니다"라는 메시지가 표시됩니다.
반면 문제가 없는 경우에는 정상적으로 업로드되며, "게시에 성공하였습니다"라는 알림이 뜹니다.

2. 입력된 텍스트 검사 (`filterText()` 함수)

- 사용자가 입력한 텍스트(`message`)를 받아서, 저장된 욕설 리스트와 비교함.
- 욕설 단어가 포함되어 있으면 해당 단어를 `*`로 대체함.
  (예: "씨발이야" → "***이야")

kotlin 복사 편집

```kotlin
if (filteredText.contains(curse, ignoreCase = true)) {
    filteredText = filteredText.replace(curse, "*".repeat(curse.length))
    detected = true
}
```

- 결과는 `Pair<필터링된텍스트, 감지여부>`로 반환됨.

kotlin 복사 편집

```kotlin
return Pair(filteredText, detected)
```

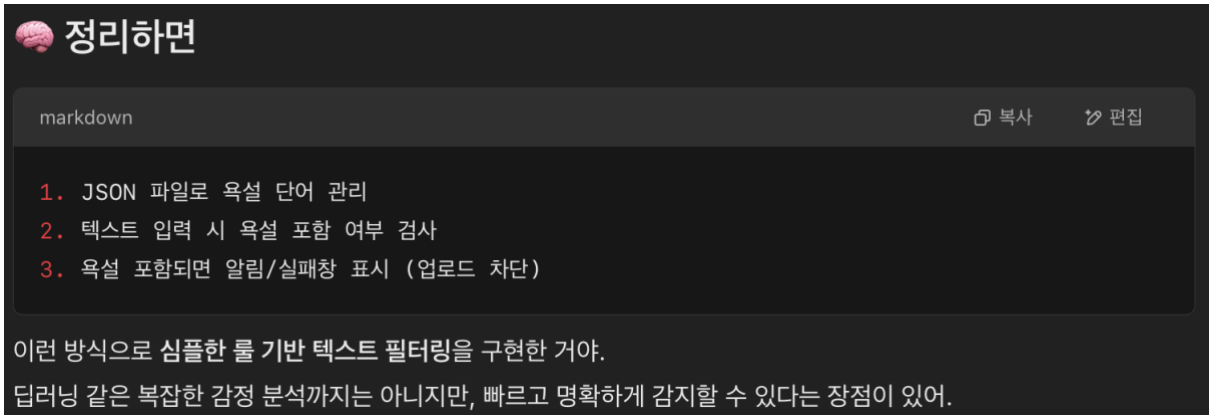
3. 감지 결과에 따라 UI 흐름 분기

- `Archieve1Fragment`에서 텍스트 제출 버튼 클릭 시 `filterText()` 호출.
- 욕설이 감지되면 `ArchieveFailureFragment`를 띄워서 메시지를 거절함.
- 욕설이 없으면 `Archieve2Fragment` → `ArchieveSuccessFragment`로 이어짐.

kotlin 복사 편집

```kotlin
val (_, detected) = CurseWordDetector.filterText(message)

if (detected) {
    // 실패 처리
    val failureFragment = ArchieveFailureFragment.newInstance(...)
    failureFragment.show(...)
} else {
    // 성공 처리
    val processingFragment = Archieve2Fragment.newInstance(...)
    processingFragment.show(...)
}
```



[9p] 구현 기능 소개 - 3

마지막으로 **부의금 송금 및 봉투 전달** 기능입니다.

먼저, 사용자가 부의금 금액을 입력하면, 해당 금액과 함께 **디지털 부의 봉투 이미지**가 생성됩니다. 이후, **카카오페이 결제 API 를 활용해 QR 결제 페이지를 생성**합니다.

현재는 사업자 등록이 되지 않은 개인 프로젝트이기 때문에 카카오페이의 정식 결제 모듈을 직접 연동하는 대신, **테스트 결제 환경에서 QR 결제 URL 을 생성**하고, 이를 통해 **다른 기기에서 결제할 수 있도록 설계**했습니다. 공기계에서는 직접 결제를 진행할 수 없기 때문에, **QR 코드를 생성하고 사용자가 개인 스마트폰으로 스캔하여 결제**할 수 있도록 구조를 구성했습니다.

결제가 성공적으로 완료되면, 서버에서 결제 결과를 확인한 뒤 **부의금 전송 완료 메시지와 함께 봉투 이미지가 상주에게 전달**되며, 조문과 함께 마음을 전할 수 있도록 했습니다.

[10p] 초기 공통 시나리오

다음으로 전체적인 앱 실행 시나리오를 설명 드리겠습니다.

먼저 앱 실행 시 상주와 조문객, 모든 사용자들이 거치는 **초기 공통 시나리오**입니다.

앱을 실행하면 가장 먼저 **홈화면**이 나타나고, 로그인 또는 회원가입 버튼을 통해 사용자 인증 절차를 진행합니다. 회원가입 시에는 이름, 전화번호, 비밀번호를 입력하고, 해당 정보는 서버에 저장되어 추후 로그인 시 활용됩니다.

로그인이 완료되면 **메인 화면으로 이동**하게 되며, 여기서 영상 조문을 위한 **촬영 기능**이나 **조문 공간 생성·참여**, **부의금 송금** 등의 주요 기능들을 사용할 수 있습니다.

[11p] 상주 시나리오

이번에는 **상주가 조문 공간을 생성하는 시나리오**입니다.

먼저, 상주는 고인의 이름과 성별, 발인일시, 장례식장 위치 등의 정보를 입력합니다. 고인을 상징할 수 있는 **대표 이미지도 함께 업로드**할 수 있도록 했습니다.

이후, 조문 공간을 관리할 상주의 정보를 입력하고, 조문객들이 조문 공간에 접근할 수 있도록 **초대 링크와 인증번호**가 자동으로 생성됩니다.

마지막 화면처럼, 조문 공간이 성공적으로 생성되면 해당 내용을 담은 **초대 메시지 형식**으로 공유할 수 있으며, 이를 통해 상주가 직접 만나지 못하는 지인들에게도 **온라인을 통해 조문 공간을 열어 안내**할 수 있도록 구현했습니다.

[12p] 상주 시나리오

이 페이지는 **상주가 조문 공간을 생성한 이후, 이를 어떻게 관리하고 소통하는지를 보여주는 시나리오**입니다.

먼저, 마이페이지에서는 생성한 조문 공간을 확인할 수 있고, 각 공간에 대해 **조문 메시지 관리**와 **부의금 관리** 메뉴로 이동할 수 있습니다. 공유 탭을 통해 **카카오톡, 문자, SNS 등으로 링크를 전송**하여 조문객들에게 조문 공간을 간편하게 안내할 수 있습니다.

메시지 관리 화면에서는 조문객들이 남긴 영상이나 텍스트 메시지를 보시는 것과 같이 한눈에 확인할 수 있고, 영상도 직접 재생해볼 수 있도록 구성했습니다.

마지막으로 부의금 관리 탭에서는, **어떤 조문객이 얼마를 보냈는지, 전체 금액 대비 비율은 어떻게 되는지**를 정리해서 보여주고, 보낸 사람 별로 **디지털 부의 봉투 이미지도 함께 저장**되도록 했습니다. 이 모든 기능은 **직접 만나지 못한 상황에서도, 상주가 조문을 정리하고 감사함을 전할 수 있도록** 직관적이고 정돈된 형태로 제공하는 것을 목표로 했습니다.

[13p] 조문객 시나리오

다음으로 **조문객의 이용 시나리오**입니다.

조문객은 앱에서 먼저 조문 영상을 촬영한 뒤, **초대받은 고인의 이름과 인증번호를 입력하여 조문 공간에 접속**할 수 있습니다.

조문 공간에 입장하면, 고인과 장례식장 정보, 상주의 안내 메시지를 확인할 수 있고, 아래의 아카이브 작성 버튼을 눌러 **조문 메시지를 남기게 됩니다**.

조문 메시지는 단순 텍스트 외에도, 영상 조문이나 감사 메시지를 함께 담을 수 있도록 구성되어 있습니다. 전송을 누르면 **필터링을 거쳐 부적절한 표현이 없는지 확인**되고, 문제가 없을 경우에는 정상적으로 게시됩니다.

이와 같이 **조문객은 실물 장례식에 참석하지 못해도, 진심을 전하고 추모할 수 있는 환경을 갖출 수 있도록** 했습니다.

[14p] 조문객 시나리오

이제 조문 공간에 접속한 조문객은 **텍스트 메시지와 영상 파일을 함께 첨부**하여 조문 메시지를 전달할 수 있습니다. 그 후, 부의금 송금 페이지로 이동하면 송금할 금액을 입력한 뒤, 마음을 담은 **디지털 부의 봉투 디자인을 선택**할 수 있도록 구성했습니다. 선택이 완료되면, **카카오페이 결제 API 를 통해 생성된 QR 코드**를 통해 결제를 진행할 수 있습니다.

이를 통해 조문객은 메시지와 부의금을 함께 전달함으로써, 직접 조문을 가지 못한 상황에서도 딸랑 돈만 보내는 것이 아니라 **정중한 애도와 마음을 전할 수 있는 방식**으로 설계되었습니다.

[15p] 조문객 시나리오

마지막으로 **조문객이 남긴 조문 기록을 마이페이지에서 확인하고 관리하는 흐름**입니다.

앱 우측 상단의 마이페이지에 들어가면, 조문객은 **자신이 참여한 조문 공간의 메시지와 부의금 기록**을 한눈에 확인할 수 있습니다.

메시지 탭에서는 직접 남긴 **조문 영상이나 글을 다시 확인**할 수 있고, 부의금 탭에서는 **전달했던 디지털 봉투 이미지와 송금 금액**도 함께 저장되어 있어 자신의 조문 내역을 추후에도 되돌아볼 수 있도록 구성했습니다.

(이 기능은 단순한 송금 기록을 넘어서, **내가 전했던 마음의 흔적을 하나의 기억으로 간직할 수 있게** 한다는 점에서 의미가 있습니다.)

[16p] 서비스 시연

마지막으로 마음길 서비스 시연 영상입니다.

1. 영상 촬영
2. 조문공간검색
3. 아카이브
4. 분석 후 부정적 텍스트 감지
5. 위로의 말과 함께 영상 전송
6. 나의 기록에서 확인 가능
7. 상주선택해서 부의금 전송
8. 이것 또한 나의 기록에서 확인 가능

1. 조문공간 관리(상주)
2. 조문객들이 보낸 메시지들 확인 가능
3. 부의금 관리도 가능

1. 회원가입 후 로그인
2. 고인의 정보들 입력해서 조문 공간 생성
3. 상주 정보 입력
4. 마이페이지에서 조문 공간 관리

✅ 예상 질문 1. AI 유해 언어 필터링은 어떤 방식으로 작동하나요?

> 단어 필터 기반인가요? 머신러닝 기반인가요?
> 새로운 욕설이나 변형된 표현엔 어떻게 대응하나요?

💬 답변 예시:

> 현재 앱에서는 성능과 구현 난이도를 고려해 **욕설 단어 필터 기반**으로 텍스트를 검사합니다.
> 욕설 리스트는 JSON 파일로 관리되고, 사용자가 작성한 메시지를 하나하나 비교해서 일치하는 단어가 있는 경우 차단합니다.
>
> 다만, Python에서 초기 연구 단계에서는 **형태 변화, 철자 변형, 초성·중성·종성 분해, 그리고 코사인 유사도 기반의 벡터 비교, 심지어 음운 유사도까지 고려한 고도화된 필터링**을 구현했습니다.
> 변형된 표현(예: "ㅅㅂ", "씨빨", "ㅅ이발")도 일정 유사도 이상이면 감지할 수 있었고, false positive는 줄이면서 감지는 잘 되는 구조였습니다.

💡 보충:

> 이런 고도화된 버전은 연산량이 많아 안드로이드 실시간 처리에 부하가 있어서 현재는 단순 키워드 필터를 사용하고 있고, 추후에는 서버 기반 필터나 머신러닝 적용도 고려 중입니다.

✅ 예상 질문 2. 영상 흑백 처리나 필터는 실시간으로 적용되나요? 성능 이슈는 없었나요?

> 공기계나 저사양 기기에서도 문제없이 작동하나요?

💬 답변 예시:

> Android 버전에서는 촬영이 끝난 후 프레임을 분해해서 후처리로 필터를 적용합니다.
> 그래서 실시간은 아니지만, 안정적으로 효과를 줄 수 있습니다.
>
> 초기엔 Python(OpenCV + MediaPipe) 기반으로 **실시간 배경 흐림 + 하단 흑백 처리**까지 구현했었습니다.
> 웹캠 프레임을 받아 `Mediapipe SelfieSegmentation`으로 사람 영역을 분리하고, 실시간으로 배경 블러 + 하단 흑백 처리까지 되는 구조였습니다.
>
> 하지만 Android로 이식하면서 성능과 구현 복잡도 이슈로 실시간 효과는 생략하고, 녹화 후 처리 방식으로 대체했습니다.
> 추후 GPU 활용이나 MediaPipe Android 버전을 적용해 실시간 처리도 고려하고 있습니다.

✅ 예상 질문 3. **영상 데이터는 어디에 저장되며, 보안은 어떻게 관리되나요?**

> 개인정보 보호나 민감한 영상 데이터의 접근 제한은 어떻게 구현했나요?

💬 답변 예시:

> 영상은 기본적으로 앱 내부 저장소에 저장되며, 처리 후 사용자의 갤러리(MediaStore)를 통해 접근 가능합니다.
> 외부 서버로는 전송되지 않고, 오직 디바이스 안에서만 처리되어 개인정보 유출 가능성은 없습니다.
>
> 안드로이드의 Scoped Storage 정책을 따라 외부 앱은 앱 내부 저장 파일에 접근할 수 없습니다.
> 또한, 앱에서 영상 접근을 할 때는 반드시 사용자 권한을 확인하고 처리하도록 되어 있습니다.
>
> 추후 서버 저장 기능이 추가된다면, **암호화 저장 + 접근 제어** 정책을 추가할 계획입니다.

✅ 왜 Android에서는 단순화되었는가?

💬 설명 포인트:

- Android는 모바일 환경이라 실시간 영상 처리나 딥러닝 기반 탐지가 성능 이슈 발생 가능
- Python은 데스크탑에서 작동 + OpenCV/MediaPipe 등 강력한 실시간 처리 지원
- 모바일에서는 최적화를 위해 단순한 키워드 필터링 + 녹화 후 영상 처리 구조로 구현

🧠 [1] Python에서 구현한 영상 처리

✅ 핵심 기능

- 실시간 영상 처리 (웹캠 스트리밍 기반)
- 사람 인식 (Mediapipe SelfieSegmentation) 으로 사람과 배경 분리
- 배경 흐림 처리 (Gaussian Blur)
- 영상 하단 50% 흑백 변환

✅ 처리 과정

1. 웹캠 프레임을 실시간으로 읽음
2. Mediapipe를 통해 사람 영역 마스크 생성
3. 배경 부분만 흐림 적용 (마스크 활용)
4. 하단 50% 흑백 변환 처리
5. 결과를 실시간으로 화면에 출력

✅ 장점

- 영상 보면서 바로 효과 확인 가능
- 사람만 컬러, 배경은 흐림 + 하단은 흑백 → 감정적 분 , 강조

📱 [2] Android에서 구현한 영상 처리 (간단화된 버전)

✅ 핵심 기능

- 녹화 후 처리 (실시간 아님)
- OpenCV로 프레임 단위 처리
  - 가우시안 블러 + 하단 흑백 처리만 적용
- Mediapipe 없음 (사람/배경 분리 불가)

✅ 처리 과정

1. 영상 촬영 완료 후 프레임 추출
2. 프레임마다:
   - 전체 블러 적용
   - 하단 마스크 생성 → 흑백 변환
3. 모든 프레임 처리 후 영상으로 다시 합침

✅ 간단화된 이유

- Android 환경에서는 실시간 처리 성능이 부족함
- Mediapipe Android는 세팅 복잡 + 기기 호환성 이슈 있음
- 실시간 대신 안정적인 후처리 방식 선택

🧠 [3] Python에서 구현한 부정적 텍스트 필터링

✅ 핵심 기능

- 단순 욕설 매칭 외에:
  - 초성/중성/종성 분해
  - 비슷한 음운(사운드) 감지
  - 코사인 유사도 기반 벡터 유사성 분석
  - 문자 변형 (예: ㅅㅂ → 씨발) 대응

✅ 예시:

"ㅅㅂ", "씌벌", "뒤져라 개새끼야" → 자동 탐지 및 마스킹 (`***` 처리)

📱 [4] Android에서 구현한 욕설 필터링 (간단화된 버전)

✅ 핵심 기능

- `curse.json` 에 있는 **고정된 욕설 리스트** 기반 단순 매칭
- 입력된 메시지에서 해당 단어가 있으면 `***` 로 치환하고 게시 차단

✅ 간단화된 이유

- Python 필터는 벡터 연산 + 유사도 비교로 연산량 큼
- Android에서는 실시간 입력 처리 성능, 메모리, 용량 이슈 있음
- 간단한 **"있는지만 보는 키워드 기반 감지"** 방식으로 구현

🧩 비교 요약

기능	Python 구현	Android 구현
🎥 영상 처리 방식	실시간 처리 (웹캠, Mediapipe)	녹화 후 후처리
사람/배경 분리	O (SelfieSegmentation)	X
흑백/블러	O (실시간)	O (프레임 후처리)
⚠️ 욕설 감지 방식	유사도 기반, 초/중/종 분석, 음운 분석	고정된 단어 리스트 비교
변형 대응	O (벡터 유사도, 발음 유사성)	X (단어 포함 여부만)
처리 성능	데스크탑 기준	모바일 저사양 기기 고려

🔚 결론

> Python에서는 고성능 환경을 활용해 복잡하고 정교한 영상/텍스트 분석을 실시간으로 수행했지만,
> Android에서는 성능 및 제한사항을 고려해 안정적인 **후처리 기반 영상 필터링**과 **단어 기반 필터링**으로 기능을 간소화했습니다.
> 이는 사용자 경험과 성능 사이의 **실용적 균형**을 고려한 설계입니다.