차량 파손 수리비, 더 이상 기다리지 마세요

쉽고 빠른 견적, 3분 견적

S사 차량 파손 데이터를 이용한 차량 파손 견적과 손상도 측정 어플

4인 프로젝트 | 2023. 12. 05 - 2024. - 01. 17
김영준 강동훈 김민지 장다연

Project Contents

프로젝트 배경

S사 비즈니스, 지출구조, 시장 분석과 전략 방향

S사 비즈니스

2012 카셰어링 서비스 S사 출시

2018 V사 모빌리티 플랫폼 출시

2019 카셰어링 멤버십 'S사패스'(현 패스포트) 출시

2021 전기차 카쉐어링 도입

2022 맞춤형 구독상품, 업무용 장기랜트 상품 출시

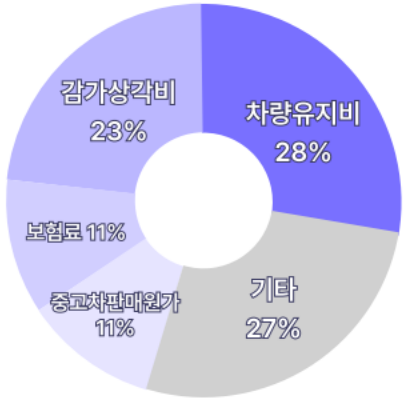
2022 S사 - KTX 묶음예약 출시
S사 패스포트 국제선 상시할인

2023 숙박연계 'S사스테이' 출시

단순 카쉐어링 서비스 넘어 교통, 숙박 아우르는 **멀티 플랫폼**으로 성장

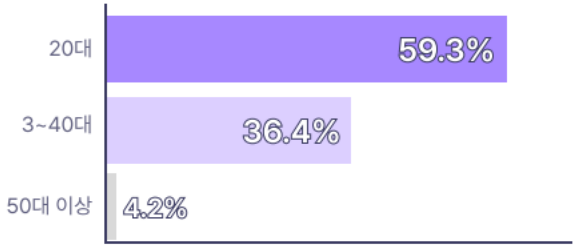
→ 다양한 기종의 **차량 관리비 / 수리비 증가 예상됨**

지출구조



S사 지출예산 중 **차량유지비**가 가장 많은 비중을 차지

시장 분석과 전략 방향



S사 서비스 이용 연령 분포도
(2020 S사 기업 블로그)

S사 G사 T사

카쉐어링 3사 모바일 어플 부정적 리뷰 순위
(2023. 11 ~ 2024. 1 앱스토어)

1. 차량유지비 확보 위해 불필요한 **인건비 절감** 필요
2. 주요 서비스 이용자 손상별 **차량보험료 개별 측정** 필요
3. 차량 사고 발생 시 고객센터 외 **빠른 피드백 요구 증가** 수용

프로젝트 배경

프로젝트 목적, 개요와 구조, 기대 효과

프로젝트 목적

Why

전략 방향에 따른
인건비 절감과 향상된 서비스 제공 필요

How

대략적인 **수리비 예측 모델** 통해
고객 요구사항과 데이터 수집 만족

So

차량 파손 견적 계산 어플 제작/공급

프로젝트 개요와 구조

Item

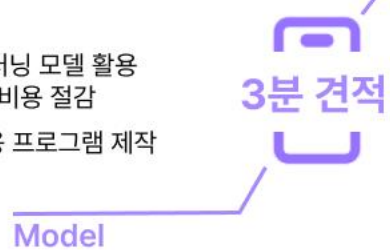
운전자가 직접 촬영한
차량 파손 이미지 데이터를 이용

차량 파손 부위 수리견적 실시간 출력

Structure

외부 머신러닝 모델 활용
시간 / 비용 절감

Python 이용 프로그램 제작



Figma 이용한 UI 제작

Android studio 기반한
기본 모델 구성

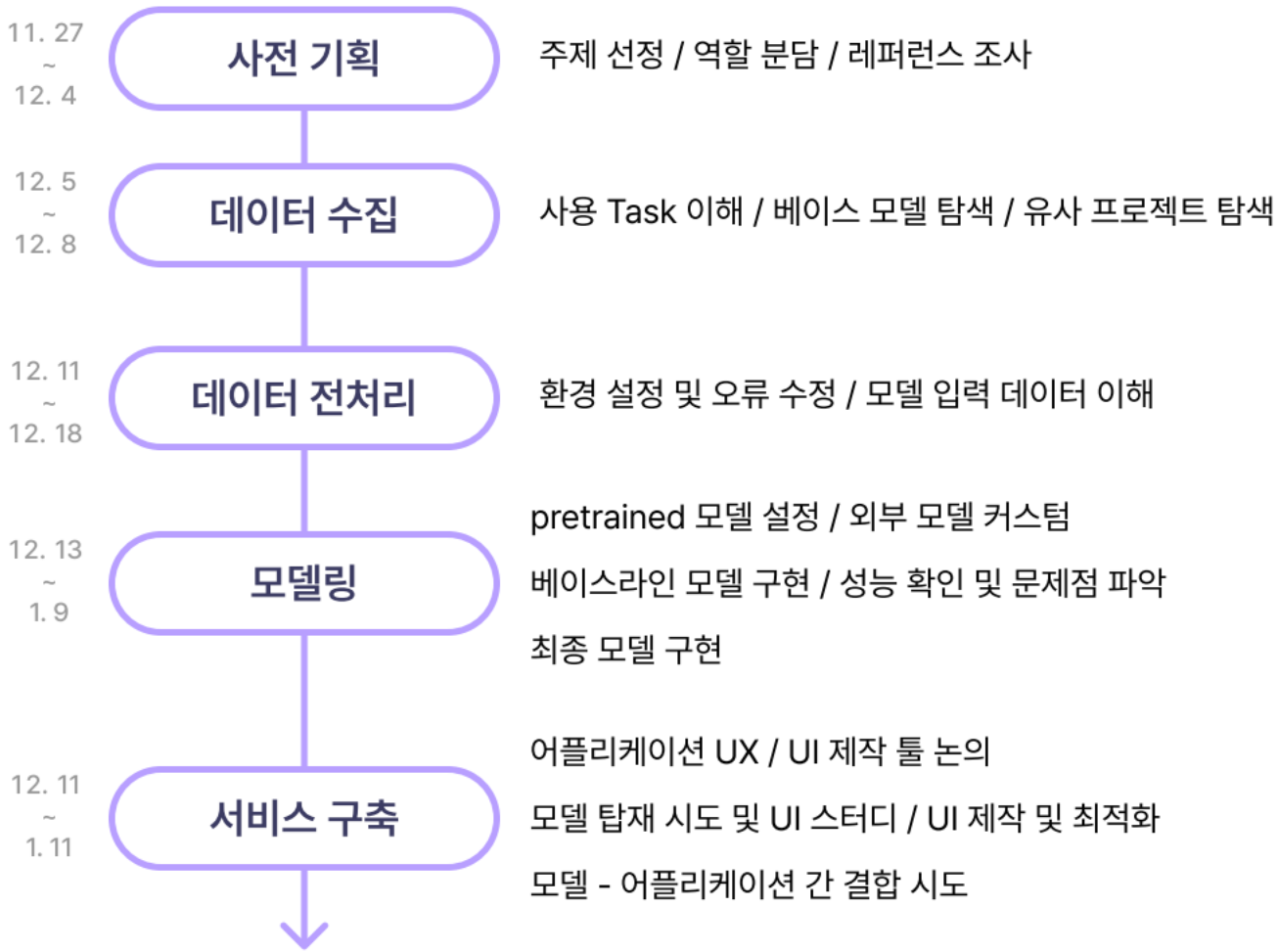
기대효과

Business

1. 차량 수리비 견적 공지 위한 불필요한
   A/S 인건비 절감
   - **차량유지비 확보, 효율적인 인력 운영**
2. 차량 파손별 손상 정도 데이터 수집
   - **예산 편성, 차량 보험료 산정 도움**
3. 어플 사용을 통한 빠른 피드백으로
   긍정적 경험 유도
   - **고객 이탈 방지, 서비스 재이용률 증가**

프로젝트 수행 과정

사전 기획, 수행 및 완료 과정



총 개발 기간 12. 11 ~ 1. 11 (총 4주)

프로젝트 결과 - 모델

알고리즘 개요

사용 모델과 라이브러리

1. 파손 유형별 추출 모델 사용 (모델 및 데이터 출처 - 쏘카)
   - [DAMAGE][Breakage_3]Unet.pt : 파손
   - [DAMAGE][Crushed_2]Unet.pt : 찌그러짐
   - [DAMAGE][Scratch_0]Unet.pt : 스크래치
   - [DAMAGE][Seperated_1]Unet.pt : 이격
2. PyTorch 및 Unet 사용

이미지 로드 방식과 전처리

1. 이미지 로드 방식 코드

```
img  = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (256, 256))

plt.figure(figsize=(8, 8))
plt.imshow(img)
```

2. 이미지 전처리 코드

```
img_input = img / 255.
img_input = img_input.transpose([2, 0, 1])
img_input = torch.tensor(img_input).float().to(device)
img_input = img_input.unsqueeze(0)

img_input.shape
```

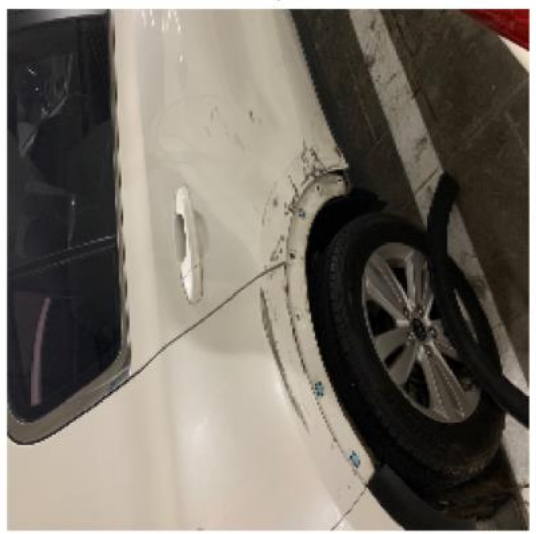
모델 추론 및 마스크 시각화

1. 모델 추론 및 마스크 시각화 코드

```
[ ]  output = model(img_input)

     output.shape

[ ]  img_output = torch.argmax(output, dim=1).detach().cpu().numpy()
     img_output = img_output.transpose([1, 2, 0])

     plt.figure(figsize=(8, 8))
     plt.imshow(img_output)
```

2. 이미지와 마스크 결합 결과

image



output

프로젝트 결과 - 모델

유형별 모델 동시사용과 파손유형별 모델 마스크

유형별 4개 모델 동시 사용 결과 시각화

```
[ ]  labels = ['Breakage', 'Crushed', 'Scratch', 'Seperated']
     models = []

     n_classes = 2
     device = 'cuda' if torch.cuda.is_available() else 'cpu'

     for label in labels:
         model_path = f'/content/drive/MyDrive/aiffel/Te/models/[DAMAGE][{label}]Unet.pt'

         model = Unet(encoder='resnet34', pre_weight='imagenet', num_classes=n_classes).to(device)
         model.model.load_state_dict(torch.load(model_path, map_location=torch.device(device)))
         model.eval()

         models.append(model)

     print('Loaded pretrained models!')
```

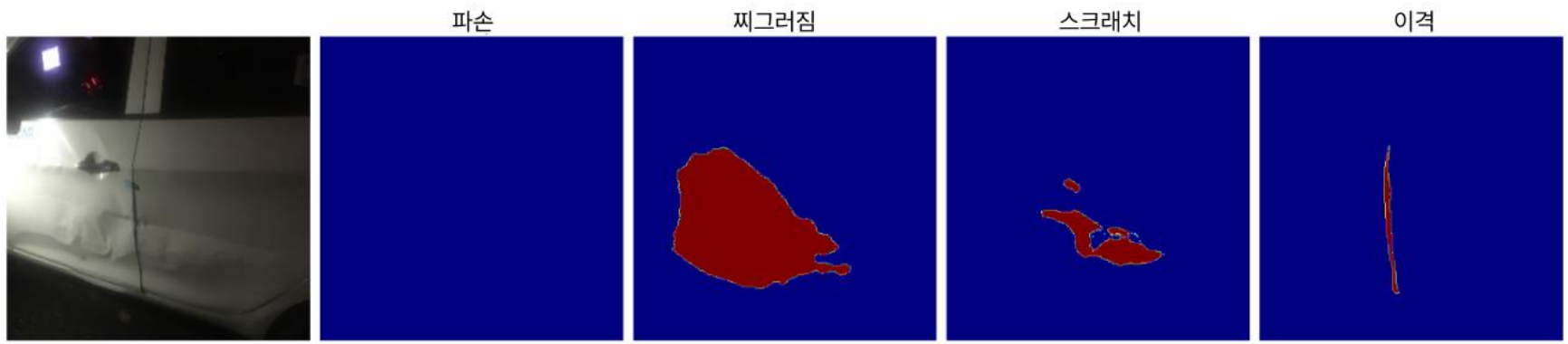
이미지 전처리 파손유형별 모델 마스크 결과 시각화

```
[ ]  img_input = img / 255.
     img_input = img_input.transpose([2, 0, 1])
     img_input = torch.tensor(img_input).float().to(device)
     img_input = img_input.unsqueeze(0)

     fig, ax = plt.subplots(1, 5, figsize=(24, 10))

     ax[0].imshow(img)
     ax[0].axis('off')

     outputs = []

     for i, model in enumerate(models):
         output = model(img_input)

         img_output = torch.argmax(output, dim=1).detach().cpu().numpy()
         img_output = img_output.transpose([1, 2, 0])

         outputs.append(img_output)

         ax[i+1].set_title(labels[i])
         ax[i+1].imshow(img_output, cmap='jet')
         ax[i+1].axis('off')

     fig.set_tight_layout(True)
     plt.show()
```

마스크 유형별 순서

왼쪽부터

파손-찌그러짐-스크래치-이격



프로젝트 결과 - 모델

손상도 알고리즘과 최종 출력 결과

손상도 알고리즘

1. 파손 유형별 면적 크기 계산 후 손상 심각도 계산

```
    img_output = torch.argmax(output, dim=1).detach().cpu().numpy()
    img_output = img_output.transpose([1, 2, 0])

    area_sum = img_output.sum()

[ ] severity = (area_breakage*3.0 + area_crushed*2.0 + area_seperated*1.2 + area_scratch*1.0) * 100 / (3*area_sum)
    severity
```

2. 가중치를 입힌 합계 점수 추론 후
   기준에 따른 손상 심각도 출력

```
[ ] if 0 <= severity < 6:
        grade = 1
    elif  severity < 26:
        grade = 2
    elif  severity < 51:
        grade = 3
    elif  severity < 81:
        grade = 4
    else:
        grade = 5

    print('손상심각도 :', grade, '등급')
```

3. 손상 심각도에 따라 달라지는 면적 계산값 (5등급)

```
[ ] if grade == 1:
      price_table = [
          200, # Breakage_3
          150, # Crushed_2
          50,  # Scratch_0
          120, # Seperated_1
      ]
    if grade == 2:
      price_table = [
          220, # Breakage_3
          170, # Crushed_2
          70,  # Scratch_0
          140, # Seperated_1
      ]
    if grade == 3:
      price_table = [
          240, # Breakage_3
          190, # Crushed_2
          90,  # Scratch_0
          160, # Seperated_1
      ]
```

```
    if grade == 4:
      price_table = [
          260, # Breakage_3
          210, # Crushed_2
          110,  # Scratch_0
          180, # Seperated_1
      ]
    if grade == 5:
      price_table = [
          280, # Breakage_3
          230, # Crushed_2
          130,  # Scratch_0
          200, # Seperated_1
      ]
```

최종 출력 결과

```
total = 0

for i, price in enumerate(price_table):
    area = outputs[i].sum()
    total += area * price

    print(f'{labels[i]}:\t영역: {area}\t가격:{area * price}원')

print('손상심각도 :', grade, '등급')
print(f'총 수리비는 {total}원 입니다.')
```

```
Breakage:       영역: 0   가격:0원
Crushed:        영역: 10508       가격:2416840원
Scratch:        영역: 1845        가격:239850원
Seperated:      영역: 453         가격:90600원
손상심각도 : 5 등급
총 수리비는 2747290원 입니다.
```

프로젝트 결과 - 어플

어플 제작 개요

사용 툴과 초기계획

사용 툴 - **Android studio, PyTorch, Onnx, Tensorflow Lite**

계획 - **Tenserflow Lite 모델 변환 후 Android studio**에서
JAVA 언어를 활용해 모델 탑재 및 모바일 기기 환경 테스트

앱 구현을 위한 시도

1. Android stdio 활용하여 어플 레이아웃 제작

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageButton
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/image"
        />

    <Button
        android:id="@+id/button2"
        android:layout_width="205dp"
        android:layout_height="wrap_content"
        android:text="button2"
        android:layout_below="@id/button1"
        android:layout_alignParentLeft="true"
        />

    <Button
        android:id="@+id/button3"
        android:layout_width="205dp"
        android:layout_height="wrap_content"
```

PyTorch 기반의 모델을 **Tenserflow Lite** 모델 변환 필요

변환 과정

Pytorch > Onnx > Tenserflow > Tenserflow Lite 순으로 변환

1. Pytorch > Onnx

```python
import onnx

# 더미 입력 데이터 생성
dummy_input = torch.randn(1, 3, 256, 256) # 모델 입력 형태

# 모델을 ONNX로 변환
torch.onnx.export(model, dummy_input, "/content/drive/MyDrive/aiffel/Te/converted_models/[DAMAGE][Scratch]Unet.onnx")
```

2. Onnx > Tenserflow

```python
import onnx
from onnx import helper

onnx_model = onnx.load(onnx_model_path)

# Define a mapping from old names to new names
name_map = {"x.1": "x_1"}

# Initialize a list to hold the new inputs
new_inputs = []

# Iterate over the inputs and change their names if needed
for inp in onnx_model.graph.input:
    if inp.name in name_map:
        # Create a new ValueInfoProto with the new name
        new_inp = helper.make_tensor_value_info(name_map[inp.name],
                                                inp.type.tensor_type.elem_type,
                                                [dim.dim_value for dim in inp.type.tensor_type.shape.dim])
        new_inputs.append(new_inp)
    else:
        new_inputs.append(inp)

# Clear the old inputs and add the new ones
onnx_model.graph.ClearField("input")
onnx_model.graph.input.extend(new_inputs)

# Go through all nodes in the model and replace the old input name with the new one
for node in onnx_model.graph.node:
    for i, input_name in enumerate(node.input):
        if input_name in name_map:
            node.input[i] = name_map[input_name]

# Save the renamed ONNX model
onnx.save(onnx_model, '/content/drive/MyDrive/aiffel/Te/converted_models/[DAMAGE][Scratch]Unet_new.onnx')
```

3. 변환 결과

- 여러 파일로 변환하는 과정에서
  버전 간 호환성 이슈 발생

  시도 1) 라이브러리 버전 통일
  시도 2) Onnx > Tenserflow Lite용
  모델 변환 툴 활용

- 결과적으로 **Tenserflow Lite 모델 변환 성공**
  작동 여부는 어플 적용 테스트 필요

프로젝트 결과 - 어플

디자인 개요, 어플 프로토타입

어플 디자인 개요

Figma 활용한 어플 UI, 아이콘, 프로토타입 제작

시그니처 컬러와 아이콘

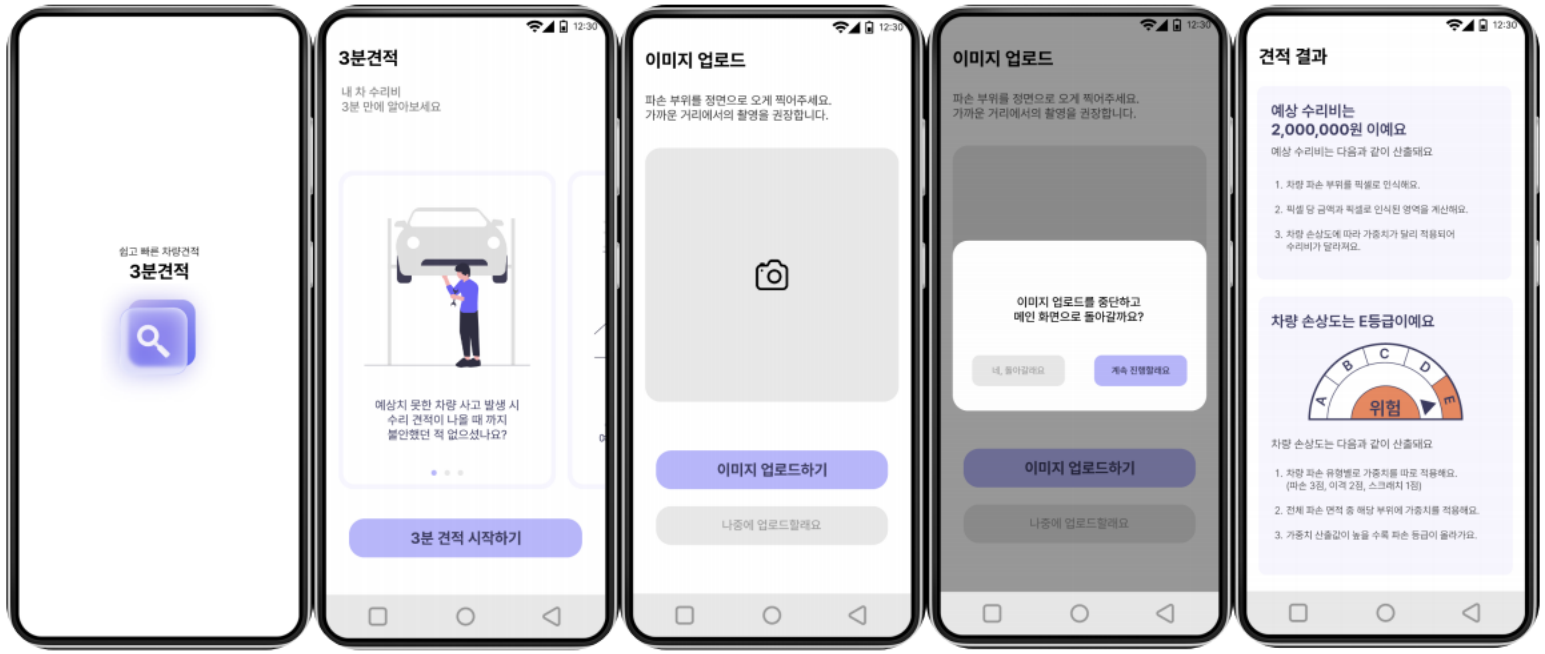
#A788FF

미스터리함과 차량 견적 산출 어려움 연계
S사의 푸른색을 더해 쏘카 연계 프로젝트임을 표현

수리 견적을 제대로 알 수 없는 답답함을
돋보기처럼 꼼꼼히 찾아 견적을 알려줌을 표현

어플 프로토타입

어플 프로토타입 링크



인트로 | 메인 화면 | 업로드 화면 | 업로드 화면 - 동작 취소 | 결과 화면

팀원 소개

팀원 별 역할

김영준

팀장

모델 사용

베이스라인 코드 작성

강동훈

어플 제작

코드 - 어플 결합

밥요정

김민지

베이스라인 코드 작성

자료 조사

레퍼런스 관리

장다연

프로젝트 아이디어 제공

어플 디자인 제작

PPT 제작

팀원 소개

회고

프로젝트 의의

- 자동화 시스템 도입으로 비용과 시간을 효과적으로 절감
- 차량 손상 견적을 빠르게 파악함으로써 고객의 긍정적 경험 유도
- 새로운 툴(Figma)를 이용한 어플 UI 제작
- 어플 제작에 관련된 전반적 경험 체험
- 각 팀원의 역량을 프로젝트에 녹여내는 경험

프로젝트 한계

- 수리 견적 알고리즘의 사전 설정 함수가
  특정 상황에서 부정확한 결과 도출
- 모델과 어플 결합의 어려움
- 파손 유형별 탐지 이후 가격 합산으로
  파손 유형 겹치는 부분은 예상보다 수리비 높게 책정
- 사진 찍는 위치에 따라 파손 부위 크기 변경으로
  촬영 위치 중요
- 수리비 선정 기준 모호 - 시세변동 민감
- 프로젝트 내용 구성 시간보다 툴 적응 시간이 더 많이 소요

질의응답

예상 질문, 질의응답

Q & A

- ‘3분 견적’ 이름의 의미는 무엇인가요?
- 파손 등급별 수리비 선정 기준은 무엇인가요?
- 모델과 어플 결합에서 구체적으로 어떤 어려움이 있었나요?
- 동일한 사진을 다른 각도에서 찍을 때
  다른 결과가 나오는 문제를 어떻게 해결하고자 했나요?

들어주셔서 감사합니다