DONGA-UNIV

TEAM 깜짝할 새

'순간포착; 새상에 이런일이'

이미지 분류 모델 구축을 통한 낙동강 조류 도감 어플리케이션

팀 소개

김미소

프론트엔드 개발

문규성

프론트엔드 개발

신성준

프론트엔드 개발
백엔드 개발

유가은

총괄
프론트엔드 개발
백엔드 개발

목차

작품 개요

개발 배경

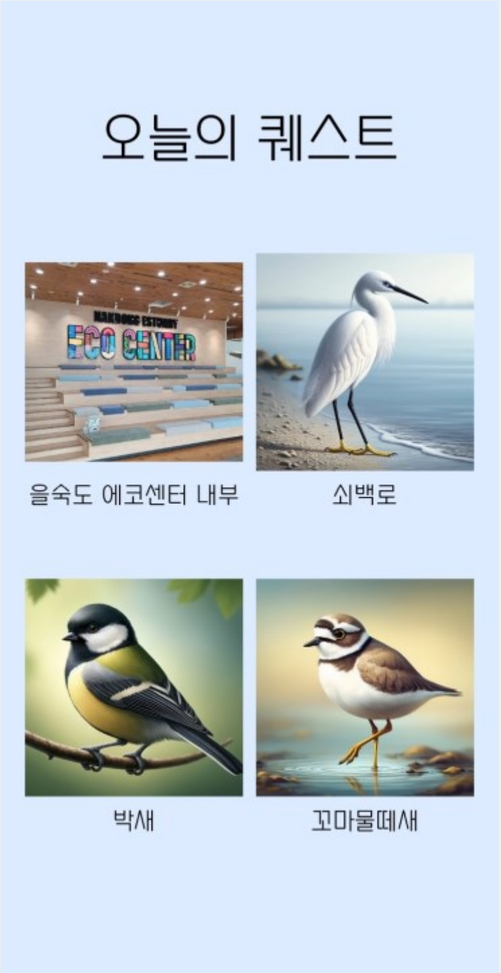
현재 부산시 내 낙동강 생태공원은 서울의 한강 공원과 달리 시민들의 방문도와 관심도가 낮은 편이다. 부산시는 이러한 문제를 해결하고자 생태공원 활성화를 위한 사업을 추진하고 있으며, 동시에 철새 보호에 관한 사업을 추친하고 있다.

이에 우리 팀은 생태공원 활성화와 철새에 대한 시민들의 관심 유도에 전반적인 도움이 되고자 '낙동강 조류 도감 어플리케이션'을 개발하였다.

목표 시스템

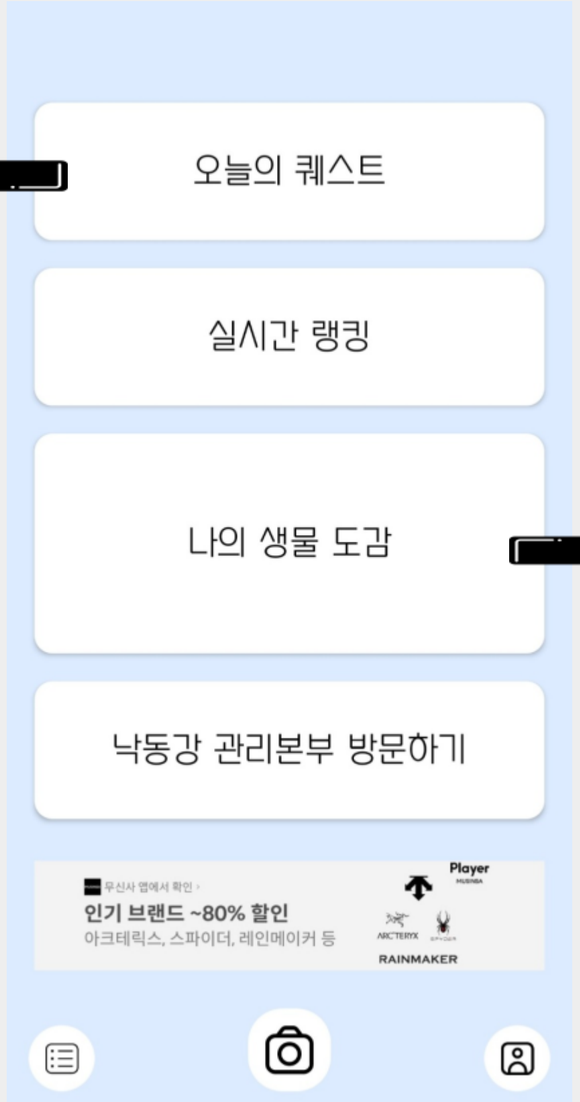
이미지 분류 모델 구축을 통한 낙동강 조류 도감 어플리케이션

작품 구성 및 상세 내용



오늘의 퀘스트 페이지

총 4개의 퀘스트가 매일 랜덤으로 주어지고
퀘스트는 장소 1개, 새 3마리로 이루어져 있다.



메인 페이지

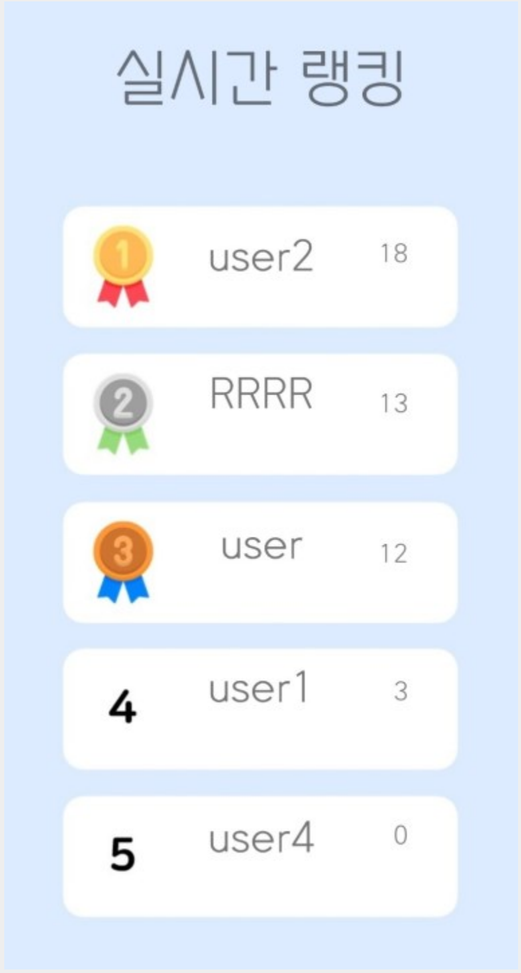
나의 생물 도감 페이지

46종의 새가 실제 새의 모습과 매우 유사한
일러스트로 되어있으며 해당 새의 사진을 찍기
전에는 흑백, 찍은 후에는 유색으로 바뀐다.

작품 구성 및 상세 내용

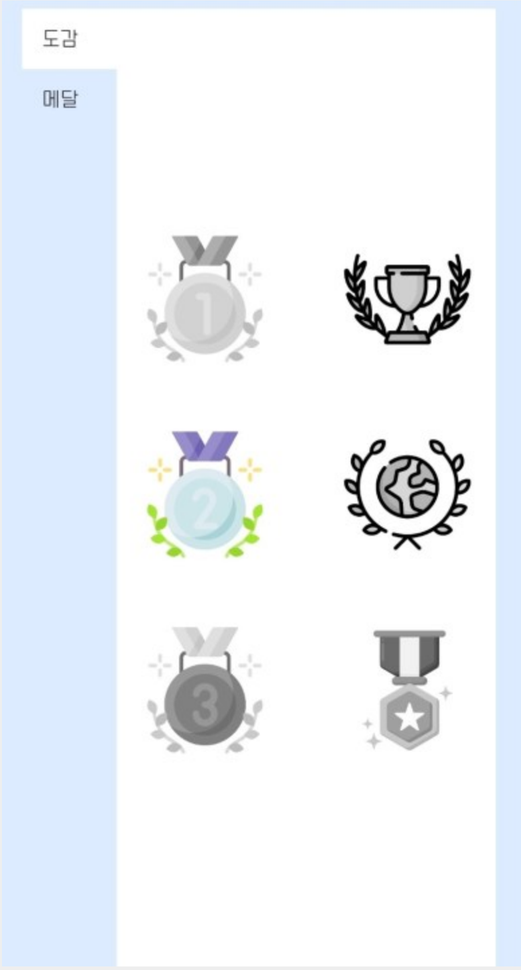
안전 주의 알림창

어플을 처음 열었을 때 나오는
안전 주의 알림창



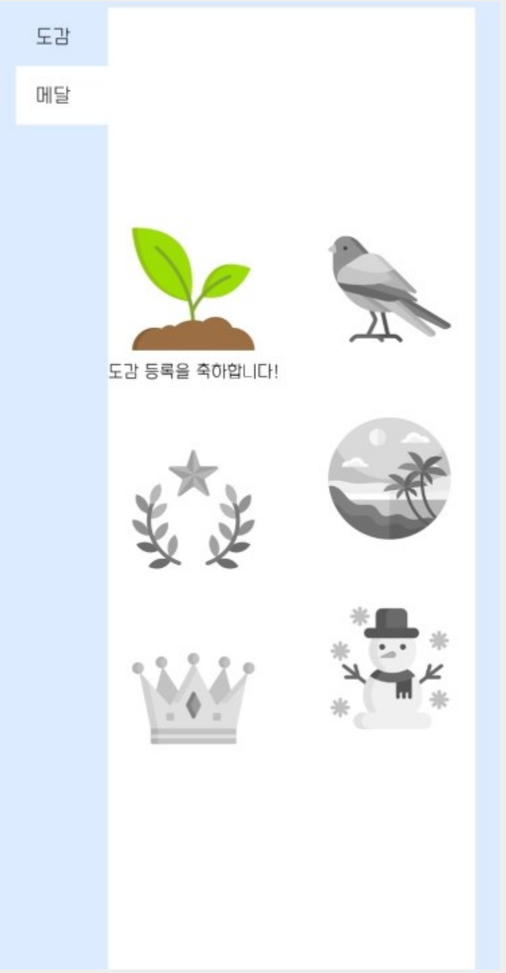
랭킹 페이지

도감 및 퀘스트 완료를 많이 한
기준으로 실시간 랭킹 갱신



메달 뱃지 페이지

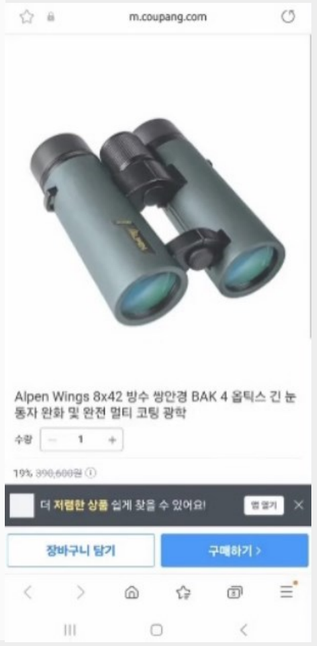
금메달, 은메달, 동메달 획득,
각각 금은동 5개 이상
획득하면 뱃지 수집



도감 뱃지 페이지

첫 등록, 도감 절반, 도감 완성,
천연기념물, 여름 철새, 겨울 철새
도감 등을 달성하면 뱃지 수집

작품 구성 및 상세 내용



낙동강관리본부 페이지

낙동강관리본부 페이지
바로가기 링크 연결

로그인 , 회원가입 페이지

어플은 로그인 후 이용 가능하며
어플 첫 이용시 회원가입 필요

마이 페이지

회원 정보 변경 및 로그아웃 가능,
탐조 리스트와 장비 링크를
구매할 수 있는 페이지

탐조 전 리스트, 장비 링크 페이지

탐조 전 리스트와 탐조 장비를 구매할
수 있는 페이지 연결

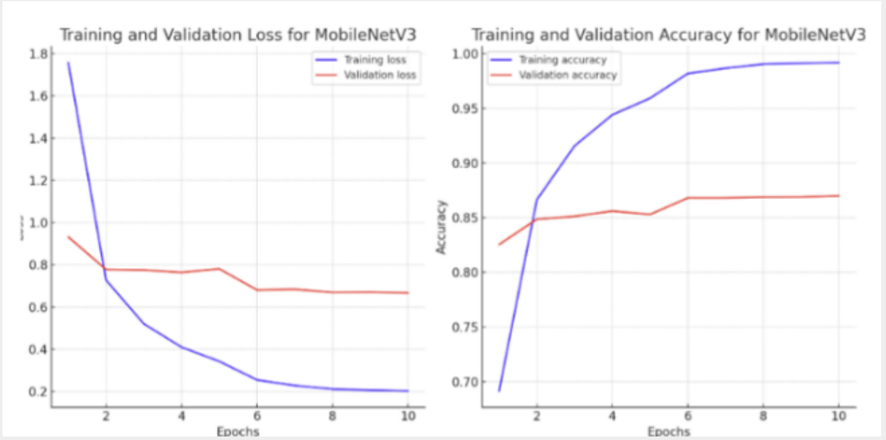
개발 세부내용

```python
# 첫 번째 이미지부터 순서대로 클릭하며 다운로드
#f12 썸네일 이미지 선택 후 class 부분 붙여넣기
#images = driver.find_elements(By.XPATH, "//img[contains(@class,'YQ4gaf')]")
images = driver.find_elements(By.XPATH, "//img[contains(@class,'_fe_image_tab_content_thumbnail_image')]")
count = 1

for img in images:
    if count > 305:
        break
    try:
        img.click()
        time.sleep(3)  # 클릭 후 이미지 로드를 위한 대기
        # 클릭 후 이미지 URL을 얻기
        #f12 큰 이미지 선택 후 class 부분 붙여넣기
        #imgUrl = driver.find_element(By.XPATH, "//img[@class='sFlh5c pT0Scc iPVvYb']").get_attribute('src')
        imgUrl = driver.find_element(By.XPATH, "//img[@class='_fe_image_viewer_image_fallback_target']").get_attribute('src')
        # 이미지 다운로드
        response = requests.get(imgUrl)
        image_name = f"image3_{count}.jpg"
        # 이미지 저장 경로 확인 및 저장
        # 이미지 저장 경로 수동 지정
        save_path = os.path.join(r"C:\Users\user\Desktop\dataset_final\Blacktailedgodwit", image_name)
        with open(save_path, 'wb') as file:
            file.write(response.content)
```

```python
# 데이터 증강을 위한 변환 설정
transform = transforms.Compose([
    transforms.ToTensor(),  # 이미지를 텐서로 변환
    #transforms.RandomHorizontalFlip(p=1),  #수평 뒤집기
    #transforms.ColorJitter(brightness=0.8, contrast=0.2, saturation=0.4),  # 색조, 대비, 채도 조정
    #transforms.Compose([transforms.RandomRotation([90, 90])]), # 90도로 회전
    #transforms.Compose([transforms.RandomRotation([45, 45])]), # 45도로 회전
    #transforms.Compose([transforms.RandomRotation([180, 180])]), # 180도로 회전
    #transforms.Compose([transforms.RandomRotation([270, 270])]), # 270도로 회전
    #transforms.GaussianBlur(kernel_size=(9, 9), sigma=(5.0)),  # 가우시안 블러 적용
```

데이터셋 구축

Python의 Selenium을 사용한 이미지 크롤링 진행 후
이미지 데이터 증강을 통한 92,000장의 이미지
데이터셋(train/validation/test set) 구축



모델 선정

초기 개발 단계에서는 MobileNetV3를 사용해 모델 훈련을
진행하였으나 EfficientNetV2를 사용했을 때
더 좋은 성능을 보였기 때문에 해당 모델을 선정하여 사용

개발 세부내용

```python
from tensorflow.keras.applications import EfficientNetV2S
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense
from tensorflow.keras.regularizers import l2
from tensorflow.keras import Sequential

def build_model(num_classes):
    base_model = EfficientNetV2S(include_top=False, input_shape=(224, 224, 3), weights='imagenet')
    base_model.trainable = True  # 더 세밀한 훈련을 위해 일부 레이어의 훈련 가능 설정

    # 특정 레이어만 훈련 가능하도록 설정
    for layer in base_model.layers[:-30]:  # 마지막 20개 레이어만 훈련
        layer.trainable = False

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dropout(0.5),
        Dense(num_classes, activation='softmax', kernel_regularizer=l2(0.01))
    ])

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

사용한 모델 - EfficientNetV2

새 이미지 분류 모델 및 퀘스트 장소 분류 모델 구축

```python
# 이미지 파일 경로
img_path = r"C:\project\test2.jpg"

# 이미지 로드 및 크기 조정
img = image.load_img(img_path, target_size=(224, 224))

# 이미지를 배열로 변환
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)  # 모델 입력을 위해 차원 확장
# 이미지 예측 (스케일링 생략)
predictions = model.predict(img_array)

# 예측 결과 출력
predicted_class = np.argmax(predictions, axis=1)
predicted_class_name = class_labels[predicted_class[0]]

# 예측 결과의 정확도 출력
confidence = predictions[0][predicted_class[0]]
print("Predicted class:", korean_labels.get(predicted_class_name, "Unknown class"))
print("Confidence:", confidence)
```

✓ 0.2s

```
1/1 [==============================] - 0s 146ms/step
Predicted class: 참새
Confidence: 0.98897785
```

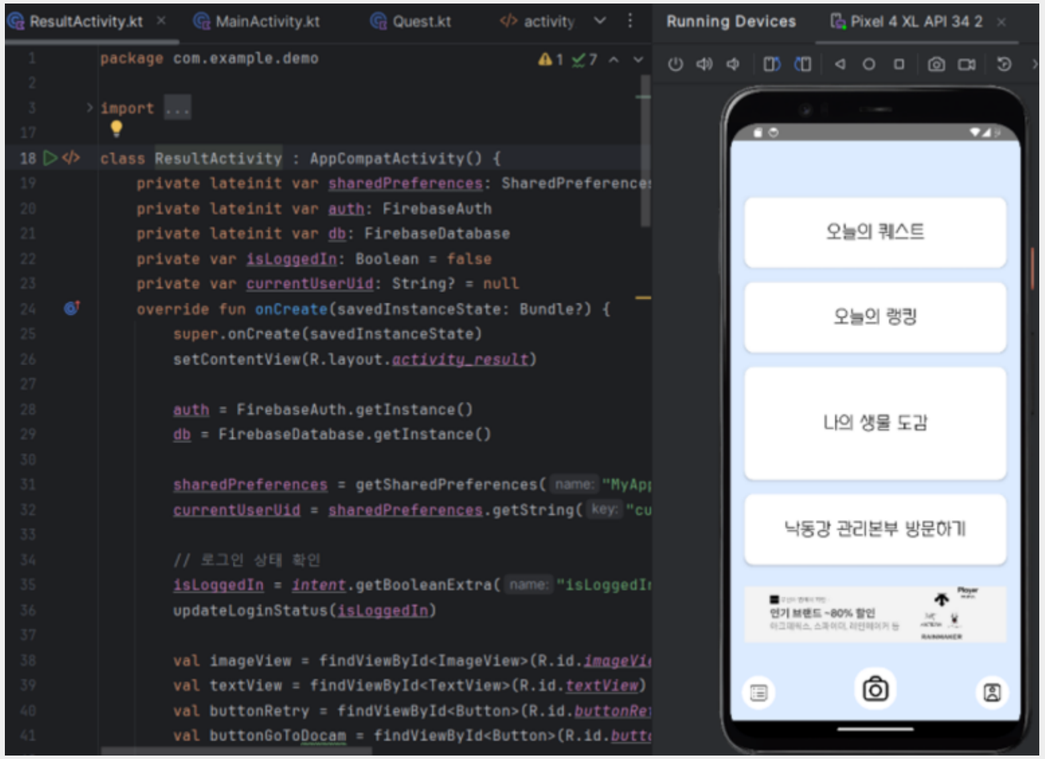
모델 테스트

직접 찍은 사진을 사용해 모델 테스트 진행

개발 세부내용

데이터베이스 - Firebase

Firebase의 Authentication, Realtime Database를 통한
데이터베이스 구축 및 회원 정보 관리

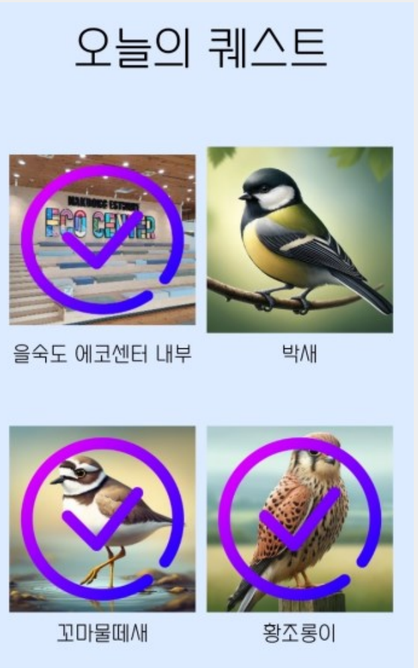


Android 어플 개발 - Kotlin, Android Studio

Kotlin언어를 사용해 어플 개발 진행

개발 세부내용

```
private fun getCurrentMonthFolderName(): String {
    val calendar = Calendar.getInstance()
    val month = calendar.get(Calendar.MONTH) + 1 // 월은 0부터 시작하므로 +1 해줌
    return "$month"
}

private fun getCurrentDate(): String {
    val calendar = Calendar.getInstance()
    return "${calendar.get(Calendar.YEAR)}-${calendar.get(Calendar.MONTH) + 1}
}

private fun showRandomImages() {
    val place = findViewById<ImageView>(R.id.imageView1)
    val placetext = findViewById<TextView>(R.id.textViewImage1)

    val imageViews = listOf(
        findViewById<ImageView>(R.id.imageView2),
        findViewById<ImageView>(R.id.imageView3),
        findViewById<ImageView>(R.id.imageView4)
    )
```



퀘스트 기능 구현

Calendar 클래스를 통해 매일 퀘스트를 새로 업데이트하고,
퀘스트는 해당 시기에 볼 수 있는 새 3종과
생태공원 장소 1곳을 랜덤으로 선정

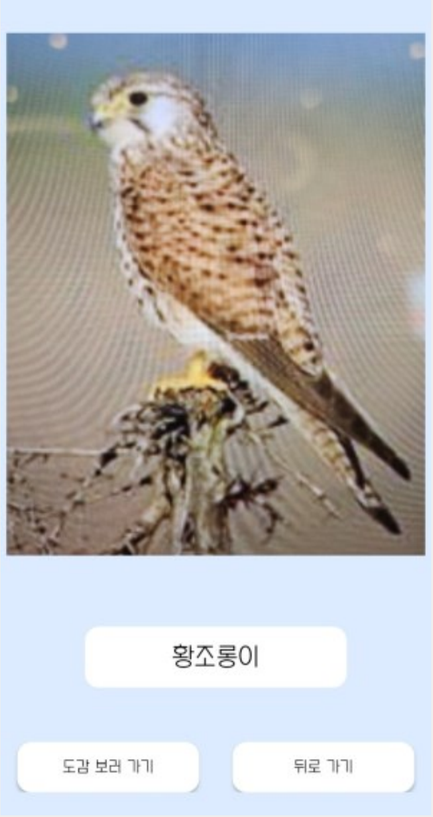
도감 기능 구현

스크롤뷰를 활용해 도감 페이지 구현
도감에 등록되지 않은 새는 흑백으로 처리, 등록된 새 이미지를
클릭하면 새 정보 확인 가능

개발 세부내용

```kotlin
private fun handleImage(image: Bitmap) {
    try {
        val resizedImage = Bitmap.createScaledBitmap(image, dstWidth: 224, dstHei
        val (result, maxConfidence) = classifyImage(resizedImage)

        // 새 모델과 장소 모델 모두 정확도가 80 이하일 때 다시 찍기
        if (maxConfidence <= 0.80) {
            showRetakeDialog()
        } else {
            val stream = ByteArrayOutputStream()
            image.compress(Bitmap.CompressFormat.PNG, quality: 100, stream)
            val byteArray = stream.toByteArray()

            // ResultActivity 시작
            val intent = if (currentModel == birdInterpreter) {
                Intent( packageContext: this, ResultActivity::class.java)
            } else {
                Intent( packageContext: this, LocationResultActivity::class.java)
            }.apply { this: Intent
                putExtra( name: "imageBitmap", byteArray)
                putExtra( name: "resultText", result)
                putExtra( name: "isLoggedIn", isLoggedIn)  // 로그인 상태 전달
            }
            startActivity(intent)
        }
    } catch (e: Exception) {
        Log.e( tag: "MainActivity",  msg: "Error handling image", e)
```

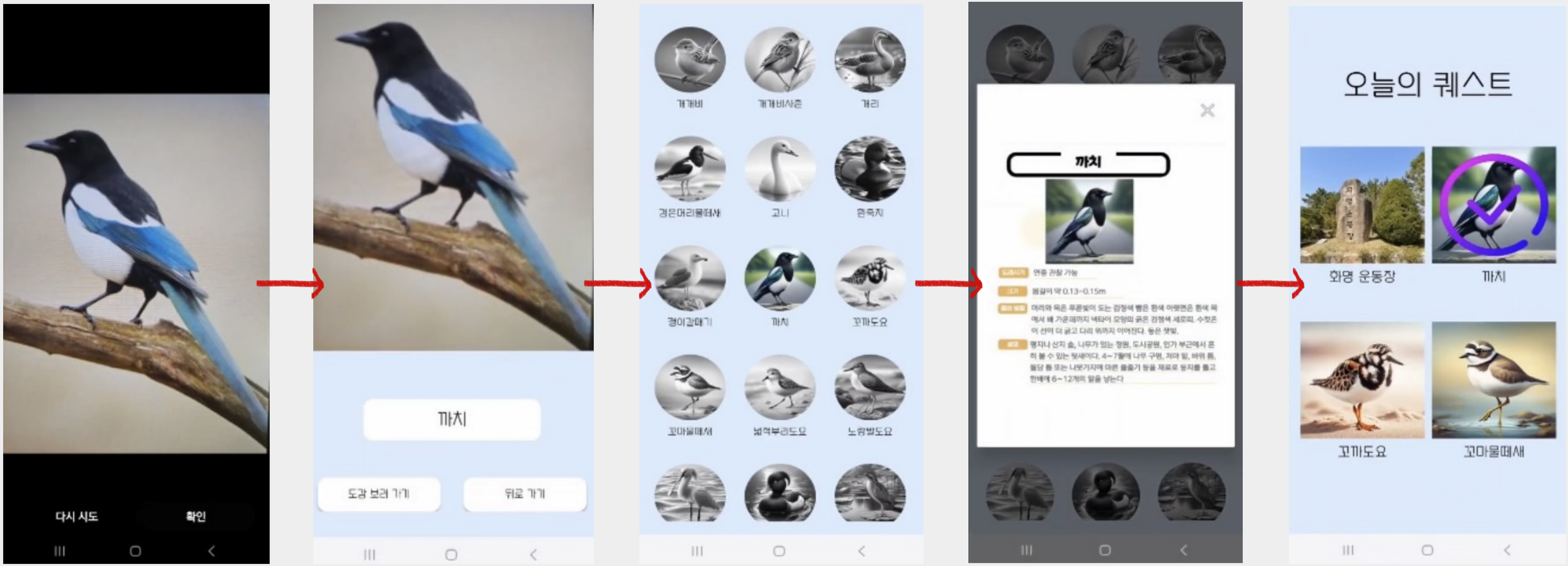
```xml
<TextView
    android:id="@+id/dialogMessage"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:gravity="center"
    android:text="새가 확인되지 않았습니다."

    android:textSize="25sp"
    android:textColor="#000000"
    app:layout_constraintBottom_toTopOf="@+id/positiveButton"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/positiveButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="다시 찍기"
    android:textSize="20sp"
    android:textColor="@android:color/black"
    android:background="@drawable/dialog_background"
    android:paddingLeft="24dp"
```



카메라 기능 구현

카메라를 통해 촬영한 새 및 장소 이미지의 분류 결과 정확도가 80% 이하일 때
다시 촬영하도록 하여 도감 및 퀘스트 진행의 오류 가능성 최소화

구현 결과



카메라를 통해 촬영한 새가 정확히 분류된 경우 도감에 등록된다.

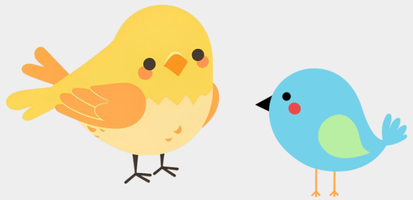
등록된 새는 유색으로 표시되며 등록된 새를 누르면 새에 대한 정보를 볼 수 있다.

오늘의 퀘스트에 나와있는 새라면 오늘의 퀘스트에서도 등록된다.

구현 결과

장소의 경우에도 동일하게 카메라 버튼 클릭 후 장소 버튼을 누르고 사진을 찍는다.
정확히 촬영되고 분류된 장소 사진은 오늘의 퀘스트에 등록된다.

기대효과



'포켓몬고' 앱과 같이 사람들이 도감을 채우는 과정에서
성취감을 느끼며 즐겁게 사용할 수 있도록 게임 형식으로 기획하였다.
또 서울의 한강에 비해 활성화 되지 않은 부산의 낙동강에 대해
‘낙동강 생태공원 활성화’를 목표로 생태계 및 생태 프로그램 정보 제공이 주 목적이다.

이후 철새 뿐만 아니라 다른 생물 도감으로 어플을 수정 혹은 확대할 계획이며,
더 나아가 부산시 어플을 시작으로 전국적으로 사용 가능한 어플 기획을 목표로 하고 있다.

감사합니다.