물고기 관리 앱(my fish)

2019301082 한예은

Ⅰ. 작품 주제 & 선택 이유

- 주제

  물고기의 사진을 찍어 정보를 알아낼 수 있는 정보 앱, 물고기에게 필요한 용품을 구할 수 있는 관리 앱

- 선택 이유

  제가 집에서 물고기를 키우는데 물고기를 키우기 시작한 초반에 어떤 물고기가 공격 받는 것을 보고 어떤 물고기를 같이 키워도 되는지에 대한 합사정보가 중요하다는 것을 깨달았습니다.
  그런데 합사정보는 물고기의 종류를 모르면 알기 어렵고 여러 사이트에 검색해야 알아낼 수 있기 때문에 이런 정보를 한 번에 알 수 있는 어플리케이션이 있으면 좋겠다고 생각해 정했습니다.

Ⅱ. 작품의 가치 + 기존 서비스와의 차이

- 기존 서비스들

어플리케이션

> 물고기 사육 일지를 작성하는 어플은 있지만, 물고기 정보를 한 번에 볼 수 있는 어플은 없음

웹 페이지 (나무위키)

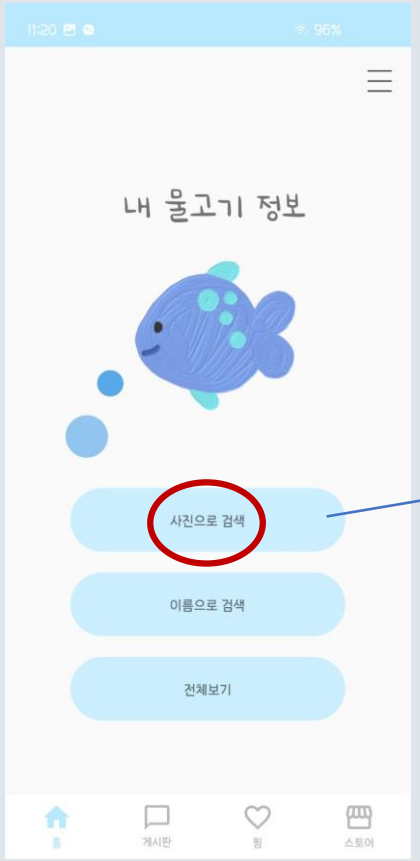
- 열대송사리 - 국내에 서식하는 송사리는 동갈치목이고 열대어들은 열대송사리목이다.
  - 난태생 열대송사리
    - 구피
    - 앤들러스
    - 몰리
    - 플래티
    - 소드테일
  - 난생 열대송사리 - '킬리피쉬'라고도 한다.
    - 램프아이
    - 세이버핀 킬리피쉬
    - 라쵸비
    - 블루쿼이스트 킬리피쉬
    - 레드스트라이프킬리
    - 발렌키아투스카프
    - 판착스
- 잉어류
  - 다니오류
    - 제브라다니오
      - 형광 제브라다니오 - 본래 환경오염 등을 감지하기 위해서 품종개량되었
      - 롱핀 제브라다니오 - 지느러미가 길게 늘어지는 종류. 제브라 다니오 중
      - 펄다니오 - 관상용으로 자주 키우는 다니오. 높이 뛰어서 뚜껑이 필요하다
    - 레오파드다니오 - 제브라와는 달리 이름 그대로 몸의 무늬가 표범처럼 점으로에 가끔씩 섞인 것을 볼 수 있다.
      - 롱핀 레오파드다니오 - 제브라 다니오와 동일.
    - 글로라이트다니오 - 그다지 대중적이지는 못한 종류.
    - 자이언트다니오

> 물고기의 정보를 어느정도 얻을 수 있지만, 합사 정보가 없는 물고기가 많음

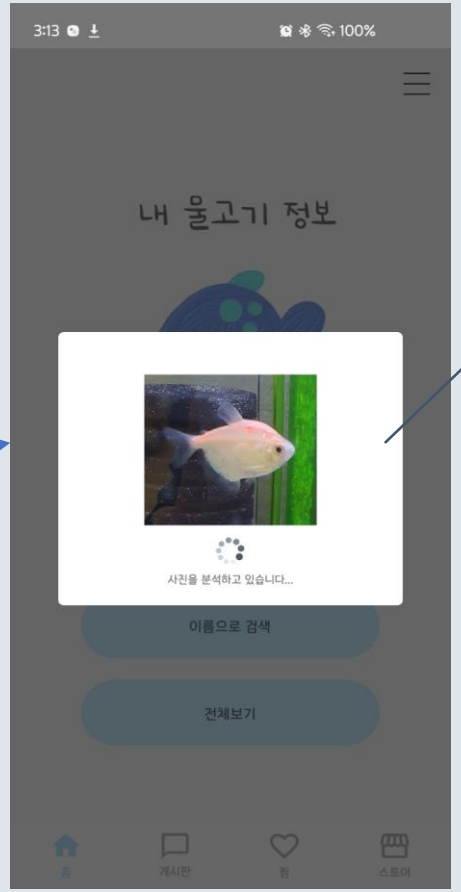
> 물고기의 정확한 이름을 모르면 찾는 데에 어려움이 생김

Ⅱ. 작품의 가치 + 기존 서비스와의 차이

- MyFish 어플리케이션 이용



사진으로 검색

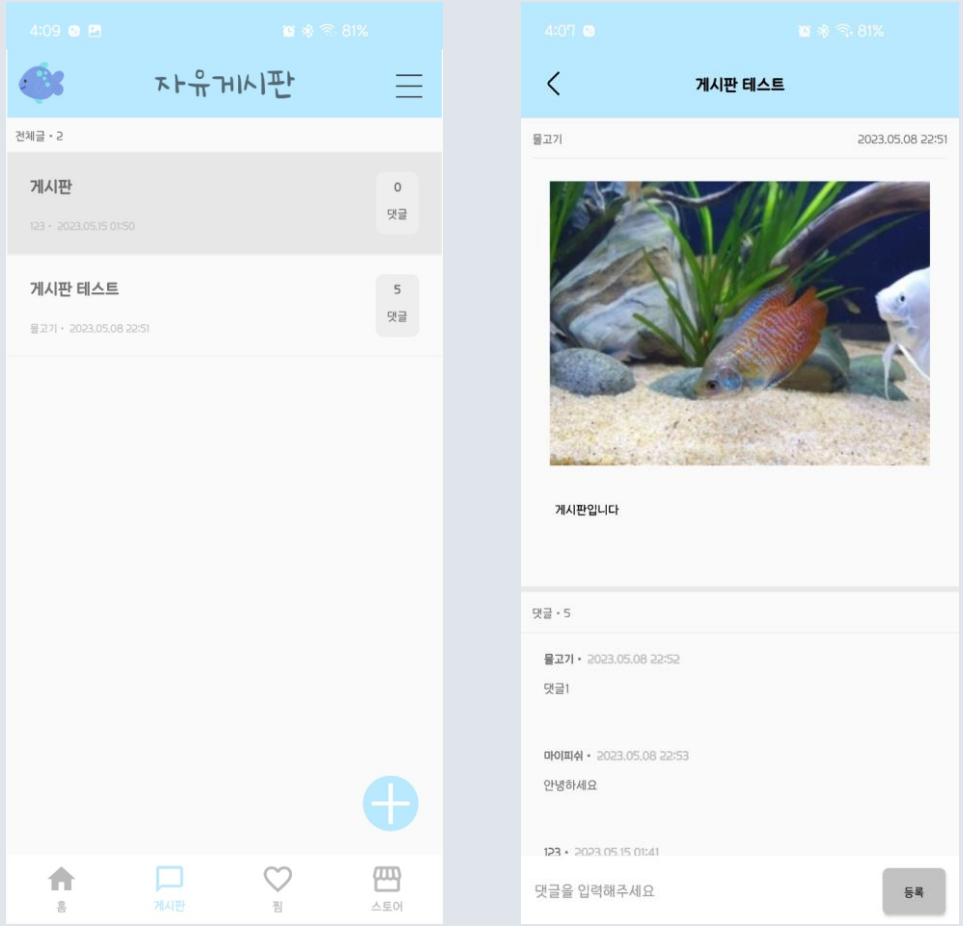


> 물고기의 정확한 이름을 모르더라도 비슷한 물고기를 알 수 있음

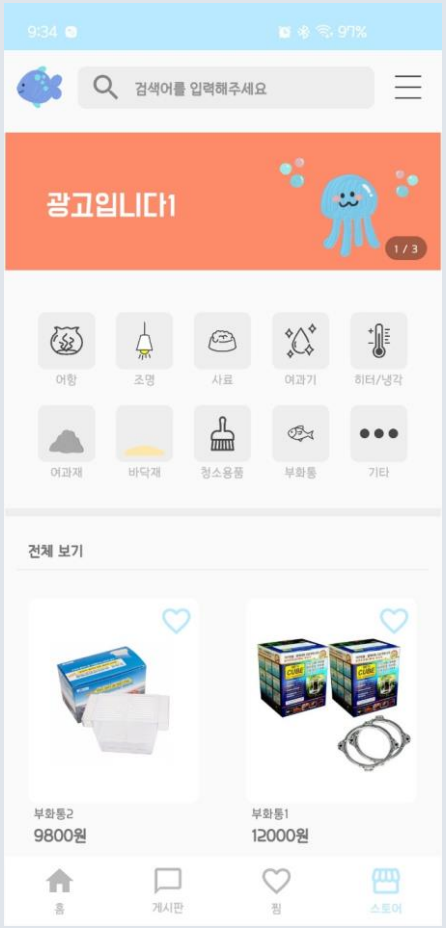
Ⅱ. 작품의 가치 + 기존 서비스와의 차이

- MyFish 어플리케이션 이용 (기타서비스)

게시판 - 정보교환

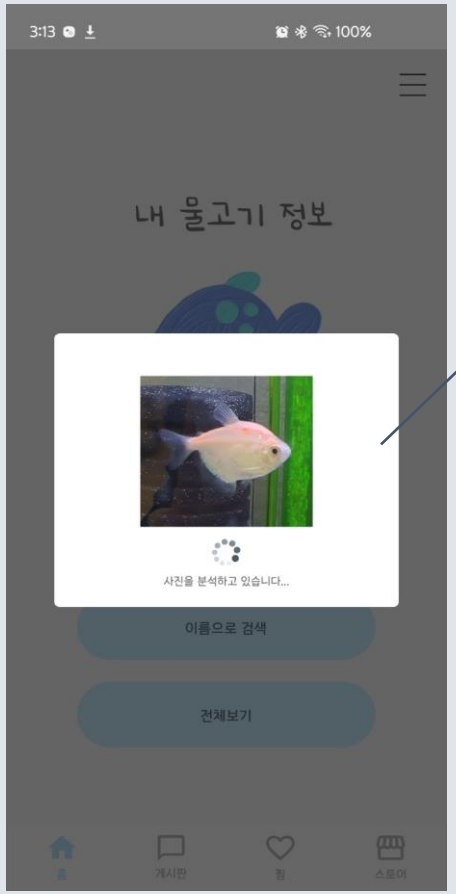


쇼핑몰 – 필요한 용품



Ⅲ. 기술적 측면에서 핵심적인 부분 (사진으로 검색 기능)

사진으로 검색



개발 순서

① 물고기 이미지 크롤링

② 물고기 이미지를 딥러닝으로 학습

③ 물고기를 학습한 모델로 인식하고 결과를 보내는 서버 만들기

④ 안드로이드 측에서 서버에 요청

Ⅲ. 기술적 측면에서 핵심적인 부분 - 크롤링

fishImageDownload.py (물고기 이름으로 이미지 다운로드 하는 함수)

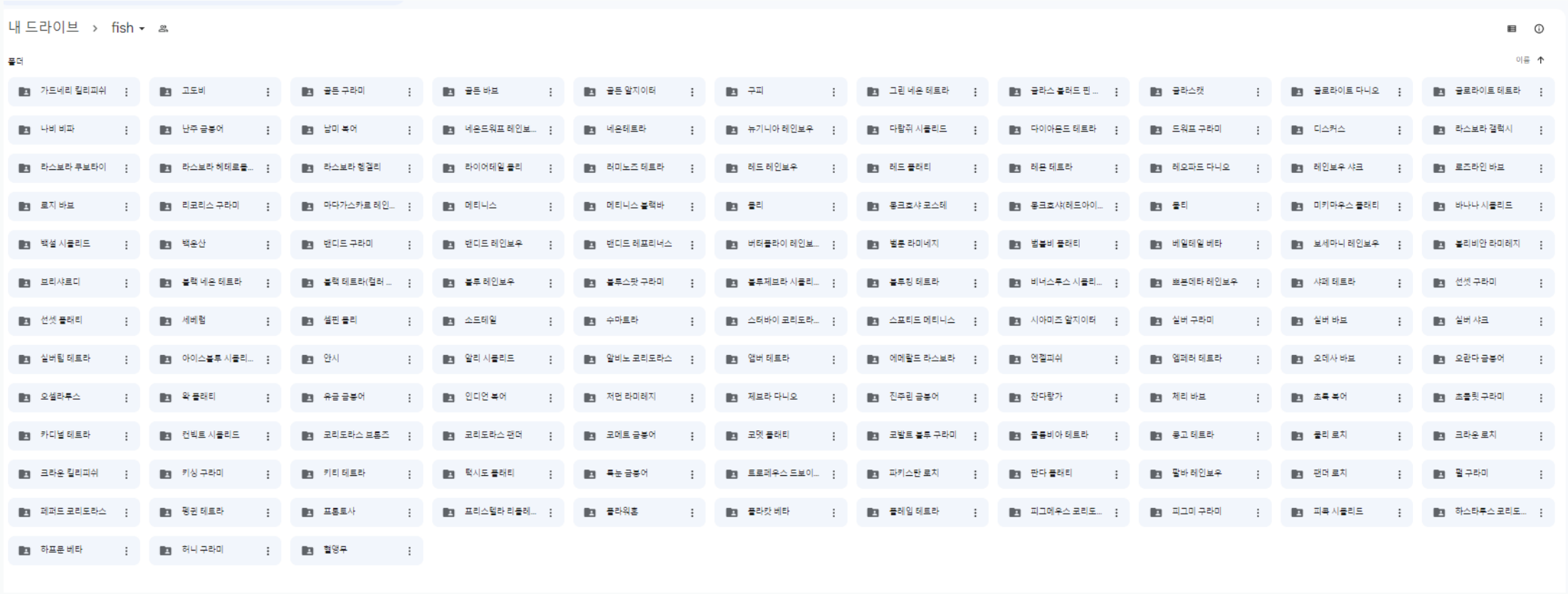
```python
#기본 다운로드
def download(name) :
    #폴더가 없으면 만들기
    if not os.path.isdir(f"{name}/"):
        os.makedirs(f"{name}/")

    #검색어로 이미지 페이지에 접속
    driver.get(f'https://www.google.com/search?q={name}&sxsrf=AJOqlzV-lyw8anLlzFcHKhEUb4ChnzkKBw:1679299540629&source=

    #페이지가 로딩 될 때까지 기다림
    driver.implicitly_wait(10)

    last_height = driver.execute_script("return document.body.scrollHeight") #페이지의 높이를 저장

    cnt = 0 #사진 개수

    #스크롤을 더보기 버튼까지 눌러서 끝까지 내림
    while True:
      driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") #스크롤을 끝까지 내려줌
      time.sleep(3)
      new_height = driver.execute_script("return document.body.scrollHeight") #스크롤을 내린 페이지의 높이를 저장
      if new_height == last_height: #스크롤을 내린 페이지와 이전 페이지의 길이를 비교해 더 이상 내릴 수 없어 같아진다면
        try:
            driver.find_element_by_css_selector('.mye4qd').click() #더보기 버튼 클릭
        except:
            break
      last_height = new_height #새로운 높이를 마지막 높이에 저장

    #이미지 다운
    for val in driver.find_elements_by_css_selector('.rg_i') :
      try :
          if(cnt >= 300) : break #300장 다운받으면 반복문을 빠져나감
          src = val.get_attribute('src')
          urllib.request.install_opener(opener)
          urllib.request.urlretrieve(str(src), f'{name}/{name}_{cnt}.jpg') #이미지 링크로 이미지 저장
          cnt += 1
      except: pass
```

> 검색어로 페이지 접속

> 페이지를 끝까지 내림

> 이미지 요소를 모두 가져와서 300장 다운

Ⅲ. 기술적 측면에서 핵심적인 부분 – 크롤링 결과



> 135종의 물고기 이미지 18953장을 수집했습니다.

Ⅲ. 기술적 측면에서 핵심적인 부분 – 딥러닝

- ResNet 50 모델로 전이학습

fishTrainAndServer.ipynb (딥러닝 학습 코드)

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D
from tensorflow.keras import datasets
from tensorflow.keras.applications.resnet50 import ResNet50

#학습 데이터셋 가져오기
train_datasets = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/fish',
    image_size = (250,250),
    batch_size = 64,
    subset = 'training',
    validation_split=0.2,
    label_mode='categorical',
    seed = 1234
)

#테스트 데이터셋 가져오기
test_datasets = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/drive/MyDrive/fish',
    image_size = (250,250),
    batch_size = 64,
    subset = 'validation',
    validation_split=0.2,
    label_mode='categorical',
    seed = 1234
)

# ResNet50 불러오기
base_model = ResNet50(include_top=False, pooling = 'avg' , input_shape = (250,250 ,3),
```

1. 학습할 이미지 가져오기

2. ResNet 모델 불러오기

```
#학습 못 하게 하기
for i in base_model.layers :
  i.trainable = False


#모델 layer 설계
reshape_layer = tf.keras.layers.Reshape((1, 1, 2048))(base_model.output)
x = tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal')(reshape_layer)
x = tf.keras.layers.experimental.preprocessing.RandomRotation(0.1)(x)
x = tf.keras.layers.experimental.preprocessing.RandomZoom(0.1)(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dropout(0.2)(x)
output = tf.keras.layers.Dense(135, activation='softmax')(x)


model_res = tf.keras.Model(base_model.input, output)


# 모델 컴파일
opt = tf.keras.optimizers.Adam(learning_rate=0.001)

model_res.compile(loss="categorical_crossentropy", optimizer=opt, metrics=['accuracy'])
history = model_res.fit(train_datasets,validation_data=test_datasets, epochs=35) #학습하기
```

3. ResNet 모델 학습 못 하게 하기

4. 모델 설계

5. 컴파일 & 학습

Ⅲ. 기술적 측면에서 핵심적인 부분 – 딥러닝 학습 결과

```
237/237 [==========] - 14s 56ms/step - loss: 0.1045 - accuracy: 0.9852 - val_loss: 0.6773 - val_accuracy: 0.8251
Epoch 15/30
237/237 [==========] - 14s 56ms/step - loss: 0.1023 - accuracy: 0.9829 - val_loss: 0.6858 - val_accuracy: 0.8201
Epoch 16/30
237/237 [==========] - 14s 56ms/step - loss: 0.0902 - accuracy: 0.9872 - val_loss: 0.6826 - val_accuracy: 0.8237
Epoch 17/30
237/237 [==========] - 14s 56ms/step - loss: 0.0862 - accuracy: 0.9857 - val_loss: 0.7000 - val_accuracy: 0.8201
Epoch 18/30
237/237 [==========] - 14s 56ms/step - loss: 0.0759 - accuracy: 0.9891 - val_loss: 0.6901 - val_accuracy: 0.8224
Epoch 19/30
237/237 [==========] - 14s 56ms/step - loss: 0.0721 - accuracy: 0.9896 - val_loss: 0.6899 - val_accuracy: 0.8301
Epoch 20/30
237/237 [==========] - 14s 56ms/step - loss: 0.0665 - accuracy: 0.9902 - val_loss: 0.6775 - val_accuracy: 0.8306
Epoch 21/30
237/237 [==========] - 14s 56ms/step - loss: 0.0667 - accuracy: 0.9877 - val_loss: 0.6965 - val_accuracy: 0.8266
Epoch 22/30
237/237 [==========] - 14s 56ms/step - loss: 0.0600 - accuracy: 0.9903 - val_loss: 0.7130 - val_accuracy: 0.8169
Epoch 23/30
237/237 [==========] - 13s 56ms/step - loss: 0.0600 - accuracy: 0.9892 - val_loss: 0.7008 - val_accuracy: 0.8245
Epoch 24/30
237/237 [==========] - 13s 56ms/step - loss: 0.0571 - accuracy: 0.9896 - val_loss: 0.7307 - val_accuracy: 0.8237
Epoch 25/30
237/237 [==========] - 14s 56ms/step - loss: 0.0543 - accuracy: 0.9906 - val_loss: 0.7174 - val_accuracy: 0.8266
Epoch 26/30
237/237 [==========] - 13s 56ms/step - loss: 0.0526 - accuracy: 0.9896 - val_loss: 0.7381 - val_accuracy: 0.8172
Epoch 27/30
237/237 [==========] - 13s 56ms/step - loss: 0.0482 - accuracy: 0.9913 - val_loss: 0.7293 - val_accuracy: 0.8261
Epoch 28/30
237/237 [==========] - 13s 56ms/step - loss: 0.0442 - accuracy: 0.9919 - val_loss: 0.7286 - val_accuracy: 0.8251
Epoch 29/30
237/237 [==========] - 13s 56ms/step - loss: 0.0447 - accuracy: 0.9917 - val_loss: 0.7776 - val_accuracy: 0.8208
Epoch 30/30
237/237 [==========] - 13s 56ms/step - loss: 0.0436 - accuracy: 0.9916 - val_loss: 0.7667 - val_accuracy: 0.8219
```

> 검증 데이터 정확률 약 82%

Ⅲ. 기술적 측면에서 핵심적인 부분 – API서버 구현

- Flask로 서버 구현

```python
app = Flask(__name__)
model = tf.keras.models.load_model('drive/MyDrive/models/model2')

@app.route('/predict', methods=['POST'])
def predict():
    # 이미지 파일을 업로드 받음
    img = BytesIO(request.files['image'].read())

    # 이미지 전처리
    img = image.load_img(img, target_size=(250, 250))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)

    # 모델 예측
    predictions = model.predict(x)
    predicted_classes = np.argsort(predictions[0])[::-1][:5]  # 사진 예측에서 값이 큰 순서대로 상위 5개 인덱스 가져오기

    print("Top 5 Predicted Labels: ")
    rank = []
    for idx in predicted_classes: #상위 5개 인덱스
        rank.append(class_name[idx])
        print(class_name[idx])
        print(" - ", idx, ": ", predictions[0][idx])


    #예측 결과 반환
    print("결과: ",rank[0], rank[1],rank[2])
    return jsonify({'prediction1': rank[0], 'prediction2' : rank[1], 'prediction3' : rank[2], 'prediction4' : rank[3], 'prediction5' : rank[4]})
```

이미지 파일을 업로드 받음

↓

학습 했던 모델을 가져와 예측

↓

상위 5개 결과를 json형식으로 반환

Ⅲ. 기술적 측면에서 핵심적인 부분 – API서버 구현 & 요청

- Ngrok을 사용해 서버 테스트

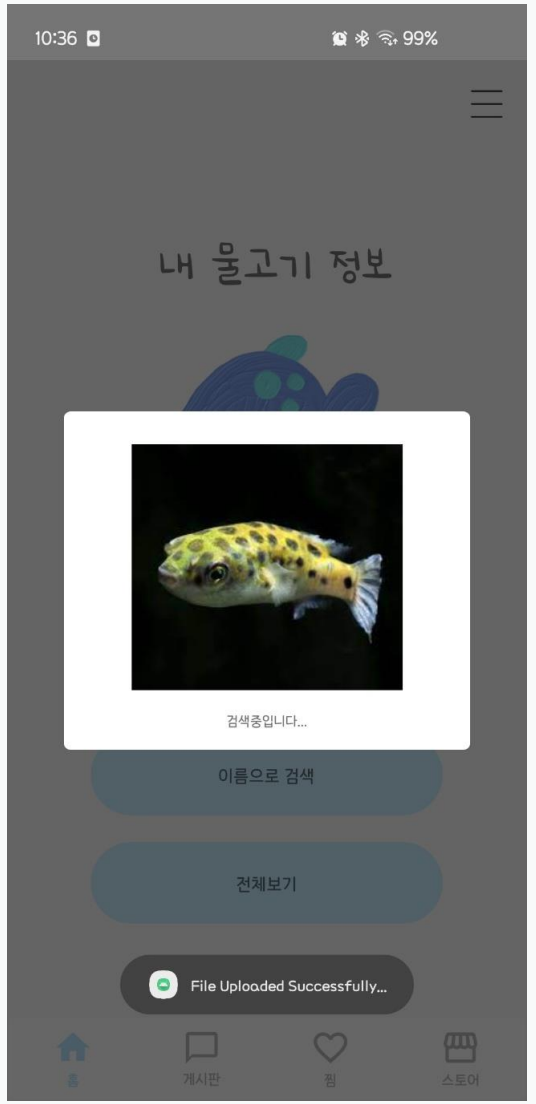
```
run_with_ngrok(app)
app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a produ
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://f575-34-90-213-165.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
1/1 [==============================] - 1s 1s/step
INFO:werkzeug:127.0.0.1 - - [01/May/2023 12:44:05] "POST /predict HTTP/1.1" 200 -
```

- (android) Retrofit으로 서버에 사진 보낸 결과

```
: rank1: 풍선 몰리, rank2: 드워프 구라미, rank3: 블랙 테트라(컬러 테트라), rank4: 초콜릿 구라미, rank5: 블루 스팟 구라미
: rank1: 찬다랑가, rank2: 초콜릿 구라미, rank3: 메티니스 블랙바, rank4: 엔젤피쉬, rank5: 쿨리 로치
: rank1: 초록 복어, rank2: 인디언 복어, rank3: 블루 스팟 구라미, rank4: 플라캇 베타, rank5: 셀핀 몰리
```

서버에 사진 전송



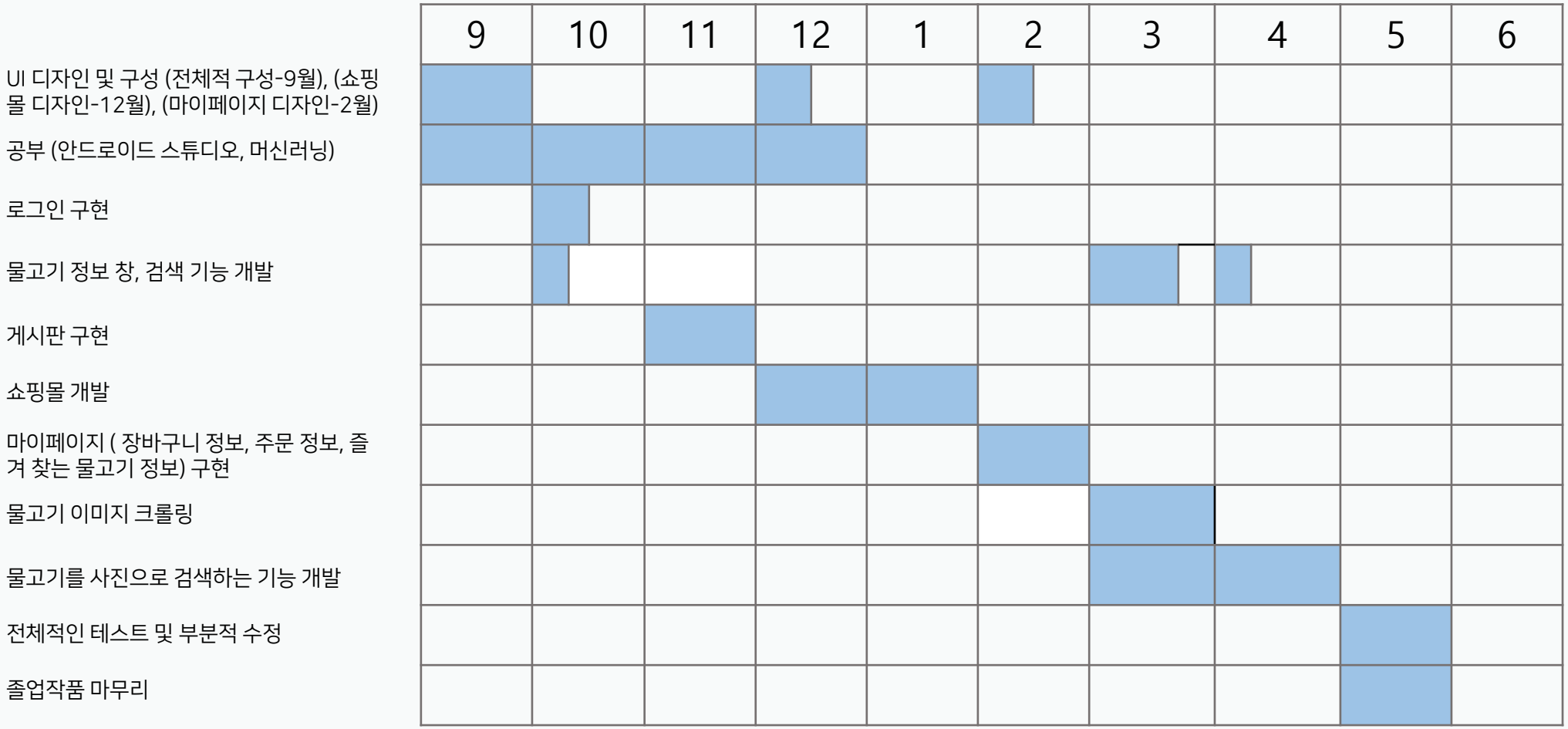
Ⅳ. 작품개발 동안 어려웠던 점 & 해결방법

하나를 꼽아서 어려운 점은 없었고 안드로이드 앱 개발이 처음이라 전반적으로 어려웠던 것 같습니다.
막히는 부분은 구글링을 통해서 알아냈습니다.

Ⅴ. 졸업작품을 마친 소감

제가 직접 필요했던 어플리케이션을 개발할 수 있어서 좋았습니다.

계획 – 전체 계획


9
10
11
12
1
2
3
4
5
6
UI 디자인 및 구성 (전체적 구성-9월), (쇼핑몰 디자인-12월), (마이페이지 디자인-2월)
공부 (안드로이드 스튜디오, 머신러닝)
로그인 구현
물고기 정보 창, 검색 기능 개발
게시판 구현
쇼핑몰 개발
마이페이지 (장바구니 정보, 주문 정보, 즐겨 찾는 물고기 정보) 구현
물고기 이미지 크롤링
물고기를 사진으로 검색하는 기능 개발
전체적인 테스트 및 부분적 수정
졸업작품 마무리

감사합니다