

## 1. 개요

주제: AI 모델을 이용하여 사용자가 입력한 이미지 데이터를 판독해 병해를 입은 상태인지 분석한 결과를 제시하고, 사용자가 등록한 개별 식물을 잘 관리할 수 있도록 수분 주기 등 여러 시각적 정보를 제공하고자 한다.

목표: 사용자가 제공한 식물의 이미지 데이터로부터 식물의 병해 종류를 판별할 수 있는 높은 신뢰도의 AI 모델을 구성하고, 개별 식물의 관리를 위한 정보를 제공할 수 있는 서비스를 Web Application의 형태로 제공한다.

수정사항: 사용자가 제공한 식물의 이미지 데이터로부터 식물의 종류를 판별할 수 있는 AI 모델을 구성하고자 했으나, '식물 집사'로 예상되는 사용자가 자신이 키우는 식물의 종류를 모르지는 않을 것이라는 교수님 Feedback을 바탕으로 AI 모델의 목표를 병해 인식으로 수정하였다.

성과: 프로젝트 <플랜트리>의 성과는 다음과 같다.

### 1) Web Application

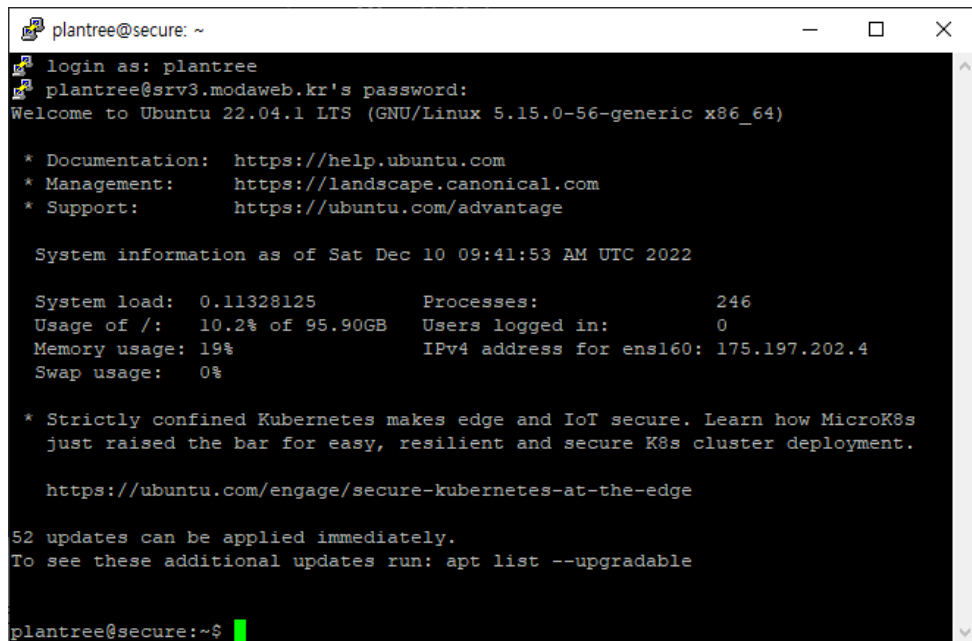
<로그인 페이지 - 회원가입과 로그인 기능을 제공한다.>

<메인 페이지(나의 식물 페이지) - 내가 등록한 식물들의 목록(아이콘, 별명 제시)을 볼 수 있다. 식물 목록의 마지막 아이콘인 '식물 등록'을 클릭하면 <식물 등록 페이지>로 넘어갈 수 있다.>

<식물 등록 페이지 - 내가 등록할 식물의 정보(별명, 식물종, 아이콘 선택)를 입력할 수 있다.>

<개별 식물 페이지 - <식물 목록 페이지>에서 식물 아이콘 클릭 시 해당 식물의 자세한 정보를 볼 수 있는 <개별 식물 페이지>로 넘어올 수 있다. 이 페이지에서는 식물 등록 시 입력한 별명, 식물종 정보를 확인할 수 있고 3번째 구역에서 병해 검사 시 병해 정보를 업데이트할 수도 있다. 2번째 구역에서는 사용자가 식물에 물을 준 주기(최근 5번)를 확인할 수 있고, 당일에 물을 줬다면 반영할 수 있도록 물 주기 아이콘을 활성화한 상태이다.>

## 2) Linux-Based 가상화 서버



```
plantree@secure: ~  
login as: plantree  
plantree@srv3.modaweb.kr's password:  
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-56-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Sat Dec 10 09:41:53 AM UTC 2022  
  
System load:  0.11328125      Processes:           246  
Usage of /:   10.2% of 95.90GB Users logged in:      0  
Memory usage: 19%           IPv4 address for ens160: 175.197.202.4  
Swap usage:   0%  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
just raised the bar for easy, resilient and secure K8s cluster deployment.  
  
https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
52 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
plantree@secure:~$
```

Linux-Based 가상화 서버를 별도로 확보해 프로젝트 진행에 사용하였다.

plantree@srv3.modaweb.kr 주소로 접근할 수 있는 서버 자원을 확보해 서비스에 제공하였으며, 서버 환경은 Ubuntu 22.04.1 LTS, 100GB Storage이다.

Web Application과 연계해 병해 Classification 결과를 안내하며, 전체 서비스가 원활하게 동작할 수 있도록 한다.

## 3) 병해 Classification 모델

MobileNetV2를 기반으로 한 병해 Classification 모델로, 주어진 데이터셋에서 97%의 정확도를 보인다.

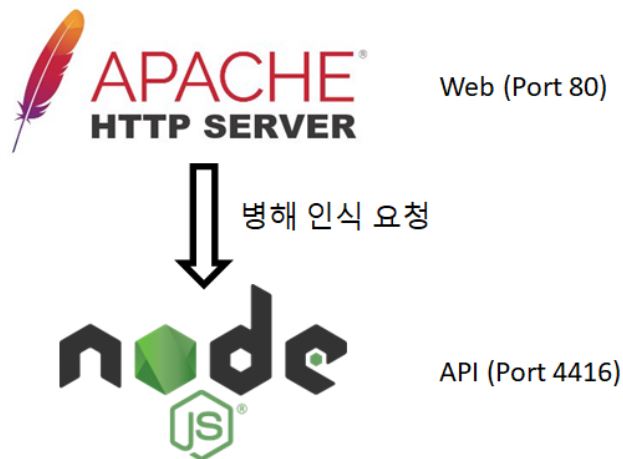
병해 Classification 모델을 이용해 진행한 병해 Prediction 결과의 예시는 다음과 같다.

```
1/1 [=====] - 0s 404ms/step  
ACTUAL CLASS: AppleCedarRust1.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.999993  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: AppleCedarRust2.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.953868  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: AppleCedarRust3.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.945962  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: AppleCedarRust4.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.970986  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: AppleScab1.JPG, PREDICTED: class: Apple_scab, confidence: 0.961121  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: AppleScab2.JPG, PREDICTED: class: Apple_scab, confidence: 0.999966  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: AppleScab3.JPG, PREDICTED: class: Apple_scab, confidence: 0.999278  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: CornCommonRust1.JPG, PREDICTED: class: Corn_Common_rust_, confidence: 0.999979  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: CornCommonRust2.JPG, PREDICTED: class: Corn_Common_rust_, confidence: 0.999983  
1/1 [=====] - 0s 22ms/step  
ACTUAL CLASS: CornCommonRust3.JPG, PREDICTED: class: Corn_Common_rust_, confidence: 0.999997
```

#### 4) Back-End 서버와 Prediction API

Node.js를 통해 서비스의 Background가 되어줄 Back-End 서버를 구성하였고, Prediction API를 구현해 사용자로부터의 병해 인식 요청에 따라 병해 인식 결과를 제공할 수 있도록 하였다.

```
$("#input#photo").change(function() {  
    var formData = new FormData($('#photo_form')[0]);  
  
    $('#overlay').addClass('active');  
  
    $.ajax({  
        url: 'http://srv3.modaweb.kr:4416/',  
        contentType : false,  
        processData : false,  
        type: 'POST',  
        data: formData,  
        success: function(data) {
```



```
const express = require('express');  
const cors = require('cors');  
const app = express();  
const SERVER_PORT = 4416;  
  
app.listen(SERVER_PORT, () => {  
    console.log(`Server Listening at port ${SERVER_PORT}`);  
})  
  
app.use(cors({  
    origin: "http://srv3.modaweb.kr",  
    credentials: true,  
    optionsSuccessStatus: 200  
}));
```

NodeJS로 동작하는 API 서버

CORS를 허용해야 함

자세한 내용은 <4. 성과>의 <산출물> 부분에 명시하였으니, 이를 참고해주시기 바랍니다.

## 2. 추진체계

팀구성: 다음의 형태로 팀을 구성하였다.

이름	역할
조형준(팀장)	전체 프로젝트 총괄, 백엔드 총괄
정다경	프론트엔드 총괄, UI/UX 디자인 구성
YANG YUNLONG	UI/UX 디자인 구성, 위키 관리
이택민	회의록 작성

프로젝트 관리: 모든 팀원이 원활하게 소통할 수 있도록 Discord와 같은 메신저를 이용하였으며, 전체 프로젝트의 형상을 공유하면서 버전을 관리할 수 있도록 GitHub Repository를 이용하였다.

The screenshot shows the GitHub interface for the repository 'lekonell / capstone\_plantree'. The repository is private and has 1 branch and 0 tags. The commit history shows a series of commits by YtoL0803, including adding files via upload, removing files, and updating the README.md. The README.md file content is displayed below the commit history, showing the project title 'capstone\_plantree' and links to the mobile UX design and UI design interfaces.

lekonell / capstone\_plantree Private

<> Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Add file <> Code

YtoL0803 Add files via upload 584a719 2 hours ago 48 commits

images	remove	2 days ago
models	add extract-leaf	7 days ago
plan	Add files via upload	2 hours ago
web	remove some images	2 days ago
보고서	add some report	2 days ago
회의록	Add files via upload	2 days ago
README.md	Update README.md	3 hours ago

README.md

### capstone\_plantree

모바일 UX (임시) <https://ovenapp.io/view/RP5dZCubqA81aa4TcQr9vvFzlytrmtw#eKA9F>

<figma - ui 디자인> 메인화면1 <https://www.figma.com/file/yGlyJ0LAO6SJNjLo0XSHpZ/1?node-id=0%3A1&t=z3JKpSYcRZpisEoV-1>

<UI 디자인 인터페이스> <https://www.figma.com/proto/VNiRowGp8a4bHqpUgxyo0v/Botany?node-id=7%3A85&scaling=min-zoom&page-id=0%3A1>

개인별 기여사항: 다음은 프로젝트 <플랜트리>의 개인별 기여사항이다.

<조형준>: 전체 프로젝트 총괄 및 백엔드 총괄

- 프로젝트 <플랜트리>에서의 서비스를 제공할 Linux-Based 가상화 서버 확보
- 병해 Classification 모델 개발 및 학습
- 병해 Prediction을 위한 이미지 전처리 프로세스 개발
- Back-End Nodejs 서버 개발 및 서비스 연동
- Web Application 개발 참여 (HTML/CSS, Javascript, PHP)
- Database Modeling 및 서비스 연동
- GitHub Repository를 이용한 프로젝트 현황 공유 및 형상 관리
- 프로젝트 발표자료 및 보고서 작성

<정다경>: 프론트엔드 총괄, UI/UX 디자인 구성

- Figma 등의 디자인 도구를 이용한 UI/UX 및 디자인 설계
- Web Application 개발 참여 (HTML/CSS)
- Database Modeling 및 서비스 연동
- GitHub Repository를 이용한 프로젝트 현황 공유 및 형상 관리
- 프로젝트 발표자료 및 보고서 작성

<YANG YUNLONG>: UI/UX 디자인 구성

- Figma 등의 디자인 도구를 이용한 UI/UX 및 디자인 설계
- Web Application 개발 참여 (HTML/CSS)
- 캡스톤디자인I 11팀 "양정이조" 위키 편집

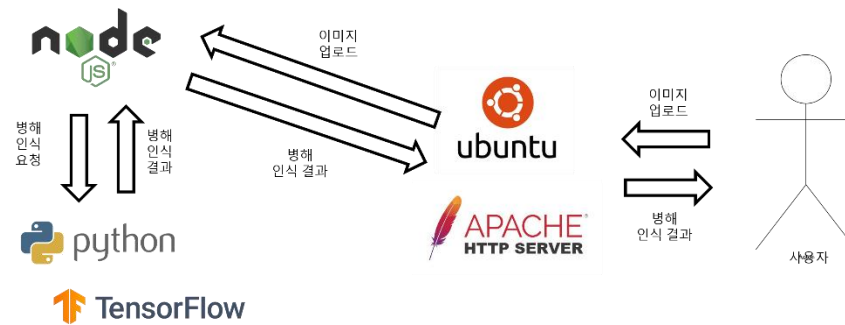
<이택민>

- 회의록 작성

### 3. 개발내용

개발범위: 사용자가 직접 마주할 UI/UX Front-End부, 병해 Classification 및 Prediction 파트, Prediction 이전 개별 유향 전처리 파트, 전체 서비스를 유기적으로 연결할 Back-End 서버가 프로젝트 <플랜트리>의 개발범위에 해당한다.

시스템 또는 SW 구조: 프로젝트 <플랜트리>는 다음의 구조도로 구성되어 있다.



프로젝트 <플랜트리>의 개발도구는 다음과 같다.



개발내용: 병해 Classification에 관련된 부분이 프로젝트 <플랜트리>에서의 주요 구성 요소이며, 해당 내용을 개발 및 구현하기 위해 다음의 여건을 고려하였다.

- 1) 데이터셋은 충분한 양이 제공되어야 하며, Classification이 되어있어야 한다.
- 2) 해당 데이터셋을 기반으로 높은 신뢰도를 가진 병해 인식 모델을 개발해야 한다.
- 3) 병해 인식은 Python 환경에서 수행되며, 사용자는 Web Application을 통해 Request를 전송하므로, Web Application으로 응답을 보낼 수 있는 적절한 IPC(Inter-Process Communication) 기법을 적용하여야 한다.
- 4) 병해를 높은 신뢰도로 인식하기 위해서는 사용자로부터 전송받은 이미지가 인식하기 좋은 형태여야 하고, 이에 따라 적절한 전처리 프로세스를 고안 및 개발해야 한다. 따라서 이미지의 Resize 및 Segmentation, Filtering 등의 과정을 고안해 병해 인식 신뢰도를 높여야 한다.

이하의 내용은 <4. 성과>의 <산출물> 부분에 명시하였으니, 이를 참고해주시기 바랍니다.

Issue 및 해결방안: 프로젝트 <플랜트리>에서의 Issue 및 해결방안은 다음과 같다.

프로젝트 목표: 프로젝트 제안발표에서의 교수님 Feedback을 바탕으로, '식물 집사'로 기대되는 사용자가 자신이 키우는 식물의 종류를 모르지는 않을 것으로 예상되므로, 기존의 프로젝트 목표였던 '식물 종류 인식'을 '식물의 병해 유무 판별 및 병해 정보 제공'으로 변경한다.

비용 문제: GPU 등 HW 자원을 이용한 AI모델 학습에는 비용이 발생할 수 있는데, 해당 문제는 Google Colaboratory 등 GPU 자원을 무료로 사용할 수 있도록 제공하는 서비스를 이용해 해결할 수 있다.

데이터셋 문제: 프로젝트 <플랜트리>에서 확보한 데이터셋에 문제가 있을 가능성이 있다.

- Classification: 아직 분류가 되지 않은 데이터셋일 경우 해당 데이터셋에 Labeling을 진행하는 데에 많은 시간이 소요되는데, 이미 Labeling이 되어있는 데이터셋을 확보해 이를 해결할 수 있다.

- Bias: 확보한 데이터셋이 특정 Class에 치우쳐진, 편향된 자료일 수 있는데, 적절한 Train/Test 데이터 분리를 통해 대응할 수 있다.

- Limitation: 많은 데이터를 수집한다고 하더라도 모든 식물종과 각 식물의 병해에 대해 Cover할 수는 없는데, 현실적으로 모든 식물종/병해에 대한 인식을 수행할 수는 없기에 인식이 가능한 대상에 대한 정확도를 높이는 방향으로 대응하고자 한다.

인식 문제: 사용자가 제공한 이미지가 식물종 및 병해 인식이 어려운 대상일 가능성이 있는데, Prediction 이전의 충분한 전처리(정규화) 과정을 통해 인식률을 높이는 방향으로 대응하고자 한다.

- 이미지 인식률을 높이기 위한 방법으로 두 가지 전처리 과정을 개발하였다. (1) HSV Filtering, (2) Leaf Segmentation이 이에 해당하는데, (2) Leaf Segmentation의 개발을 시도해 개별 잎에 대한 분리가 가능하도록 하고자 했으나 구현한 개별 잎 Segmentation 함수의 정확도가 낮아 활용하기 어려운 것으로 판단해, (1) HSV Filtering만 적용하도록 하였다. HSV Filtering은 전체 이미지에서 잎의 색감에 해당하는 부분만 Masking을 통해 살려두고, 나머지는 Noise인 '배경'으로 취급해, 배경에 해당하는 부분을 검정색으로 마킹하는 방식을 말한다. 자세한 내용은 <4. 성과>의 <산출물> 부분에 명시하였으니, 이를 참고해주시기 바랍니다.

호환성 문제: Mobile Application의 경우 사용할 API에 따라 사용자의 환경에서 호환되지 않을 가능성이 존재하는데, 가능한 범용적으로 사용되는 API를 이용해 많은 시스템 버전을 Cover할 수 있도록 하고자 한다.

- 이 Issue는 프로젝트 초기 계획이었던 Mobile Application 개발에서 Web Application 개발로 프로젝트 진행 방향을 변경하면서 해소되었습니다. (더 이상 고려할 필요가 없음)



#### 4. 성과

목표치: 사용자가 프로젝트 <플랜트리>의 서비스를 이용하면서 자신이 기르고 있는 식물을 등록 및 관리할 수 있고, 개별 식물의 수분 주기를 관리할 수 있어야 하며, 식물의 병해 검사를 요구할 경우 서비스에서 병해 검사 결과를 높은 신뢰도로 제공할 수 있어야 한다.

수행결과: 현재 프로젝트 <플랜트리>에서는 Web Application을 개발 및 서비스하고 있으며, 병해 Classification을 수행하는 Prediction Model의 개발 및 학습을 완료하였고, 프로젝트에서 사용할 Database와 Web Application을 연동해 서비스를 제공하고 있으며, Web Application과 Back-End 서버간의 통신을 이용해 병해 Prediction 결과를 제공하고 있다.

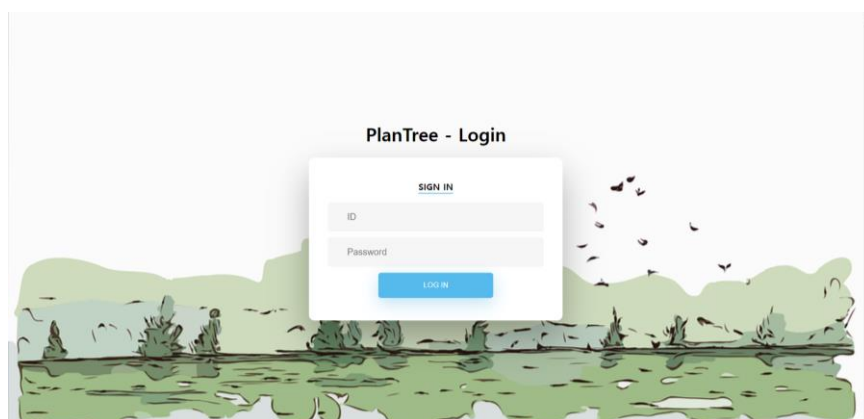
목표 대비 95%의 성과를 보였으며, 미흡한 부분은 다음과 같다.

병해 인식 정확도 문제: 일반적으로 사용자가 식물의 잎을 촬영해 병해를 검사하고자 할 때, (1) 잎을 댄 뒤 (2) 이를 배경 등 Noise가 적은 환경에 배치해 촬영해 검사하기보다는 (1') 잎은 떼지지 않은 상태이고, (2') 배경 등의 Noise가 있을 가능성이 현저히 높은 환경의 이미지로부터 병해 검사를 요구할 것이다. 이에 따라, 검사하고자 하는 잎을 개별적으로 추출할 수 있는 Segmentation 과정을 구현해야 하는데, 해당 Segmentation 부분의 개발을 시도했으나 Segmentation 전처리 함수를 통해 도출되는 개별 잎 분리 결과에 대한 정확도가 현저히 낮아 활용이 불가능할 것으로 보고, 배경의 Noise만을 제거하기 위한 HSV Filtering을 전처리 과정에 사용하였다. 즉, 프로젝트 <플랜트리>에서의 병해 검사 신뢰도를 높이기 위한 전처리 과정이 미흡했다고 평가할 수 있다.

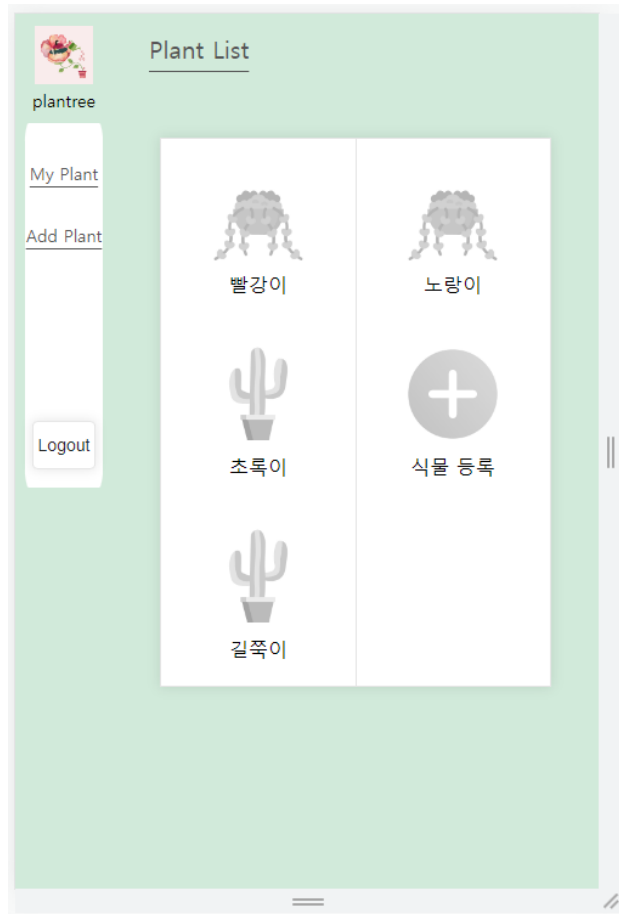
산출물: 프로젝트 <플랜트리>의 산출물은 다음과 같다.

##### 1) Web Application

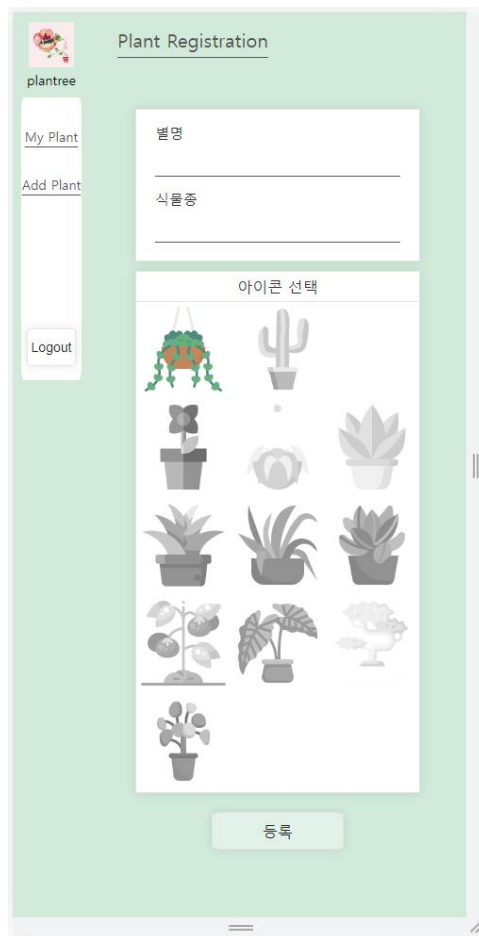
<로그인 페이지>: 회원가입 및 로그인 기능을 제공한다.



<메인 페이지>: 자신이 등록한 식물 목록을 확인할 수 있다. 개별 식물을 선택할 경우 <식물 정보 페이지>로 이동할 수 있고, '식물 등록' 아이콘을 선택할 경우 식물을 등록할 수 있는 <식물 등록 페이지>로 이동할 수 있다.

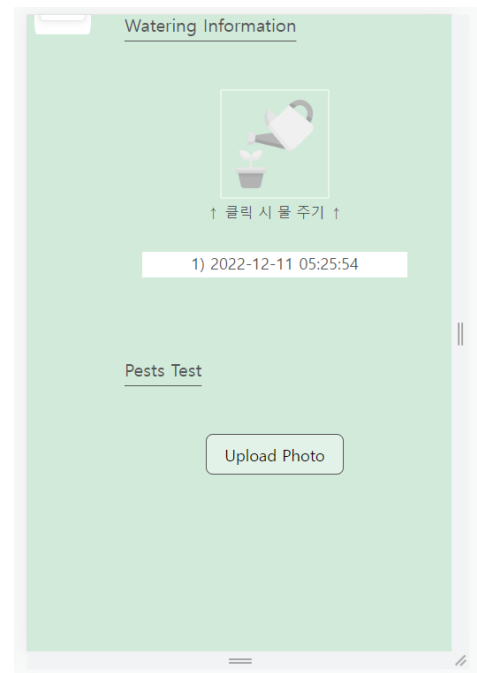


<식물 등록 페이지>: 별명과 식물 종류, 아이콘을 선택해 간편하게 자신의 식물을 등록할 수 있다.

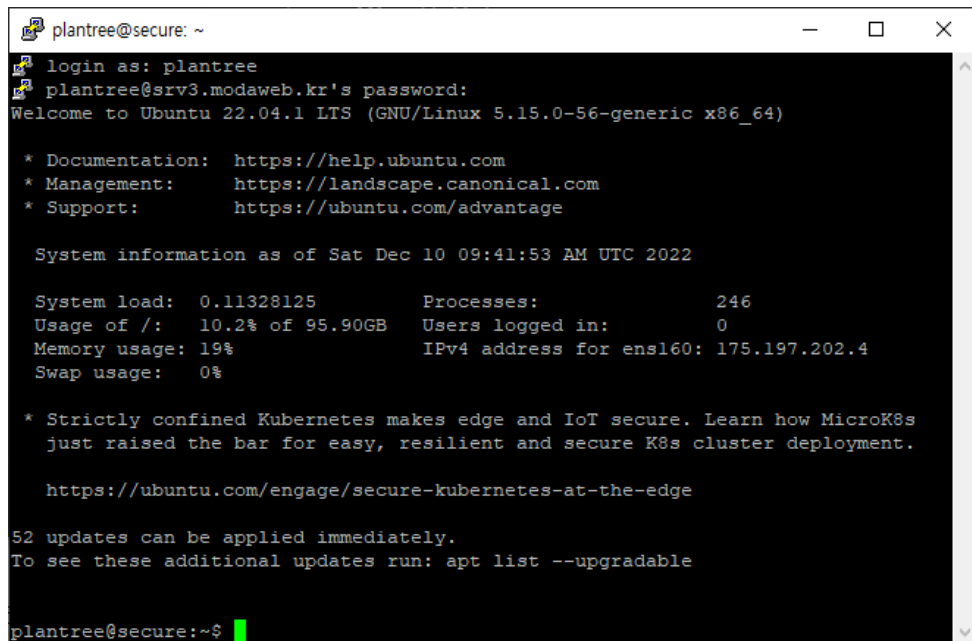


The image shows a mobile application interface for plant registration. The app is titled "plantree" and has a green background. On the left side, there is a vertical menu with the following options: "plantree" (with a small plant icon), "My Plant", "Add Plant", and "Logout". The main content area is titled "Plant Registration". It contains two input fields: "별명" (Nickname) and "식물종" (Plant Type). Below these fields is a section titled "아이콘 선택" (Icon Selection) which displays a grid of 12 different plant icons. At the bottom of the screen, there is a green button labeled "등록" (Register).

<식물 정보 페이지>: 자신이 등록한 식물의 정보를 확인하고, 수분 주기를 관리하거나 병해 검사를 요청할 수 있다.



## 2) Linux-Based 가상화 서버



```
plantree@secure: ~  
login as: plantree  
plantree@srv3.modaweb.kr's password:  
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-56-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Sat Dec 10 09:41:53 AM UTC 2022  
  
System load:  0.11328125      Processes:           246  
Usage of /:   10.2% of 95.90GB Users logged in:       0  
Memory usage: 19%           IPv4 address for ens160: 175.197.202.4  
Swap usage:   0%  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s  
  just raised the bar for easy, resilient and secure K8s cluster deployment.  
  
  https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
52 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
plantree@secure:~$
```

프로젝트 <플랜트리> 서비스의 기반이 되는 서버로, plantree@srv3.modaweb.kr로 연결할 수 있다. Ubuntu 22.04.1 LTS, 100GB Storage 환경으로 구성되어 있다.

## 3) 병해 Classification 모델

병해 Classification 모델을 구현하기 위해 (1) 멘토링 이전의 시도(First Try), (2) 멘토링 이후의 시도(Based on MobileNetV2)가 있었다. 멘토링 과정에서 프로젝트 <플랜트리>에서의 예고가 되어 줄 병해 Classification 모델에 대한 조언을 받았고, 이에 따라 기존의 모델에서 개선된 새 버전의 병해 Classification 모델을 구성할 수 있었다. First Try에서의 Model Accuracy는 약 84%이고, Based on MobileNetV2에서의 Model Accuracy는 약 97%로 13%p의 정확도 개선이 있었다. (Error Rate 대비 -16%에서 -3% 수준으로의 급격한 신뢰도 상승이 있었다.)

```

model.add(Conv2D(32, (3, 3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))

model.summary()

opt = Adam(learning_rate=LR, decay=(LR / EPOCHS))
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

history = model.fit(augment.flow(x=x_train, y=y_train, batch_size=BATCH_SIZE),
                    validation_data=(x_test, y_test),
                    steps_per_epoch=(len(x_train) // BATCH_SIZE),
                    epochs=EPOCHS,
                    verbose=1)

```

위는 First Try에서의 병해 Classification 모델 학습 방식이다.

아래는 Based on MobileNetV2에서의 병해 Classification 모델 학습 방식이다.

```

mobilenetv2 = keras.applications.mobilenet_v2.MobileNetV2(
    include_top=False,
    input_shape=(224, 224, 3)
)
mobilenetv2.trainable = False

input_layer = keras.Input(shape=(224, 224, 3))
x = mobilenetv2(input_layer, training=False)

```

멘토링 조언에 따라 병해 Classification 모델의 구현 방식을 변경하면서 얻은 성과는 <5. 검증>의 <검증결과>, <분석>부분에 명시하였으니, 이를 참고해주시기 바랍니다.

#### 4) Node.js 기반 Back-End 서버

```
const express = require('express');
const cors = require('cors');
const app = express();
const SERVER_PORT = 4416;

app.listen(SERVER_PORT, () => {
  console.log(`Server Listening at port ${SERVER_PORT}`);
})

app.use(cors({
  origin: "http://srv3.modaweb.kr",
  credentials: true,
  optionsSuccessStatus: 200
}));
```

사용자로부터의 병해 인식 요청을 수용하고 병해 인식 결과를 전달할 수 있도록 Node.js를 기반으로 한 Back-End API 서버를 구성하였다. 웹서비스는 80포트에서 Apache에 의해 제공되고, API 서버는 4416포트에서 Node.js에 의해 제공된다. 사용자가 Web에서 AJAX를 이용한 병해 인식 요청을 시도하면, Back-End API 서버는 child\_process를 이용해 Python-based Predict Process를 진행하고, 해당 결과를 전달받아 사용자 Request에 응답한다.

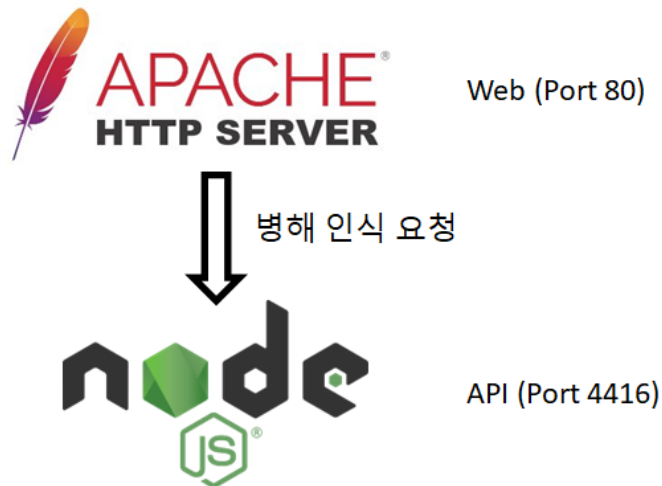
```

$("input#photo").change(function() {
    var formData = new FormData($('#photo_form')[0]);

    $('.overlay').addClass('active');

    $.ajax({
        url: 'http://srv3.modaweb.kr:4416/',
        contentType : false,
        processData : false,
        type: 'POST',
        data: formData,
        success: function(data) {

```



사용자의 병해 인식 요청이 Node.js에 도달하면, 이미지 파일을 적절한 위치에 업로드한 뒤 Predict Process를 진행한다.

```

const multer = require('multer');
const uploader = multer({ dest: __dirname + '/predict_images/' });

```

Multer를 이용한 이미지 파일 업로드

```

app.post('/', uploader.single('photo'), (req, res) => {
    var img_path = req.file.path;
    var resSended = false;

    const spawn = require('child_process').spawn;
    const result = spawn('python3', ['predict.py', img_path]);

```

child\_process를 이용한 Prediction 요청



## 5) 병해 Prediction을 위한 이미지 전처리 Process

병해를 높은 신뢰도로 Predict하기 위해서는 사용자로부터 전송받은 이미지에 적절한 전처리(정규화) 과정이 선행되어야 한다. 이에 따라, 다음의 이미지 전처리 Process를 개발하였다.

### (1) 이미지 Resize

```
def load_image(filename):  
    img = cv2.imread(filename)  
    img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]))  
    img = img / 255  
  
    return img
```



이미지 크기를 Resize한다.

### (2) HSV Filtering

```
def hsv_filter(img):  
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)  
  
    mask_green = cv2.inRange(hsv, (36,0,0), (86,255,255))  
    mask_brown = cv2.inRange(hsv, (8, 60, 20), (30, 255, 200))  
    mask_yellow = cv2.inRange(hsv, (21, 39, 64), (40, 255, 255))  
  
    mask = cv2.bitwise_or(mask_green, mask_brown)  
    mask = cv2.bitwise_or(mask, mask_yellow)  
  
    ret = cv2.bitwise_and(img, img, mask=mask)  
  
    return ret
```



앞에 해당하는 부분만 가져와 Noise를 최소화한다.

HSV Filtering은 앞에 해당하는 부분의 이미지만 Mask해 나머지는 Noise(배경)으로 간주하고 배경을 검정색으로 마킹해 병해 Prediction의 정확도를 높이는 데에 기여한다.

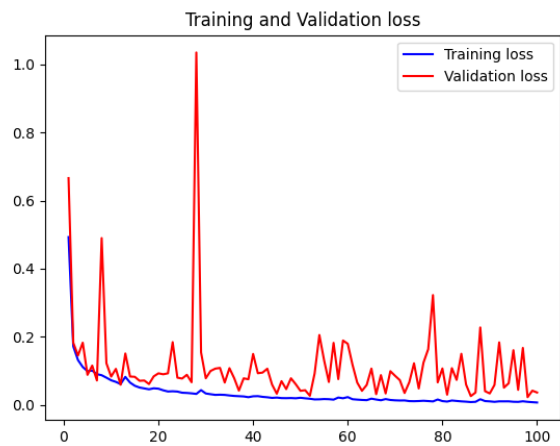
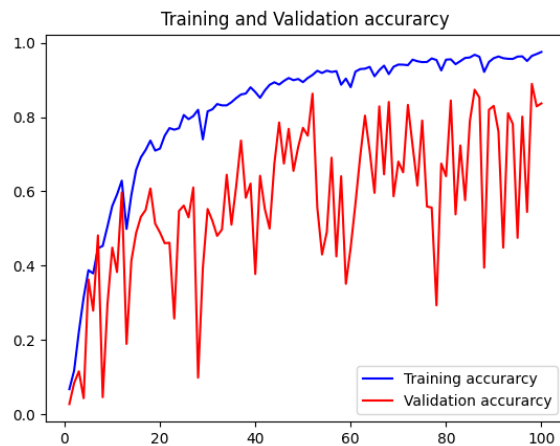
멘토 평가의견: (\* 멘토로부터 프로젝트 평가의견을 전달받고자 두 차례 메일을 전송하였으나,  
별다른 답변을 받지 못해 멘토 평가의견을 남기지 못했습니다. 평가의견 전송  
요청 메일의 이력을 교수님께 별도로 전송했습니다.)

## 5. 검증

검증결과: 프로젝트 <플랜트리>의 품질 검증은 다음과 같이 진행하였다.

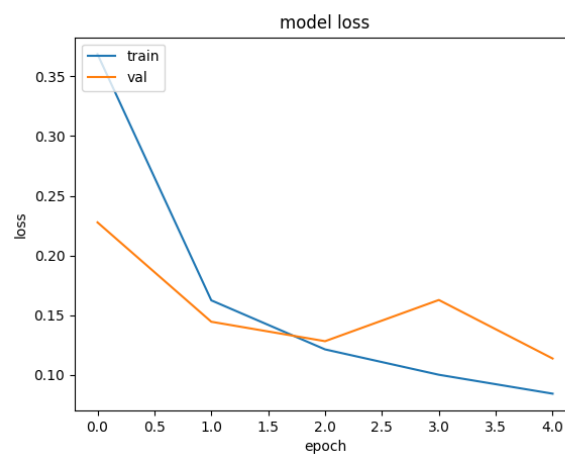
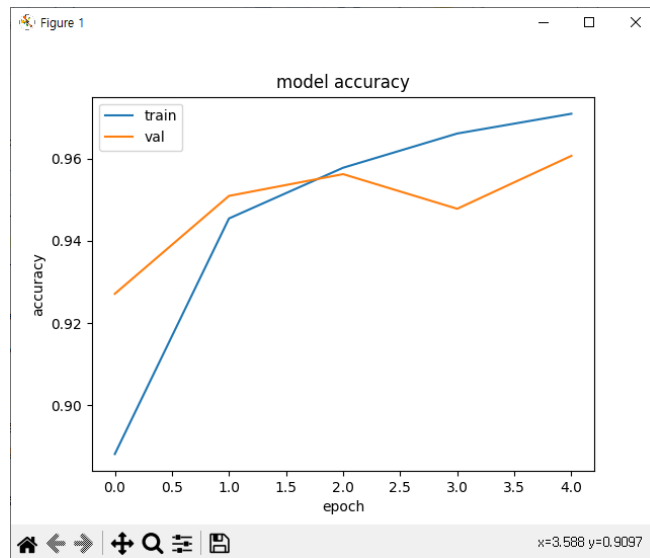
병해 Classification 모델에서 학습 Epoch별 Accuracy, Loss 곡선을 그리고 Model Accuracy, Model Loss를 바탕으로 모델 품질을 산정한다. Model Accuracy가 높을수록, Model Loss가 낮을수록 좋은 모델이다. Model Accuracy는 병해 인식에서 가장 중요한 부분으로, 높은 정확도가 보장되어야 한다.

병해 Classification 모델 (First Try)



해당 Model에서의 Prediction Accuracy는 약 84%이다.

## 병해 Classification 모델 (Based on MobileNetV2)



해당 Model에서의 Prediction Accuracy는 약 97%이다.

분석: 각 학습 Epoch마다 Model Accuracy는 증가하는 방향으로, Model Loss는 감소하는 방향으로 그래프가 그려져야 안정적인 학습이 진행되고 있는 것이다.

<병해 Classification 모델 (First Try)>에서의 병해 Prediction Accuracy는 약 84%였고, 학습 Epoch에 따라 Validation 편차가 커 안정적인 학습이 진행되고 있는 것으로 판단하기 어렵다.

<병해 Classification 모델 (Based on MobileNetV2)>에서의 병해 Prediction Accuracy는 약 97%였고, 이전의 모델(First Try)에 비해 Model Accuracy는 증가하고 Model Loss는 감소한, 더 안정적으로 학습된 모델의 그래프를 보여주고 있다.

개선된 병해 Classification 모델이 기존 모델에 비해 더 좋은 모델임을 확인할 수 있다.

개선된 병해 Classification 모델에서의 병해 Prediction 결과 예시는 다음과 같다.

```
1/1 [=====] - 0s 404ms/step
ACTUAL CLASS: AppleCedarRust1.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.999993
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: AppleCedarRust2.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.953868
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: AppleCedarRust3.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.945962
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: AppleCedarRust4.JPG, PREDICTED: class: Cedar_apple_rust, confidence: 0.970986
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: AppleScab1.JPG, PREDICTED: class: Apple_scab, confidence: 0.961121
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: AppleScab2.JPG, PREDICTED: class: Apple_scab, confidence: 0.999966
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: AppleScab3.JPG, PREDICTED: class: Apple_scab, confidence: 0.999278
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: CornCommonRust1.JPG, PREDICTED: class: Corn_Common_rust_, confidence: 0.999979
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: CornCommonRust2.JPG, PREDICTED: class: Corn_Common_rust_, confidence: 0.999983
1/1 [=====] - 0s 22ms/step
ACTUAL CLASS: CornCommonRust3.JPG, PREDICTED: class: Corn_Common_rust_, confidence: 0.999997
```

병해 추측 정확도(Prediction Confidence)가 90% 이상인 경우는 높은 신뢰도를 보이고 있음을 확인할 수 있다.

First Try 모델에서의 멘토링 평가 의견으로, '현재 공유해주신 Curve와 비교했을 때 현재 모델이 아주 우수한 성능을 보이고 있지는 않은 것으로 보인다.', '학습 Epoch에 따라 Validation 편차가 매우 크게 발생하고 있어 안정적인 학습이 진행중인 것으로 판단하기 어렵다.', '계산량이 적은 Network를 사용하고 있어 발생하는 문제는 아닌지?', 'Scratch 학습중인 경우에는 ImageNet 등의 적절한 Pre-trained Model을 활용하는 방법도 필요' 등의 의견을 제시받았고, 이에 따라 MobileNetV2를 Backbone으로 하는 학습 모델을 구현해 높은 Accuracy와 낮은 Loss를 가지는 신뢰도 높은 병해 Classification 모델을 구성할 수 있었다.

## 6. 향후개선방안

미해결 Issue: 현재 프로젝트 <플랜트리>에서 Cover할 수 있는 병해의 종류가 제한적이고, 해당 병해가 농작물군에 편중되어 있는 문제가 있다.

보완사항: '식물 집사'로 기대되는 사용자가 일반적으로 키우는 식물('다육이' 등)에 대한 이미지 데이터를 수집해 해당 식물들의 병해에 대해서도 Cover할 수 있도록 한다.

향후계획: 사용자의 식물 이미지로부터 개별 잎에 대한 Seperation 및 병해 Classification이 가능하도록, 전처리 과정을 개선해 개별 잎의 병해 판독이 가능하도록 한다.

캡스톤디자인I 11팀 "양정이조"

조형준 | 정다경 | YANG YUNLONG | 이택민 배상.

감사합니다.