

Software project deployment and user manual

Table of contents

- [Deploy the project to the server](#)
 - [Step1: create a server in nectar](#)
 - [Step2: set the security group](#)
 - [Step3: Generate the key](#)
 - [Step4: connect to the server](#)
 - [Step5: grant permissions to the folder](#)
 - [Step6: mport the project locally or clone the project from GitHub to the server](#)
 - [Step7: start the project](#)
 - [Step8: Run Python service](#)
 - [Step9: connecting Farmbot](#)
- [Added or modified code\(Partial code\)](#)
- [User manual for new features of webapp](#)



Deploy the project to the server

Step1: create a server in nectar

Follow the tutorial below to create a server in nectar

Nectar : <https://dashboard.rc.nectar.org.au/project/instances/>

Displaying 1 item

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavour	Key Pair	Status	Availability Zone	Task	Power State	Age	Ac
<input type="checkbox"/>	testInstance	NeCTAR Ubuntu 22.04 LTS (Jammy) amd64	qld-data 10.255.134.219 qld 203.101.230.232	m3.small	0502	Active	 QRISCloud	None	Running	1 day, 2 hours	

Displaying 1 item

Tutorial: <https://tutorials.rc.nectar.org.au/launching-virtual-machines/01-overview>



Step2: set the security group

After creating an instance, set the security group to allow all ports to be opened, or open ports as needed.

Security Groups	
http	ALLOW IPv6 to ::/0 ALLOW IPv4 to 0.0.0.0/0 ALLOW IPv4 tcp from 0.0.0.0/0 ALLOW IPv4 443/tcp from 0.0.0.0/0 ALLOW IPv4 80/tcp from 0.0.0.0/0
icmp	ALLOW IPv4 to 0.0.0.0/0 ALLOW IPv4 icmp from 0.0.0.0/0 ALLOW IPv6 to ::/0
default	ALLOW IPv4 from default ALLOW IPv4 to 0.0.0.0/0 ALLOW IPv6 from default ALLOW IPv6 to ::/0
ssh	ALLOW IPv4 to 0.0.0.0/0 ALLOW IPv4 22/tcp from 0.0.0.0/0 ALLOW IPv6 to ::/0

Step3: Generate the key

Generate the key file for the server and change the suffix of the private key to pem

 0502private.pem	2024/5/3 16:34
 0502public.txt	2024/5/3 16:34

Step4: connect to the server

Use ssh command and private key to connect to the server

```
ssh -i 0502private.pem ubuntu@203.101.230.232
```

Step5: grant permissions to the folder

After entering the main folder, create a new folder and grant permissions to the folder

```
cd/  
sudo mkdir webapp  
sudo chown -R ubuntu:ubuntu /webapp
```

Step6: mport the project locally or clone the project from GitHub to the server

Sample command for importing a project locally (this command is entered locally):

```
scp -i C:\Users\28482\Desktop\0502private.pem -r C:\Users\28482\Desktop\FA-Wombat ubuntu@203.101.230.232:/webapp
```

Example command for cloning a GitHub project (enter this command on the server):

```
git clone --branch thermal-page https://ghp_7d5HxgthqTtU9L0omAUTI94ETMY03G0Lo94m@github.com/COMP90082-2024-SM1/FA-Wombat.git
```

Step7: start the project

The command to start the project refers to the official webapp startup process

https://github.com/FarmBot/Farmbot-Web-App/blob/staging/ubuntu_example.sh

Install Docker

```
# Remove old (possibly broke) docker versions
sudo apt remove docker-engine
sudo apt remove docker docker.io containerd runc

# Install docker and docker compose
sudo apt update
sudo apt install ca-certificates curl gnupg -y
source /etc/os-release
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/$ID/gpg | sudo gpg --dearmor -o /etc/apt/keyring
s/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://
download.docker.com/linux/$ID $VERSION_CODENAME stable" | sudo tee /etc/apt/sources.list.d/do
cker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-pl
ugin -y
# Verify installation
sudo docker run hello-world
sudo docker compose version
```

Configure the environment file and change the IP address to the IP address of your own server.

```
nano .env
```

After entering the project folder, install dependencies and start

```
cd Farmbot-Web-App
# Install the correct version of bundler for the project
sudo docker compose run web gem install bundler
# Install application specific Ruby dependencies
sudo docker compose run web bundle install
# Install application specific Javascript deps
sudo docker compose run web npm install
```

```
# Create a database in PostgreSQL
sudo docker compose run web bundle exec rails db:create db:migrate
# Generate a set of *.pem files for data encryption
sudo docker compose run web rake keys:generate # ⚠ SKIP THIS STEP IF UPGRADING!
# Build the UI assets via ParcelJS
sudo docker compose run web rake assets:precompile
# Run the server! ٩(^_^)٩
# NOTE: DONT TRY TO LOGIN until you see a message similar to this:
#   "✨ Built in 44.92s"
# THIS MAY TAKE A VERY LONG TIME ON SLOW MACHINES (~3 minutes on DigitalOcean)
# You will just get an empty screen otherwise.
# This only happens during initialization
sudo docker compose up
```

The possibility of failure during this period may be due to the lack of permissions for the folder. Please refer to the following command to grant permissions

```
sudo chown -R ubuntu:ubuntu /webapp
```

Step8: Run Python service

Enter the folder

```
cd /webapp/FA-Wombat/src/PythonServer
```

Install Python dependencies

```
sudo apt install python3-pip

pip install Flask

pip install numpy

pip install flask-cors
```

Run Python service

```
nohup python3 Imagetransform.py > output.log 2>&1 &
```

How to close the Python service process? Find the Python service process number and kill it.

```
ps -ef | grep [Imagetransform.py](http://imagetransform.py/)
kill 1510371
```

Step9: connecting Farmbot

The method of connecting Farmbot to this webapp is the same as the official webapp, please refer to <https://software.farm.bot/v15/docs/getting-started>

It is worth noting that the email address for account registration must be xxx@student.unimelb.edu.au. This setting can be modified in the .env file

Added or modified code(Partial code)

1: Modify the code add a new tab

```
src/Farmbot-Web-App-staging/frontend/farm_designer/panel_header.tsx
```

```
export enum Panel {
  Map = "Map",
  Plants = "Plants",
  Weeds = "Weeds",
  Points = "Points",
  Groups = "Groups",
  Curves = "Curves",
  SavedGardens = "SavedGardens",
  Sequences = "Sequences",
  Regimens = "Regimens",
  FarmEvents = "FarmEvents",
  Zones = "Zones",
  Controls = "Controls",
  Sensors = "Sensors",
  Photos = "Photos",
  Farmware = "Farmware",
  Tools = "Tools",
  Messages = "Messages",
  Logs = "Logs",
  Help = "Help",
  Settings = "Settings",
  Shop = "Shop",
  Thermal = "Thermal",
}

export const TAB_COLOR: Record<Panel, PanelColor> = {
  [Panel.Map]: PanelColor.gray,
  [Panel.Plants]: PanelColor.green,
  [Panel.Weeds]: PanelColor.red,
  [Panel.Points]: PanelColor.teal,
  [Panel.Groups]: PanelColor.blue,
  [Panel.Curves]: PanelColor.gray,
  [Panel.Sequences]: PanelColor.gray,
  [Panel.Regimens]: PanelColor.gray,
  [Panel.SavedGardens]: PanelColor.navy,
  [Panel.FarmEvents]: PanelColor.yellow,
  [Panel.Zones]: PanelColor.brown,
  [Panel.Controls]: PanelColor.gray,
  [Panel.Sensors]: PanelColor.gray,
  [Panel.Photos]: PanelColor.gray,
  [Panel.Farmware]: PanelColor.gray,
  [Panel.Tools]: PanelColor.gray,
  [Panel.Messages]: PanelColor.gray,
```

```

[Panel.Logs]: PanelColor.gray,
[Panel.Help]: PanelColor.gray,
[Panel.Settings]: PanelColor.gray,
[Panel.Shop]: PanelColor.gray,
[Panel.Thermal]: PanelColor.gray,
};

export const TAB_ICON: Record<Panel, string> = {
  [Panel.Map]: FilePath.icon(Icon.map),
  [Panel.Plants]: FilePath.icon(Icon.plant),
  [Panel.Weeds]: FilePath.icon(Icon.weeds),
  [Panel.Points]: FilePath.icon(Icon.point),
  [Panel.Groups]: FilePath.icon(Icon.groups),
  [Panel.Curves]: FilePath.icon(Icon.curves),
  [Panel.Sequences]: FilePath.icon(Icon.sequence),
  [Panel.Regimens]: FilePath.icon(Icon.regimens),
  [Panel.SavedGardens]: FilePath.icon(Icon.gardens),
  [Panel.FarmEvents]: FilePath.icon(Icon.calendar),
  [Panel.Zones]: FilePath.icon(Icon.zones),
  [Panel.Controls]: FilePath.icon(Icon.controls),
  [Panel.Sensors]: FilePath.icon(Icon.sensors),
  [Panel.Photos]: FilePath.icon(Icon.photos),
  [Panel.Farmware]: FilePath.icon(Icon.farmware),
  [Panel.Tools]: FilePath.icon(Icon.tool),
  [Panel.Messages]: FilePath.icon(Icon.messages),
  [Panel.Logs]: FilePath.icon(Icon.logs),
  [Panel.Help]: FilePath.icon(Icon.help),
  [Panel.Settings]: FilePath.icon(Icon.settings),
  [Panel.Shop]: FilePath.icon(Icon.shop),
  [Panel.Thermal]: FilePath.icon(Icon.thermal),
};

export const PANEL_SLUG: Record<Panel, string> = {
  [Panel.Map]: "",
  [Panel.Plants]: "plants",
  [Panel.Weeds]: "weeds",
  [Panel.Points]: "points",
  [Panel.Groups]: "groups",
  [Panel.Curves]: "curves",
  [Panel.Sequences]: "sequences",
  [Panel.Regimens]: "regimens",
  [Panel.SavedGardens]: "gardens",
  [Panel.FarmEvents]: "events",
  [Panel.Zones]: "zones",
  [Panel.Controls]: "controls",
  [Panel.Sensors]: "sensors",
  [Panel.Photos]: "photos",
  [Panel.Farmware]: "farmware",
  [Panel.Tools]: "tools",
  [Panel.Messages]: "messages",
  [Panel.Logs]: "logs",
  [Panel.Help]: "help",
  [Panel.Settings]: "settings",
  [Panel.Shop]: "shop",
  [Panel.Thermal]: "thermal",
};

export const PANEL_TITLE = (): Record<Panel, string> => ({
  [Panel.Map]: t("Map"),
  [Panel.Plants]: t("Plants"),

```



```

[Panel.Weeds]: t("Weeds"),
[Panel.Points]: t("Points"),
[Panel.Groups]: t("Groups"),
[Panel.Curves]: t("Curves"),
[Panel.Sequences]: t("Sequences"),
[Panel.Regimens]: t("Regimens"),
[Panel.SavedGardens]: t("Gardens"),
[Panel.FarmEvents]: t("Events"),
[Panel.Zones]: t("Zones"),
[Panel.Controls]: t("Controls"),
[Panel.Sensors]: t("Sensors"),
[Panel.Photos]: t("Photos"),
[Panel.Farmware]: t("Farmware"),
[Panel.Tools]: t("Tools"),
[Panel.Messages]: t("Messages"),
[Panel.Logs]: t("Logs"),
[Panel.Help]: t("Help"),
[Panel.Settings]: t("Settings"),
[Panel.Shop]: t("Shop"),
[Panel.Thermal]: t("Thermal"),
});

render() {
  const tab = getCurrentPanel();
  const hidden = this.props.hidden ? "hidden" : "";
  const color = TAB_COLOR[tab || Panel.Plants];
  return <div className={`panel-nav ${color}-panel ${hidden}`}>
    {!this.state.atEnd && <div className="scroll-indicator" />}
    <div className="panel-tabs" onScroll={this.updateScroll}>
      <NavTab panel={Panel.Map} />
      <NavTab panel={Panel.Plants} />
      <NavTab panel={Panel.Weeds} />
      <NavTab panel={Panel.Points} />
      <NavTab panel={Panel.Curves} />
      <NavTab panel={Panel.Sequences} />
      <NavTab panel={Panel.Regimens} />
      <NavTab panel={Panel.FarmEvents} />
      {DevSettings.futureFeaturesEnabled() && <NavTab panel={Panel.Zones} />}
      {showSensors() && <NavTab panel={Panel.Sensors} />}
      <NavTab panel={Panel.Photos} />
      {showFarmware() && <NavTab panel={Panel.Farmware} />}
      <NavTab panel={Panel.Tools} />
      <NavTab panel={Panel.Messages} />
      <NavTab panel={Panel.Help} />
      <NavTab panel={Panel.Settings} />
      <NavTab panel={Panel.Thermal} />
    </div>
  </div>;
}

```

src/Farmbot-Web-App-staging/frontend/internal_urls.ts

```

export enum Icon {
  map = "map",
  plant = "plant",
  weeds = "weeds",
}

```

```

point = "point",
groups = "groups",
curves = "curves",
sequence = "sequence",
regimens = "regimen",
gardens = "gardens",
calendar = "calendar",
zones = "zones",
controls = "controls",
sensors = "sensors",
photos = "photos",
farmware = "farmware",
tool = "tool",
messages = "messages",
logs = "logs",
help = "help",
documentation = "documentation",
support = "support",
settings = "settings",
shop = "shop",
developer = "developer",
logout = "logout",
settings_small = "settings_small",
thermal = "thermal",
}

```

src/Farmbot-Web-App-staging/frontend/nav/nav_links.tsx

```

export const getLinks = (): Panel[] => [
  Panel.Map,
  Panel.Plants,
  Panel.Weeds,
  Panel.Points,
  Panel.Curves,
  Panel.Sequences,
  Panel.Regimens,
  Panel.FarmEvents,
  ...(showSensors() ? [Panel.Sensors] : []),
  Panel.Photos,
  ...(showFarmware() ? [Panel.Farmware] : []),
  Panel.Tools,
  Panel.Messages,
  Panel.Help,
  Panel.Settings,
  Panel.Thermal,
];

```

2: Add thermal interface

src/Farmbot-Web-App-staging/frontend/thermal

3 : Add a proxy

src/Farmbot-Web-App-staging/setupProxy.js

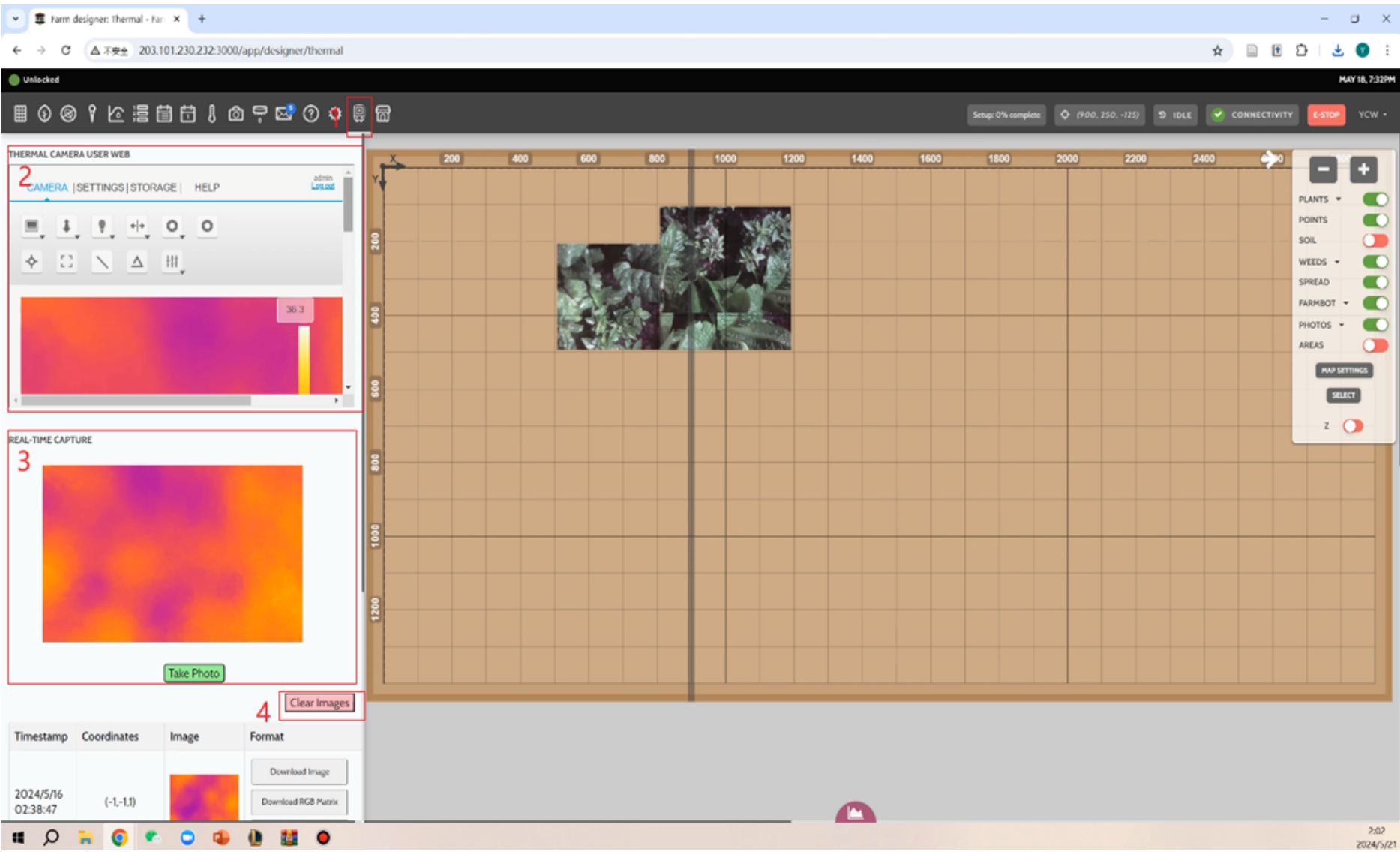
src/Farmbot-Web-App-staging/app/controllers

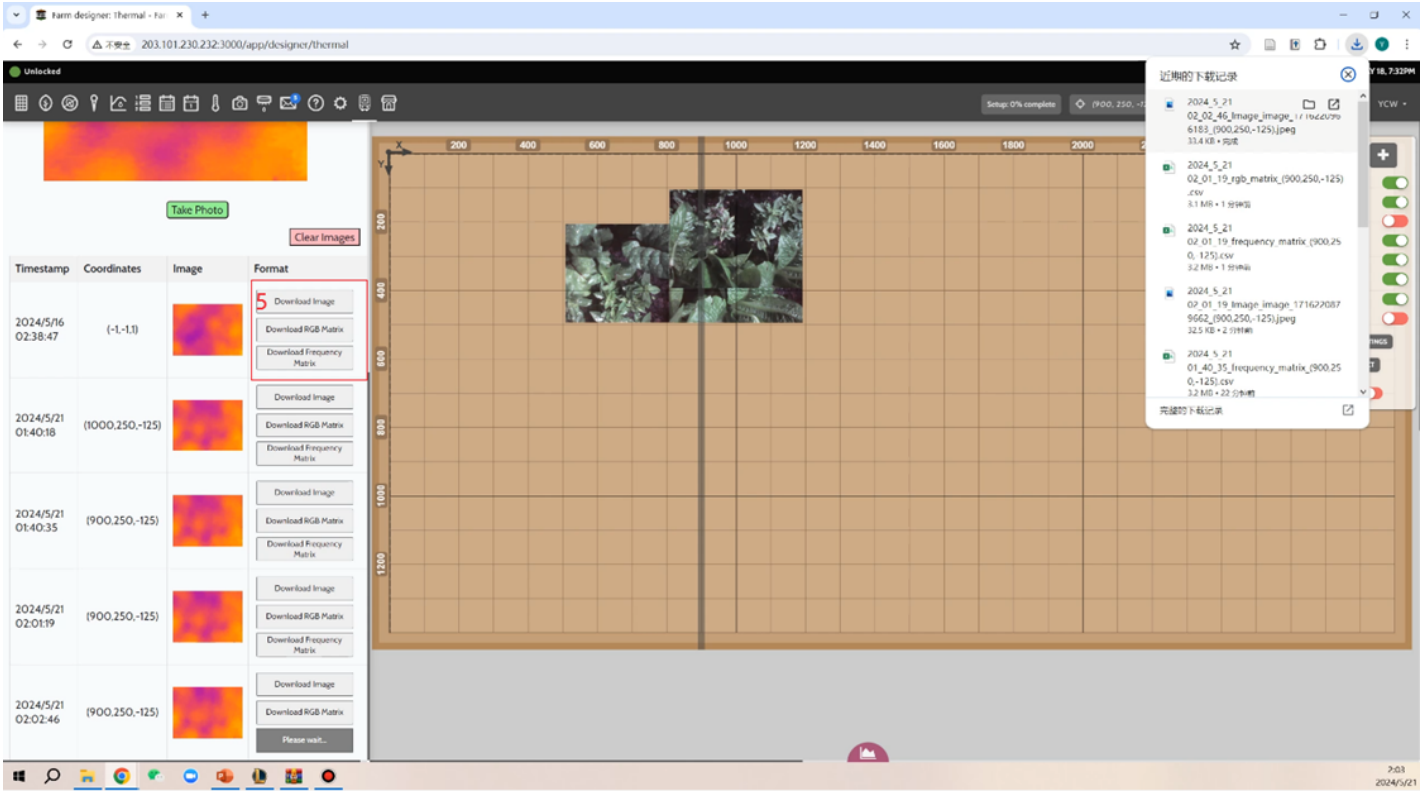
modify code : src/Farmbot-Web-App-staging/frontend/api/api.ts

```
get getImage() {
  return `${this.baseUrl}/api/proxy_to_external_server`
}
get getImageRGBMatrix() {
  return `${this.baseUrl}/api/upload_img_to_external_server`
}
}
```

4 : Add a python server
src/PythonServer

User manual for new features of webapp





1	New Thermal Imaging tab
2	Integrated thermal imaging camera management web, users can adjust camera settings
3	Clicking the green button will take a thermal image and save it in the table below.
4	The red button clears all data in the table below
5	Users can choose to download the image in different formats