PURDUE
UNIVERSITY
Computer Graphics Technology
Polytechnic Institute CGT 575 / ASM 591 Data Visualization Tools & Applications Spring 2024

Week 7
Deploying Deep Learning Models on Web and Smartphone Applications Using Streamlit API
Thursday, February22nd 5:30 PM – 6:45 PM
DUE: Thursday, February 29th

Streamlit

Figure 1. Web and Smartphone Applications for Real-Time Deep Learning (https://tinyurl.com/bdh36zyb, https://tinyurl.com/3fc9wby6)

DESCRIPTION

Although training deep learning models and extracting image meta data are important skills that can be applied for various domains, it is important to deploy and extend these skills for the end-user by developing a web or smartphone application. Therefore, in this lab, you will be using the Streamlit Python API for developing a web application. You will be deploying a deep learning model, allowing users input images, make predictions, and map geocoordinates.

PREREQUISITES

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

1. Complete Labs 1, 2, 3, 4, and 5

LEARNING OBJECTIVES

By the end of this lab, students will complete the following:

1. Setup Streamlit onto the edge device and launch a basic application
2. Create additional pages
3. Deploy the image classification model trained in an earlier lab onto the web application
4. Create a map by extracting image metadata on the web application
5. Homework Assignment

ASSIGNMENT

<u>Objective 1: Setup Streamlit onto the edge device and launch a basic application</u>

1. Power ON the edge device
2. First install Visual Studio Code on the edge device **if you do not have it installed already**:
    a. Open the browser
    b. Go to: https://code.visualstudio.com/download
    c. Download the version circled in **RED** below

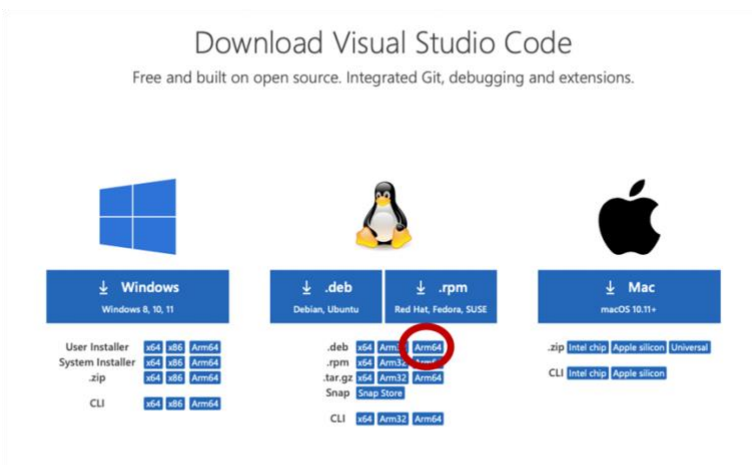

Figure 2. Installing Visual Studio Code

3. Now launch the terminal
4. Activate the anaconda environment and install Streamlit (**figure 3**):
    a. Activate the anaconda environment used in the last two labs by typing conda activate dl and press "enter"

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

b. To install Streamlit, type pip install streamlit==0.84.2 and press “enter”

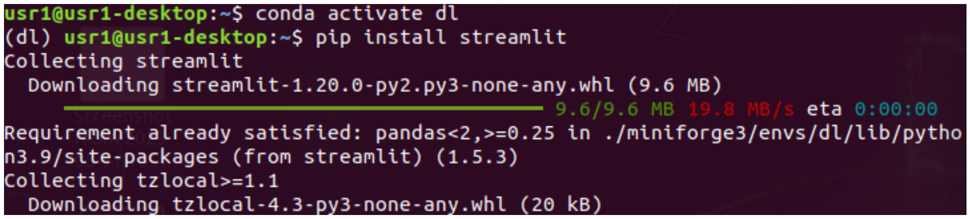

Figure 3. Install Streamlit

c. Type pip install streamlit-folium and press “enter”

d. Type pip install plotly and press “enter”

5. Navigate to the correct folder and create a Python file (**figure 4**):
   a. Create a new directory called lab 6 by typing mkdir lab6 and press “enter”
   b. Navigate to the new directory by typing cd lab6 and press “enter”
   c. Type gedit main.py and press “enter”
   d. First copy the code shown in **figure 5**.
   e. Save the file and close the Python window

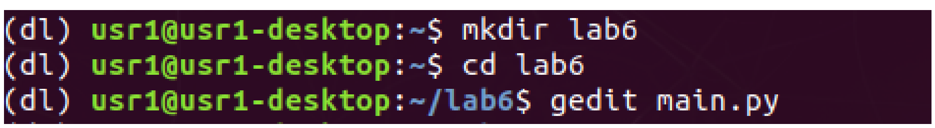

Figure 4. Create a Lab 6 Folder and enter the dl anaconda environment

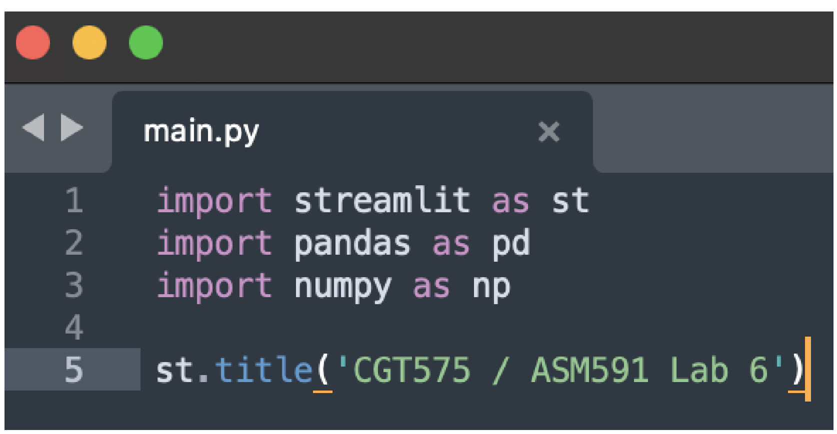

Figure 5. main.py starter code

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

6. To launch the application, type streamlit run main.py in the terminal and press “enter” (**figure 6**). The browser will automatically launch, and the application home page will appear, as shown in **figure 7**.

```
(dl) usr1@usr1-desktop:~/lab6$ streamlit run main.py
```

Figure 6. Run the application

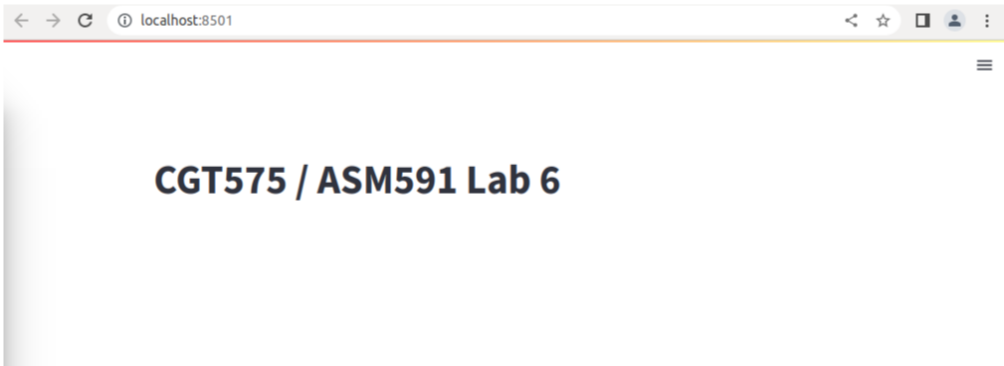

Figure 7. Application homepage after starter code

7. Open Visual Studio Code and open the main.py file within the Lab6 folder. This will allow you to edit the code while you are also using the terminal.
8. Import additional libraries as shown in **figure 8**:

```
main.py

import streamlit as st
import pandas as pd
import numpy as np
import tensorflow as tf
import keras
import folium
from exif import Image as exif
import geopandas as gpd
import plotly.express as px
import matplotlib.pyplot as plt
from streamlit_folium import folium_static
from PIL import Image

st.title('CGT575 / ASM591 Lab 6')
```

Figure 8. Import additional libraries

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

9. After adding code, you do not need to stop and run the application. Simply go to your browser where the application is running and click on the "Rerun" button on the top right corner of the page as shown in **figure 9**.

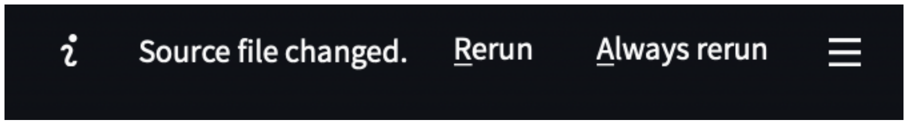

Figure 9. Rerun application

10. Reference: https://docs.streamlit.io/library/get-started/create-an-app
11. **NOTE: After every modification made to the code in main.py Python file, save the code by typing "ctrl+s" and press "enter". You will only be able to rerun the application from the browser (shown in figure 9) after saving the code.**

Objective 2: Create additional pages

1. Now you will create three pages for the web application
2. Create a sidebar as shown in **figure 10** by using the Streamlit sidebar function: st.sidebar.selectbox(). Type the code shown in **figure 10** within the main.py Python file after the line with st.title().

```
# SIDEBAR
task = st.sidebar.selectbox("Select Task: ", ("Homepage", "Deep Learning", "Mapping"))
st.write(task)
```

Figure 10. Create sidebar

3. You will create 3 pages in this lab: "Homepage", "Deep Learning", and "Mapping"
4. Once you rerun the application, the sidebar will appear on the left side of the screen as shown in figure 11.

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

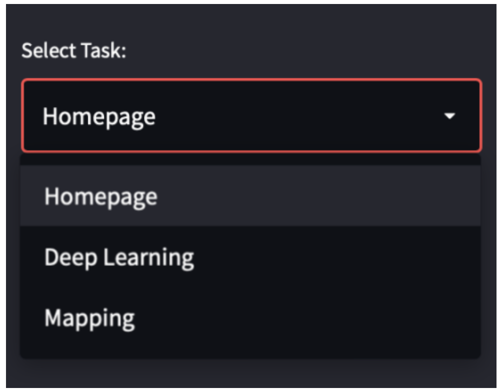

Figure 11. Sidebar

5. Create a Python function named pages() as shown in figure 12 after the line with st.write(). This function will be used to design the different pages in your application. The three pages that were created were "Homepage", "Deep Learning", and "Mapping". The code for designing the "Homepage" has been provided where your application will show an interactive map. You will popular the code for designing "Deep Learning" and "Mapping" pages in Objectives 3 and 4, respectively.

```python
def pages(task):
    if task == "Homepage":
        st.title('Lab 6')
        lat = 40.424146
        lon = -86.918105
        map_data = pd.DataFrame({'lat': [lat], 'lon': [lon]})
        st.map(map_data)

    elif task == "Deep Learning":

        st.title(task)

    elif task == "Mapping":

        st.title(task)


pages(task)
```

Figure 12. Function to modify the different pages

6. Call the Python function at the end of the file as circled in **RED** in **figure 12**.
7. Now the application will show the Homepage with a map as shown in **figure 13**.

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

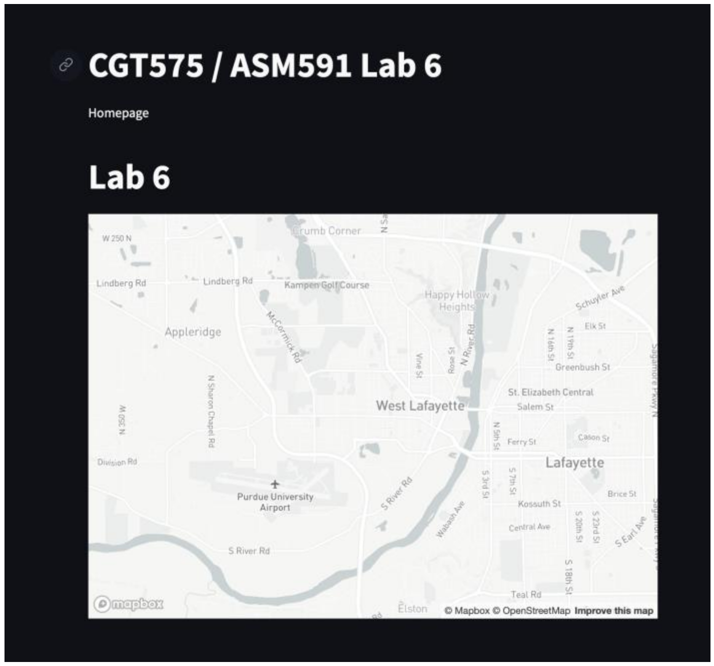

Figure 13. Homepage with map of Purdue University

Objective 3: Deploy the image classification model trained in an earlier lab onto the web application

1. Import additional libraries required at the top of the Python file as circled in **RED** in **figure 14**:
   a. VGG16 is the Image Classification model which is trained to identify everyday items
   b. image is used for loading images
   c. preprocess_input is used for preprocessing the image for making predictions
   d. decode_predictions will help identify the classes and their probabilities

```
main.py
import streamlit as st
import pandas as pd
import numpy as np
import tensorflow as tf
import keras
import folium
from exif import Image as exif
import geopandas as gpd
import plotly.express as px
import matplotlib.pyplot as plt
from streamlit_folium import folium_static
from PIL import Image

# deep learning libraries
from keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input,decode_predictions
```

Figure 14. Add additional libraries for deep learning

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

2. Download an image from Google of any object (e.g., car, dog, cat, etc.) and save the image into the Lab6 folder using a simple name (e.g., car.jpg).
3. Within the pages() function where task is "Deep Learning", you will now use a deep learning model to make predictions and load the predictions on the Web Application (**figure 15**):
    a. Load the VGG16 model with ImageNet weights using the model = VGG16(weights='imagenet')
    b. Provide the image path using the image name that was used for saving the file and place it as shown in figure 15 where img_path = 'car.jpg'
    c. Load the image in correct color format and target size: img = image.load_img(img_path, color_mode='rgb', target_size=(224, 224))
    d. Convert the PIL image into a numpy array using the code line: x = image.img_to_array(img)
    e. Expand the dimensions of the image: x = np.expand_dims(x, axis=0)
    f. Preprocess the image: x = preprocess_input(x)
    g. Obtain the features after predicting the item in the image: features = model.predict(x)
    h. Identify the class name and probability: p = decode_predictions(features)

```python
def pages(task):
    if task == "Homepage":
        st.title('Lab 6')
        lat = 40.424146
        lon = -86.918105
        map_data = pd.DataFrame({'lat': [lat], 'lon': [lon]})
        st.map(map_data)

    elif task == "Deep Learning":

        st.title(task)
        model = VGG16(weights='imagenet')
        img_path = 'car.jpg'
        img = image.load_img(img_path, color_mode='rgb', target_size=(224, 224))
        x = image.img_to_array(img)  # convert PIL image into numpy array
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
        features = model.predict(x)
        p = decode_predictions(features)

        st.image(img)

        for preds in p[0]:
            st.write('The image is ' + preds[1] + ' with probability: ' + str(preds[2]))
```

Figure 15. Deep Learning page

    i. To display the image on the website, use the st.image() function from the Streamlit API
    j. Finally, output the predictions correctly using st.write() function as shown in **figure 15**

4. Rerun the application and navigate to the Deep Learning page from the sidebar (**figure 16**)

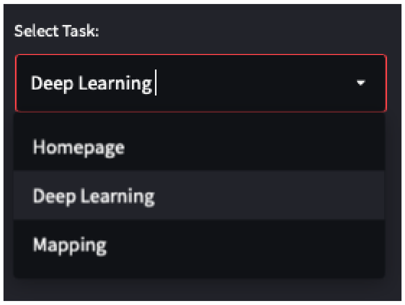

Figure 16. Select Deep Learning Tab in the sidebar

5. Finally, you will see your image and the predictions from the deep learning model on a web application as shown in **figure 17**

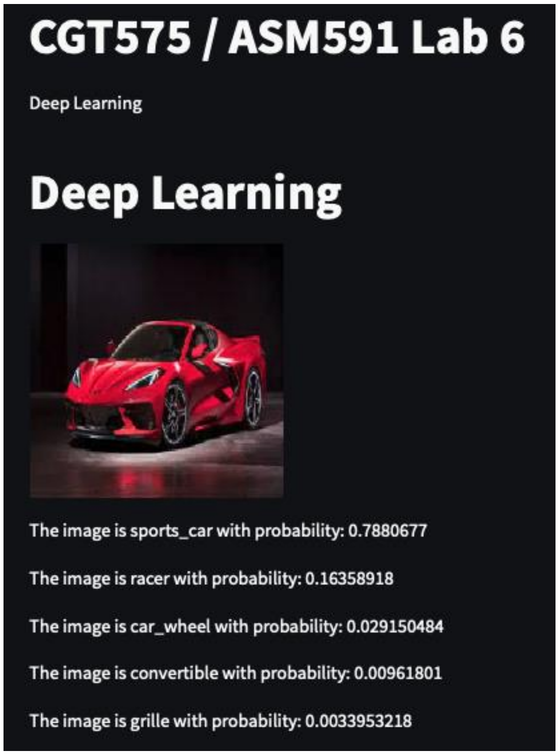

Figure 17. Result of the Deep Learning page

6. Reference: https://towardsdatascience.com/how-to-use-a-pre-trained-model-vgg-for-image-classification-8dd7c4a4a517
7. Now you will also allow users to upload images for deep learning-based image classification by adding code in this step below the part circled in **RED** in **figure 15**:

a. Upload the image by using the st.file_uploader() function as shown in **figure 18**. The if statement is added to confirm if whether or not a file is uploaded.

b. Using the PIL library open the image using the Image.open() function as shown in figure

```
# allow user to upload files
uploaded_file = st.file_uploader("Upload Image")
if uploaded_file is not None:

    # display the image
    display_image = Image.open(uploaded_file)
    st.image(display_image)
```

Figure 18. Allow users to upload images and read the image

c. Resize and preprocess the uploaded image as shown in **figure 19**.

```
# Resize the image for tensorflow predicition
temp_img = display_image.resize((224, 224), Image.ANTIALIAS)
img_tensor = tf.keras.preprocessing.image.img_to_array(temp_img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor /= 255.
```

Figure 19. Image preprocessing for deep learning

d. Make the predictions and display them on the web application using the code shown in **figure 20**.

```
# Use the deep learning model to make the prediction and highlight the section of the image
prediction = model.predict(img_tensor)
p2 = decode_predictions(prediction)
for preds in p2[0]:
    st.write('The image is ' + preds[1] + ' with probability: ' + str(preds[2]))
```

Figure 20. Made predictions from uploaded image

e. The overall code for the "Deep Learning" page is shown in **figure 21**. **NOTE: Make sure the indentation of the code is correct.**

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

```python
elif task == "Deep Learning":

    st.title(task)
    model = VGG16(weights='imagenet')
    img_path = 'car.jpg'
    img = image.load_img(img_path, color_mode='rgb', target_size=(224, 224))
    x = image.img_to_array(img)  # convert PIL image into numpy array
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    features = model.predict(x)
    p = decode_predictions(features)

    st.image(img)

    for preds in p[0]:
        st.write('The image is ' + preds[1] + ' with probability: ' + str(preds[2]))


    # allow user to upload files
    uploaded_file = st.file_uploader("Upload Image")
    if uploaded_file is not None:

        # display the image
        display_image = Image.open(uploaded_file)
        st.image(display_image)

        # Resize the image for tensorflow predicition
        temp_img = display_image.resize((224, 224), Image.ANTIALIAS)
        img_tensor = tf.keras.preprocessing.image.img_to_array(temp_img)
        img_tensor = np.expand_dims(img_tensor, axis=0)
        img_tensor /= 255.

        # Use the deep learning model to make the prediction and highlight the section of the image
        prediction = model.predict(img_tensor)
        p2 = decode_predictions(prediction)
        for preds in p2[0]:
            st.write('The image is ' + preds[1] + ' with probability: ' + str(preds[2]))
```

Figure 21. Overall code for step 7

f. Users can upload images by clicking on the "Browse Files" button circled in **RED** in **figure 22**.

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

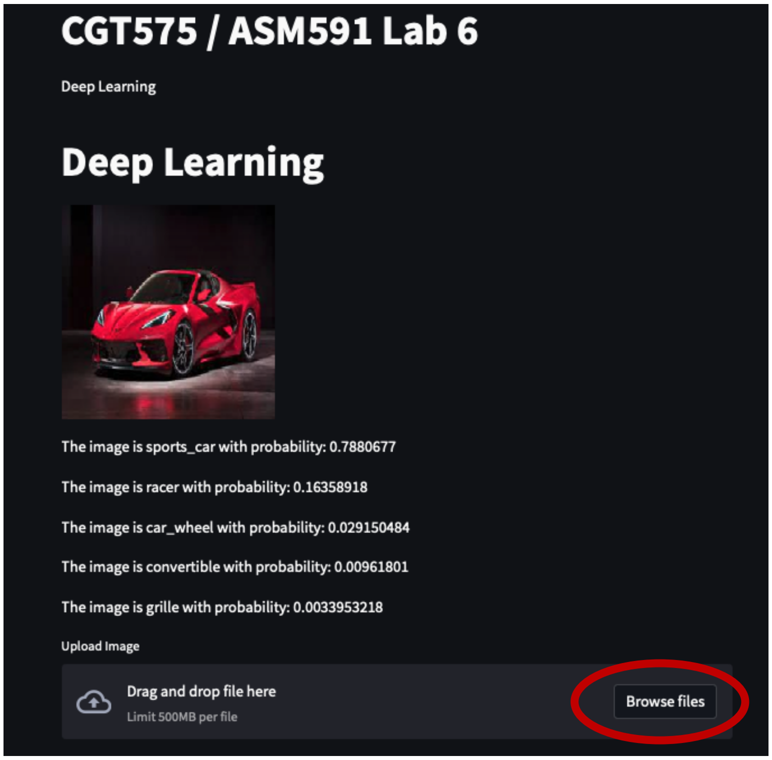

Figure 22. Deep Learning page

<u>Objective 4: Create a map on the web application</u>

1. In this part of the lab, you will be creating a map on the "Mapping" page. Additionally, you will be adding a marker on the map corresponding to the geolocation (**figure 23**):
    a. The latitude and longitude values are shown
    b. As you created maps using folium in Lab 5, use folium to create a map using the folium.Map() function
    c. Create a marker on the map using the folium.Marker() function and replace the popup and tooltip text with "Lab6"
    d. To display the map, use the folium_static() function

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

```python
elif task == "Mapping":

    st.title(task)
    lat = 40.424146
    lon = -86.918105
    m = folium.Map (location=[lat,lon], zoom_start=15)
    folium.Marker([lat,lon], popup="Field", tooltip="Diseased Field").add_to(m)

    # how to show map on the webapplication
    folium_static(m)
```

Figure 23. Code for the Mapping page of the web application

2. Finally, the Mapping page should look like the image shown in **figure 24**.

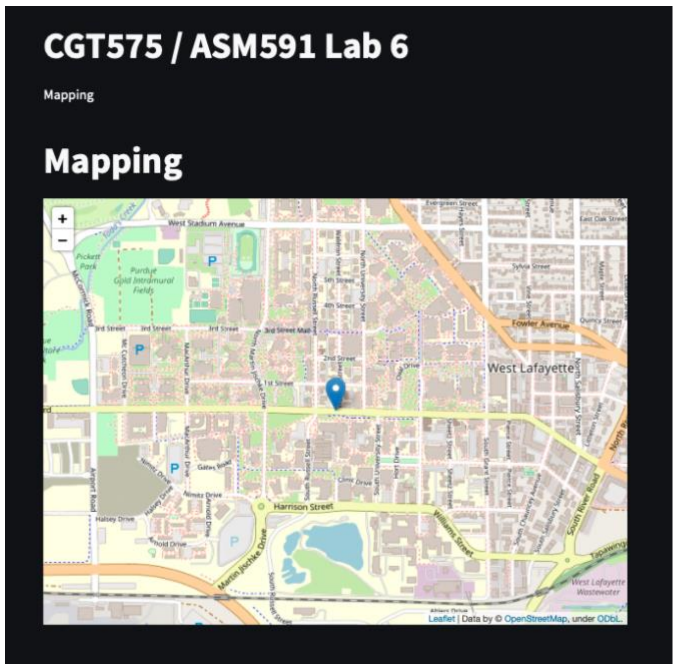

Figure 24. Mapping page

<u>Objective 5: Homework Tasks</u>

1. Modify to task = st.sidebar.selectbox() function from **figure 10** to add an additional page called "Homework"
2. You will also have to modify the pages() function by adding the line elif task == "Homework" at the end of the function in the same way the "Deep Learning" and "Mapping" pages were created (**figure 12**)
3. The page should allow users to upload image files using the uploaded_file = st.file_uploader("Upload Image") function

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

4. Upload one of the images that were provided for Lab 5 (**figure 25**)
5. Display the image on the page (**figure 25**)
6. Extract the geocoordinates (**figure 25**)

```
uploaded_file = st.file_uploader("Upload Image")
if uploaded_file is not None:
    # display the image
    st.image(uploaded_file)
    img = exif(uploaded_file)
    coords = (decimal_coords(img.gps_latitude,img.gps_latitude_ref), decimal_coords(img.gps_longitude,img.gps_longitude_ref))
```

Figure 25. Functions for extracting geocoordinates from files.

7. Using the geocoordinates, map them and drop a marker as you did in **Objective 4**
8. The final "Homework" page is shown in **figure 26**.

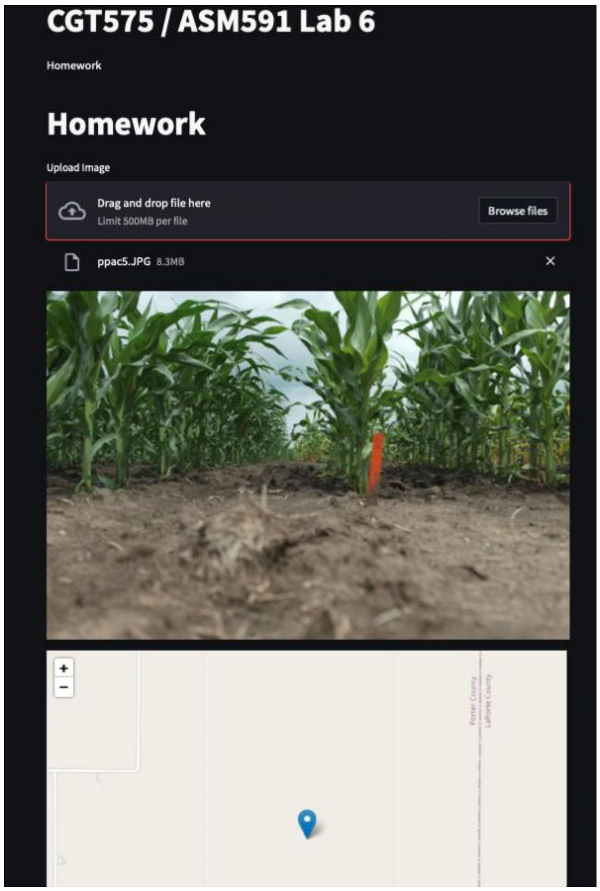

Figure 26. Homework page

9. You will submit:
   a. Screenshots for each of the pages that were created in this lab

Instructors:
Dr. Vetria Byrd (vlbyrd@purdue.edu)
Dr. Dharmendra Saraswat (saraswat@purdue.edu)
Dr. Aanis Ahmad (ahmad31@purdue.edu)
Temitope Ibrahim Amosa (tamosa@purdue.edu)

b. The “main.py” Python file

REFERENCES / ADDITIONAL RESOURCES

1. Creating dashboards in Streamlit: https://towardsdatascience.com/a-multi-page-interactive-dashboard-with-streamlit-and-plotly-c3182443871a
2. Interactive dashboard in Streamlit: https://www.turing.com/kb/how-to-build-an-interactive-dashboard-in-python-using-streamlit
3. Bubble chart dashboard in Streamlit: https://towardsdatascience.com/building-a-dashboard-in-under-5-minutes-with-streamlit-fd0c906ff886