FIT3164 - Data Science Software Project

User Guides

Workshop: Thursday, 2pm - 4pm

Group: FIT3163_CL_04

Members: Elaine Liong (ID: 29942357),

Jack Ooi (ID: 29037077),

Vionnie Tan (ID: 30092809)

Table of Contents

End User Guide

Introduction

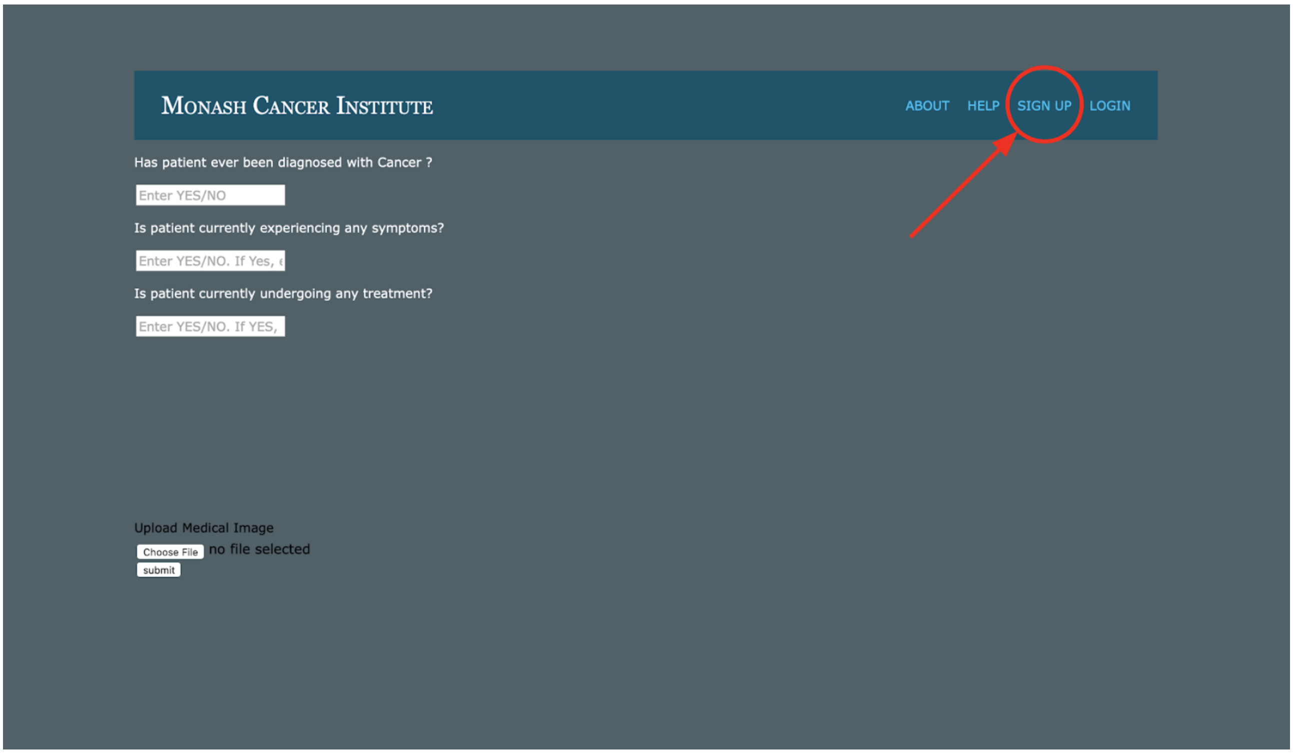
Our website provides the functionality to easily predict cancer in just a few clicks. This user guide will give step by step instructions on how to use our website such as how to sign up/ log in and how to predict cancer.

Accessing the Website

You can access our website through the link: https://cancerprediction-4.herokuapp.com/

Signing Up

- Click the SIGN UP button on the right side of the navigation bar.



Has patient ever been diagnosed with Cancer ?

Enter YES/NO

Is patient currently experiencing any symptoms?

Enter YES/NO. If Yes, (

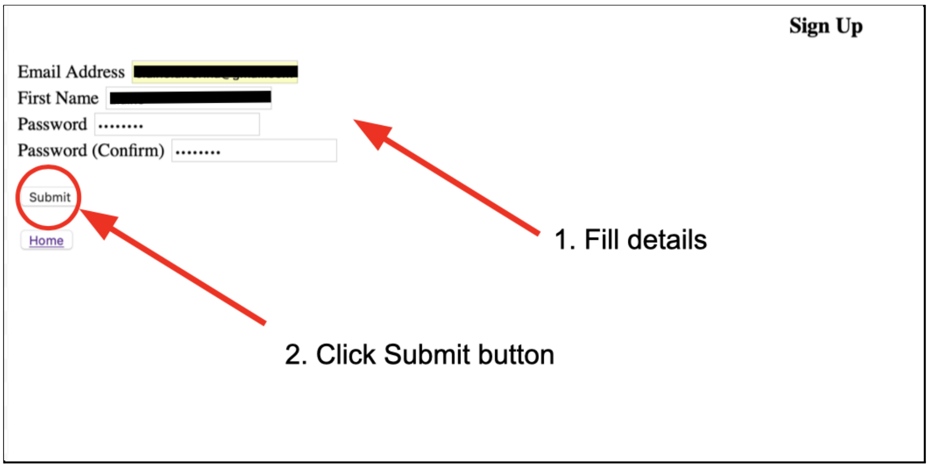
Is patient currently undergoing any treatment?

Enter YES/NO. If YES,

Upload Medical Image

no file selected

- Fill in your details (email address, first name, and password), then click the Submit button.



Sign Up

Email Address [redacted]

First Name [redacted]

Password ········

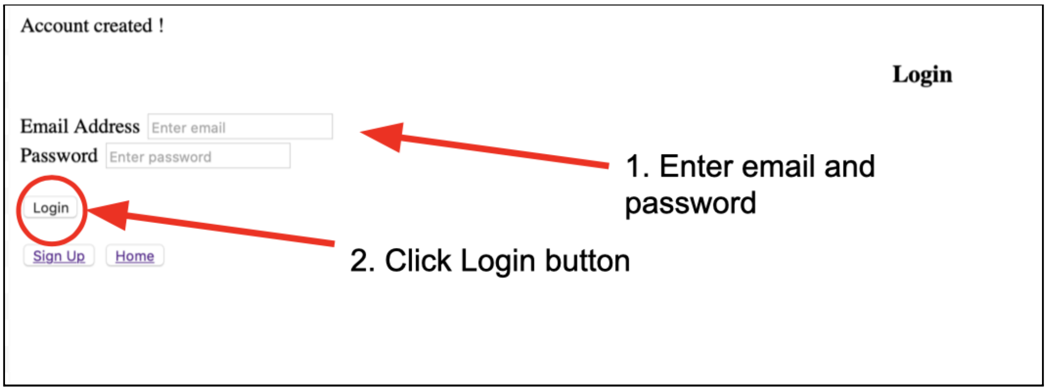
Password (Confirm) ········

1. Fill details

2. Click Submit button

Logging In

- Enter your email address and password then click the Login button



Account created !

Login

Email Address Enter email

Password Enter password

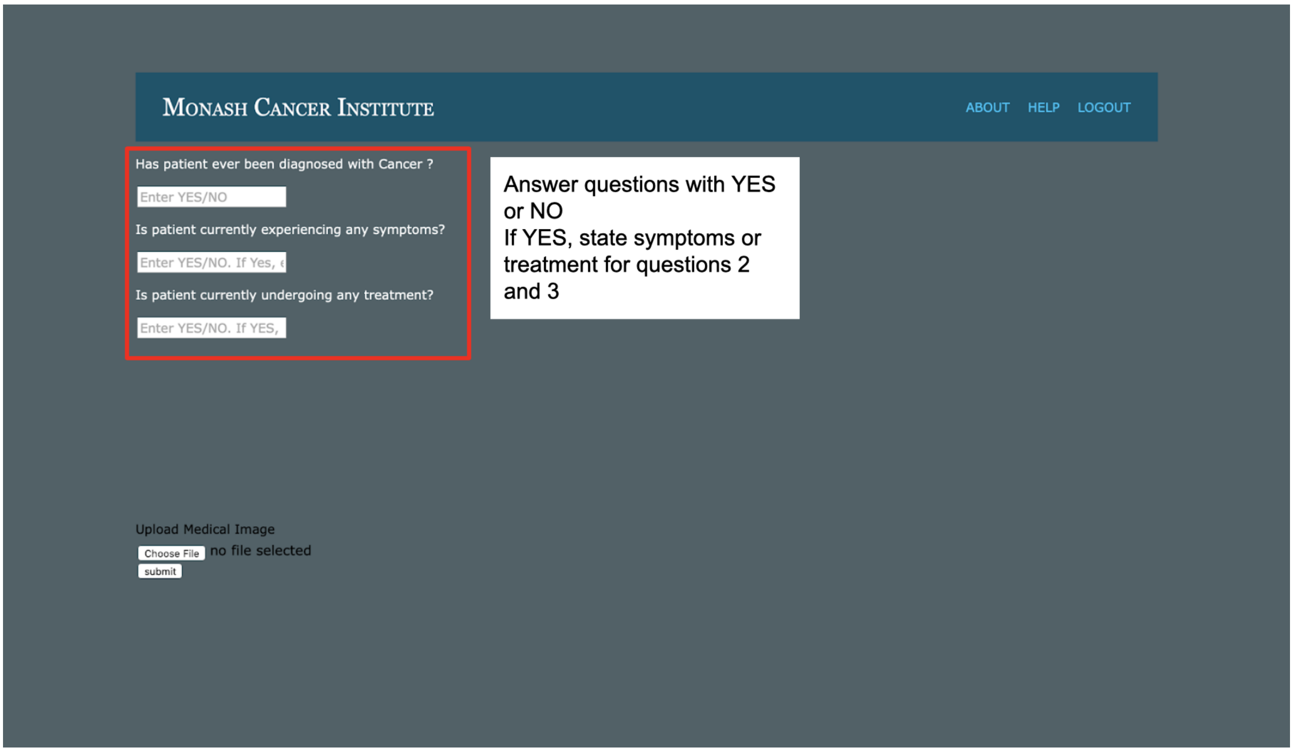
1. Enter email and password

2. Click Login button

Predicting Cancer

Filling up the form

- Fill in the questions on the home page.



Has patient ever been diagnosed with Cancer ?

Enter YES/NO

Is patient currently experiencing any symptoms?

Enter YES/NO. If Yes, e

Is patient currently undergoing any treatment?

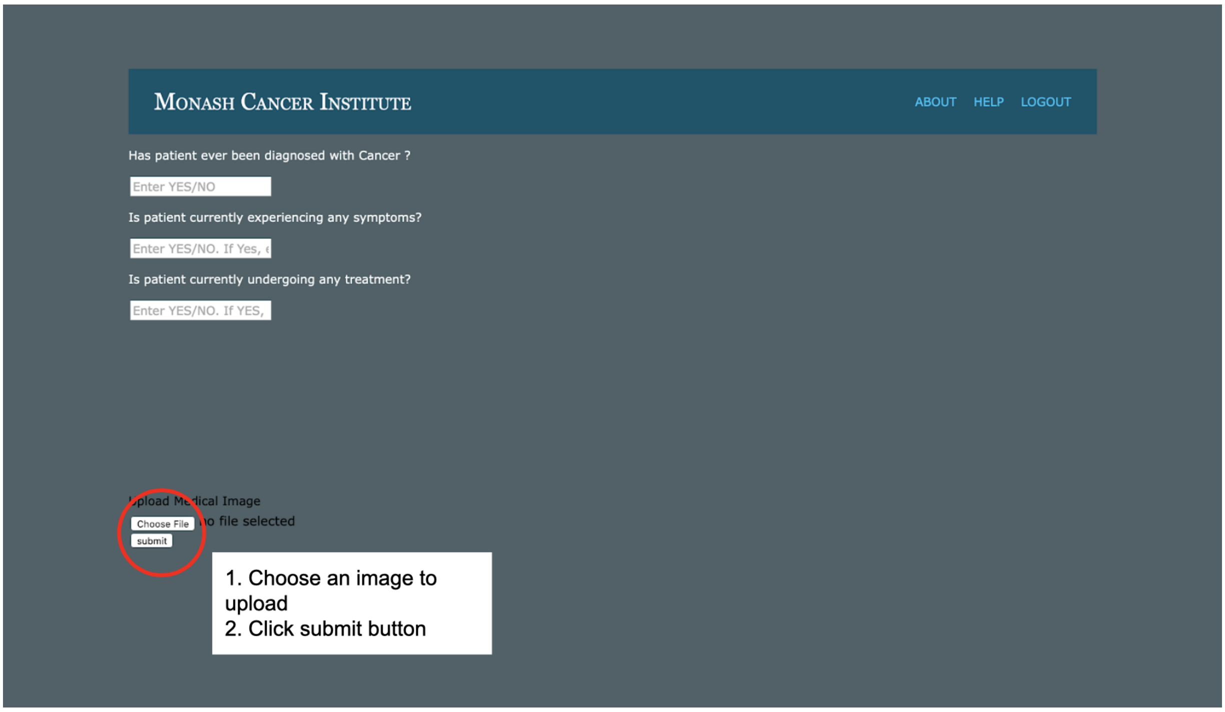
Enter YES/NO. If YES,

Answer questions with YES or NO
If YES, state symptoms or treatment for questions 2 and 3

Upload Medical Image

Uploading image and viewing result

- Choose an image from your device by clicking the Choose File button, then click the submit button once the image has been chosen



Has patient ever been diagnosed with Cancer ?

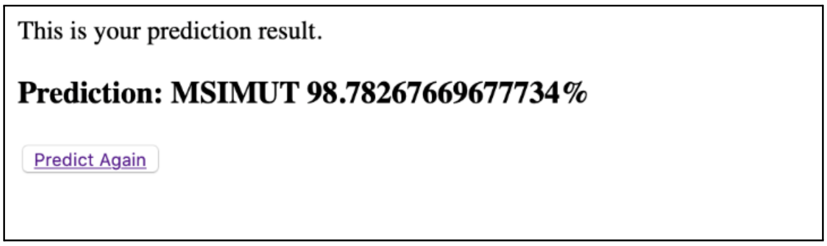
Is patient currently experiencing any symptoms?

Is patient currently undergoing any treatment?

Upload Medical Image

1. Choose an image to upload
2. Click submit button

- You will be redirected to the result page, where the prediction result is shown.



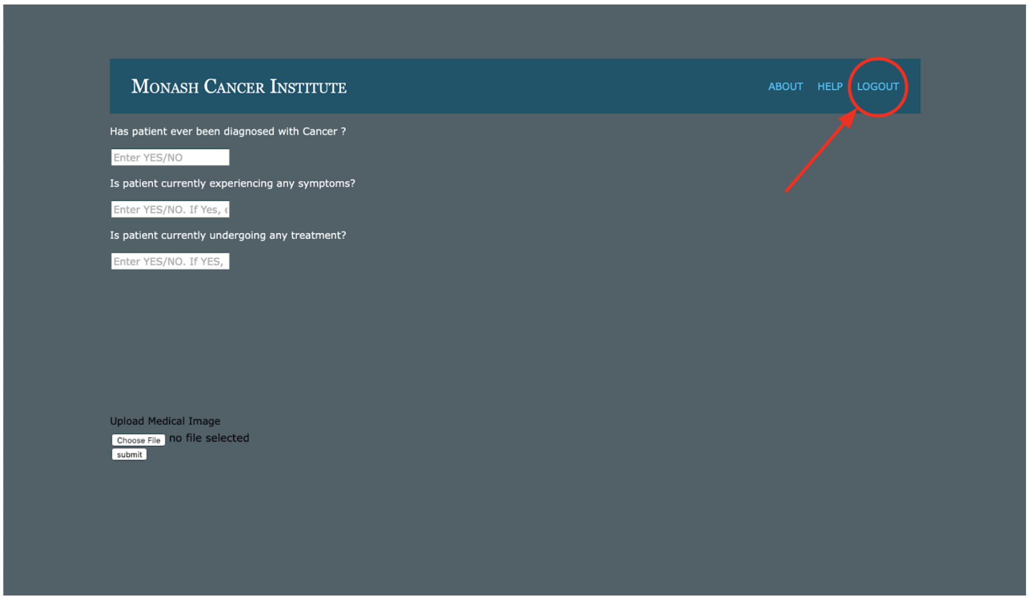
This is your prediction result.

Prediction: MSIMUT 98.78267669677734%

- To return to the main page, click the Predict Again button under the prediction result.

Logging Out

- Click the LOGOUT button on the right side of the navigation bar on the main page.



Has patient ever been diagnosed with Cancer ?

Is patient currently experiencing any symptoms?

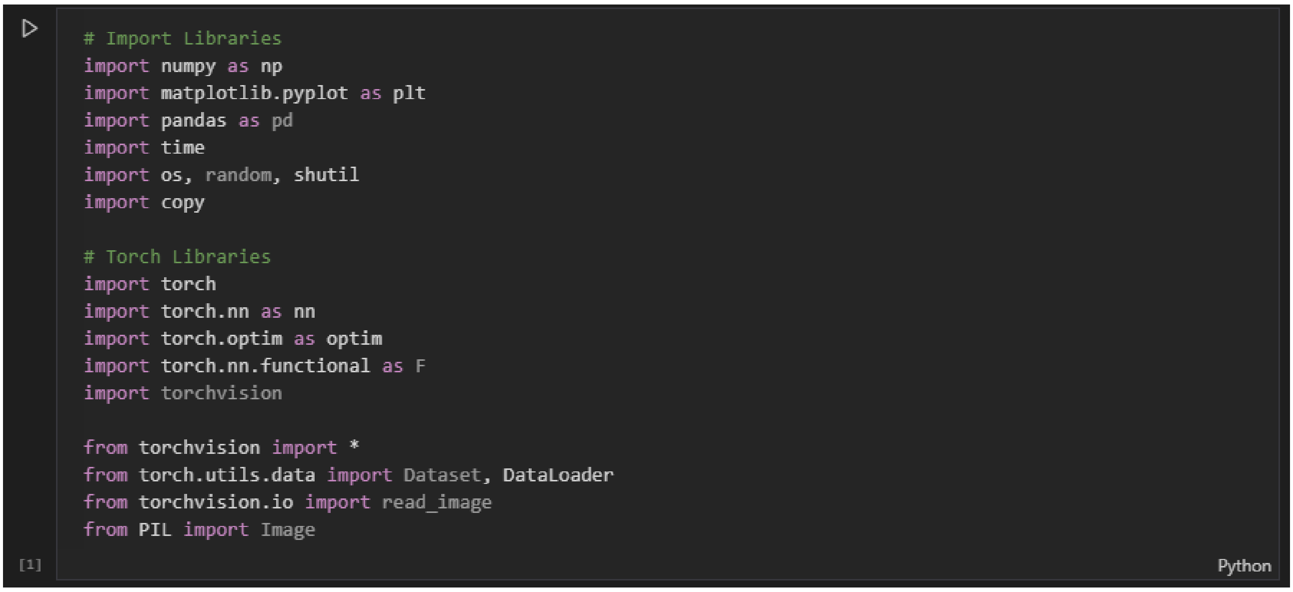
Is patient currently undergoing any treatment?

Upload Medical Image

Technical Guide

Softwares

1. Pull from Github Repository at https://github.com/elainealverina/FIT3164.git
2. The code is written in Python and in Jupyter Notebook, so preferably have applications that support these. For example, Visual Studio Code. Visual Studio Code can be downloaded from this link: https://code.visualstudio.com/Download
3. Install the necessary Python packages - torch, torchvision, numpy, matplotlib, pandas, flask, etc

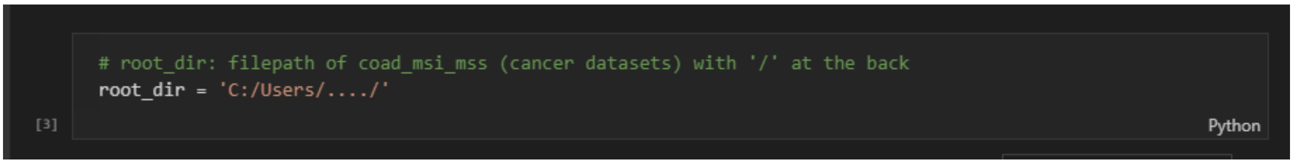


```python
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import time
import os, random, shutil
import copy

# Torch Libraries
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torchvision

from torchvision import *
from torch.utils.data import Dataset, DataLoader
from torchvision.io import read_image
from PIL import Image
```

Predictive Model Component

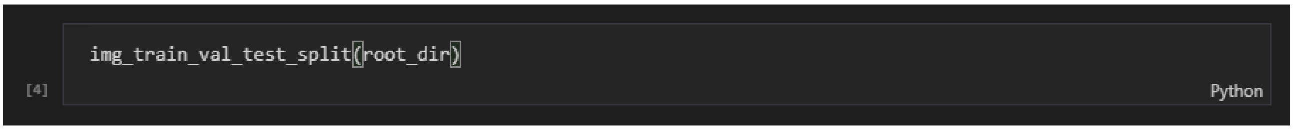
Dividing Dataset

- The Main Function is called **img_train_val_test_split(root_dir),** changing root_dir to the local directory where the full dataset is located.
- Remember to include a slash '/' at the end of file path.
- For example: **root_dir** = 'C:/Users/abc/…./'



```python
# root_dir: filepath of coad_msi_mss (cancer datasets) with '/' at the back
root_dir = 'C:/Users/..../'
```

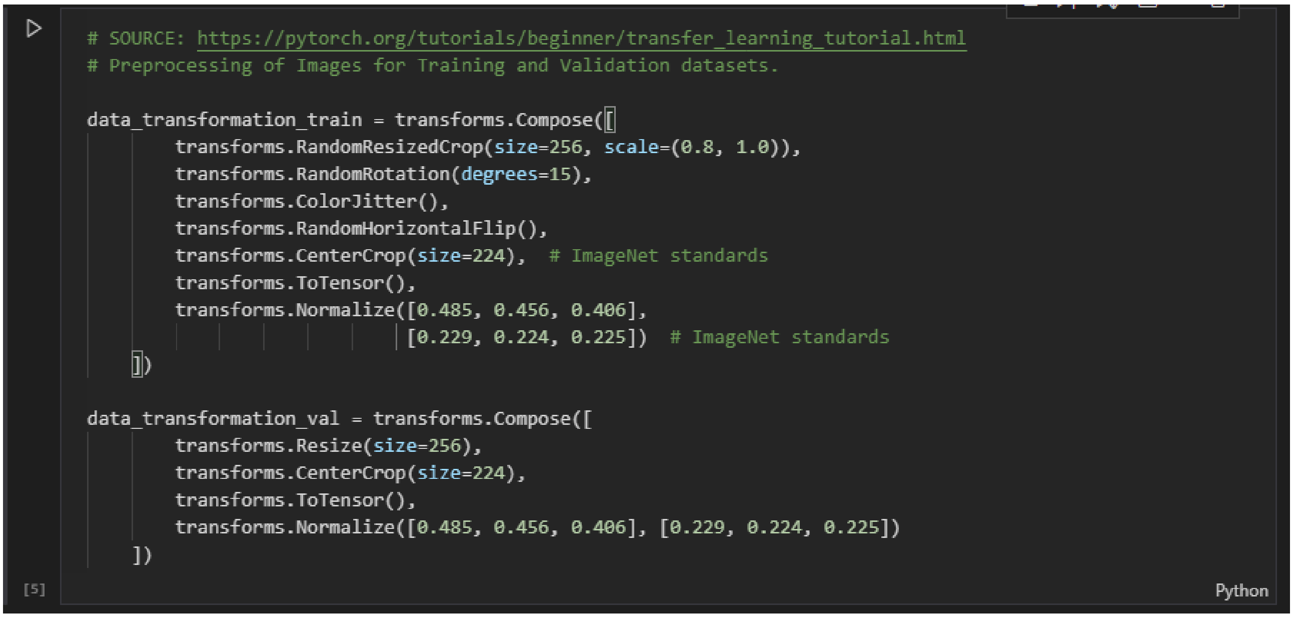
- Run the function to split images into training, validation and testing



```python
img_train_val_test_split(root_dir)
```

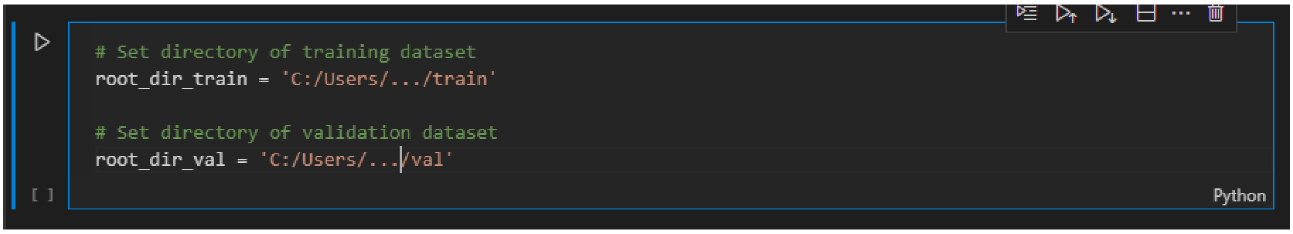
Preprocessing: Data Augmentation and Normalization

- To change the types of alterations done to the training and validation dataset, change the block of code located in **CELL NUMBER 5,** with the types of preprocessing.
- Transformations for Training dataset is on variable **data_transformation_train**
- Transformations for Validation dataset is on variable **data_transformation_val**



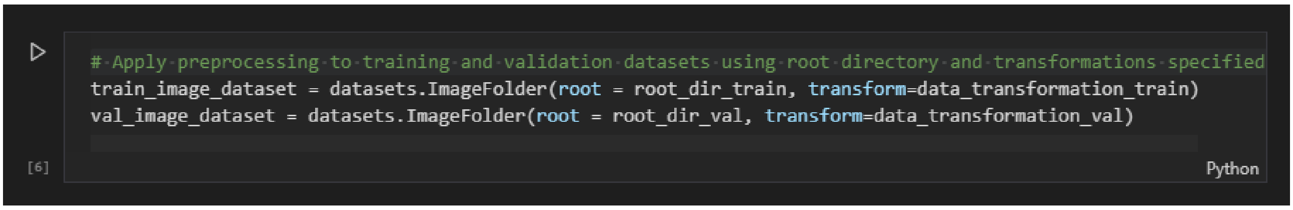
```python
# SOURCE: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
# Preprocessing of Images for Training and Validation datasets.

data_transformation_train = transforms.Compose([
        transforms.RandomResizedCrop(size=256, scale=(0.8, 1.0)),
        transforms.RandomRotation(degrees=15),
        transforms.ColorJitter(),
        transforms.RandomHorizontalFlip(),
        transforms.CenterCrop(size=224),  # ImageNet standards
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])  # ImageNet standards
    ])

data_transformation_val = transforms.Compose([
        transforms.Resize(size=256),
        transforms.CenterCrop(size=224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
```

- Set directory of training (on variable **root_dir_train**) and validation dataset (on variable **root_dir_val**) on **CELL NUMBER 6**



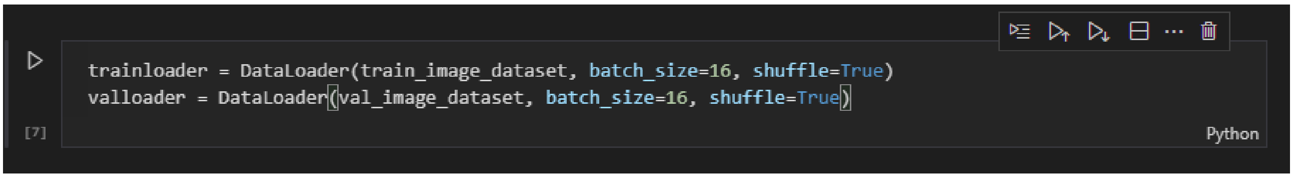
```python
# Set directory of training dataset
root_dir_train = 'C:/Users/.../train'

# Set directory of validation dataset
root_dir_val = 'C:/Users/.../val'
```

- Apply the preprocessing to training and validation datasets using root directory and transformations specified on **CELL NUMBER 7**



```python
# Apply preprocessing to training and validation datasets using root directory and transformations specified
train_image_dataset = datasets.ImageFolder(root = root_dir_train, transform=data_transformation_train)
val_image_dataset = datasets.ImageFolder(root = root_dir_val, transform=data_transformation_val)
```

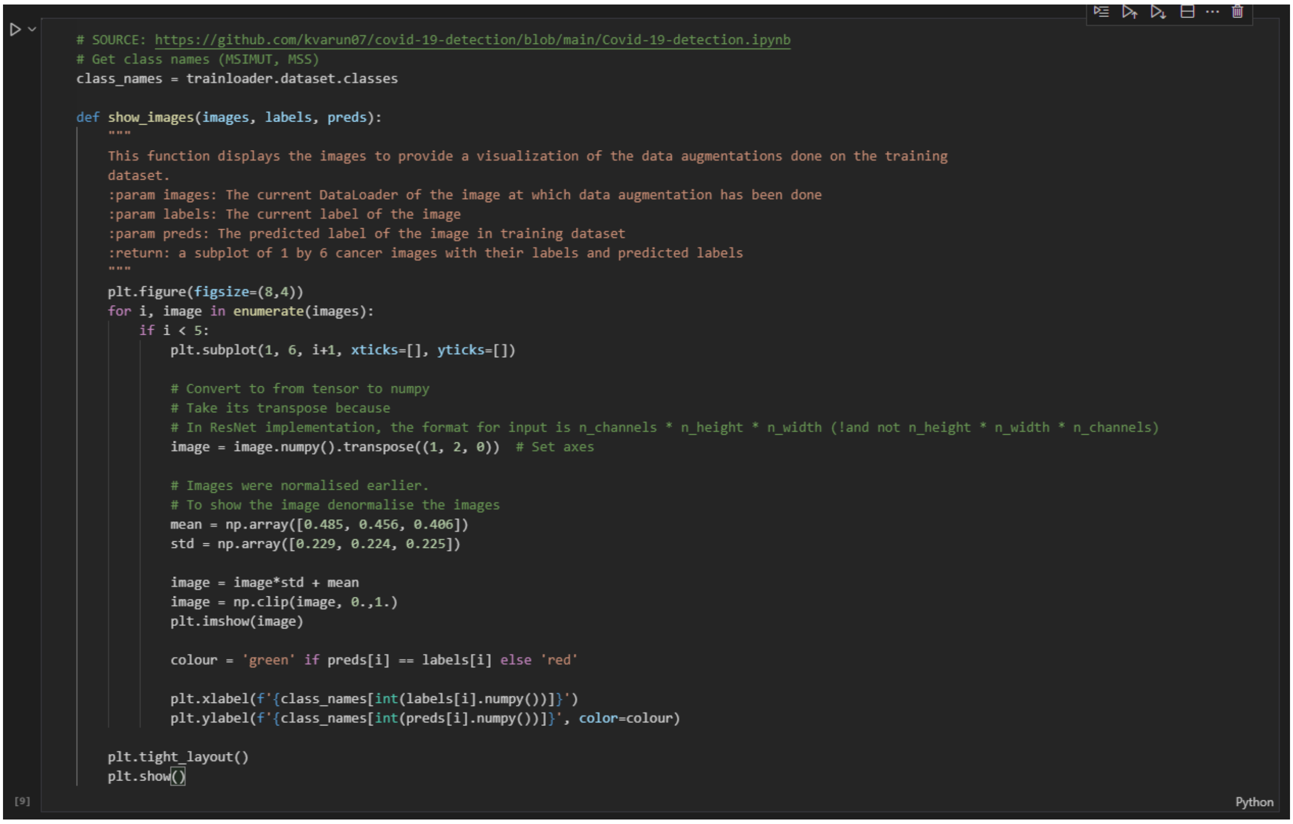
- Prepare DataLoader for training and validation datasets on **CELL NUMBER 8**
  - Change Batch Size according to specific requirements by the client, and set shuffle=False if do not want to shuffle images



```python
trainloader = DataLoader(train_image_dataset, batch_size=16, shuffle=True)
valloader = DataLoader(val_image_dataset, batch_size=16, shuffle=True)
```

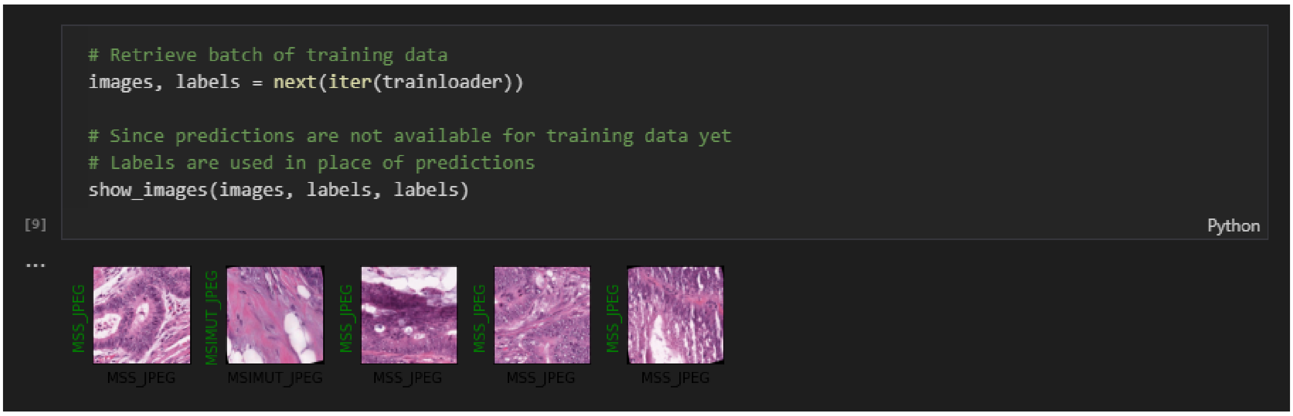
Data Visualization - Displaying Some Images

- The main function is located in **CELL NUMBER 9,** called **show_images.**



```python
# SOURCE: https://github.com/kvarun07/covid-19-detection/blob/main/Covid-19-detection.ipynb
# Get class names (MSIMUT, MSS)
class_names = trainloader.dataset.classes

def show_images(images, labels, preds):
    """
    This function displays the images to provide a visualization of the data augmentations done on the training
    dataset.
    :param images: The current DataLoader of the image at which data augmentation has been done
    :param labels: The current label of the image
    :param preds: The predicted label of the image in training dataset
    :return: a subplot of 1 by 6 cancer images with their labels and predicted labels
    """
    plt.figure(figsize=(8,4))
    for i, image in enumerate(images):
        if i < 5:
            plt.subplot(1, 6, i+1, xticks=[], yticks=[])

            # Convert to from tensor to numpy
            # Take its transpose because
            # In ResNet implementation, the format for input is n_channels * n_height * n_width (!and not n_height * n_width * n_channels)
            image = image.numpy().transpose((1, 2, 0))  # Set axes

            # Images were normalised earlier.
            # To show the image denormalise the images
            mean = np.array([0.485, 0.456, 0.406])
            std = np.array([0.229, 0.224, 0.225])

            image = image*std + mean
            image = np.clip(image, 0.,1.)
            plt.imshow(image)

            colour = 'green' if preds[i] == labels[i] else 'red'

            plt.xlabel(f'{class_names[int(labels[i].numpy())]}')
            plt.ylabel(f'{class_names[int(preds[i].numpy())]}', color=colour)

    plt.tight_layout()
    plt.show()
```

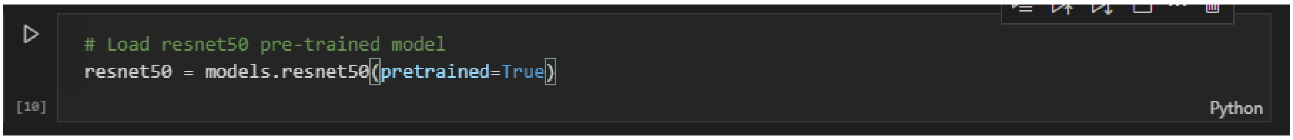
- After running **CELL NUMBER 9**, run **CELL NUMBER 10** to display the images. The images would be displayed in a 1 by 5 subplot shown below



```python
# Retrieve batch of training data
images, labels = next(iter(trainloader))

# Since predictions are not available for training data yet
# Labels are used in place of predictions
show_images(images, labels, labels)
```

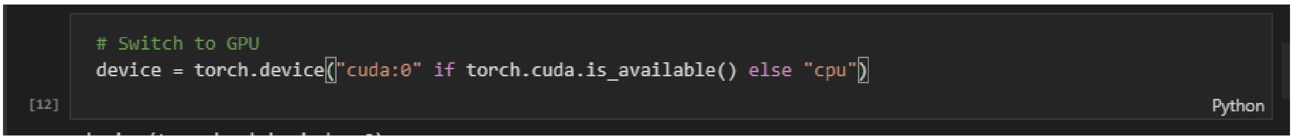
Loading the Model

- Here, we are using pre-trained Convolutional Neural Networks using the package resnet50. If client would like to change the type of package, the list of available packages supported by PyTorch can be found in this link:
- This can be found in **CELL NUMBER 11**



```python
# Load resnet50 pre-trained model
resnet50 = models.resnet50(pretrained=True)
```

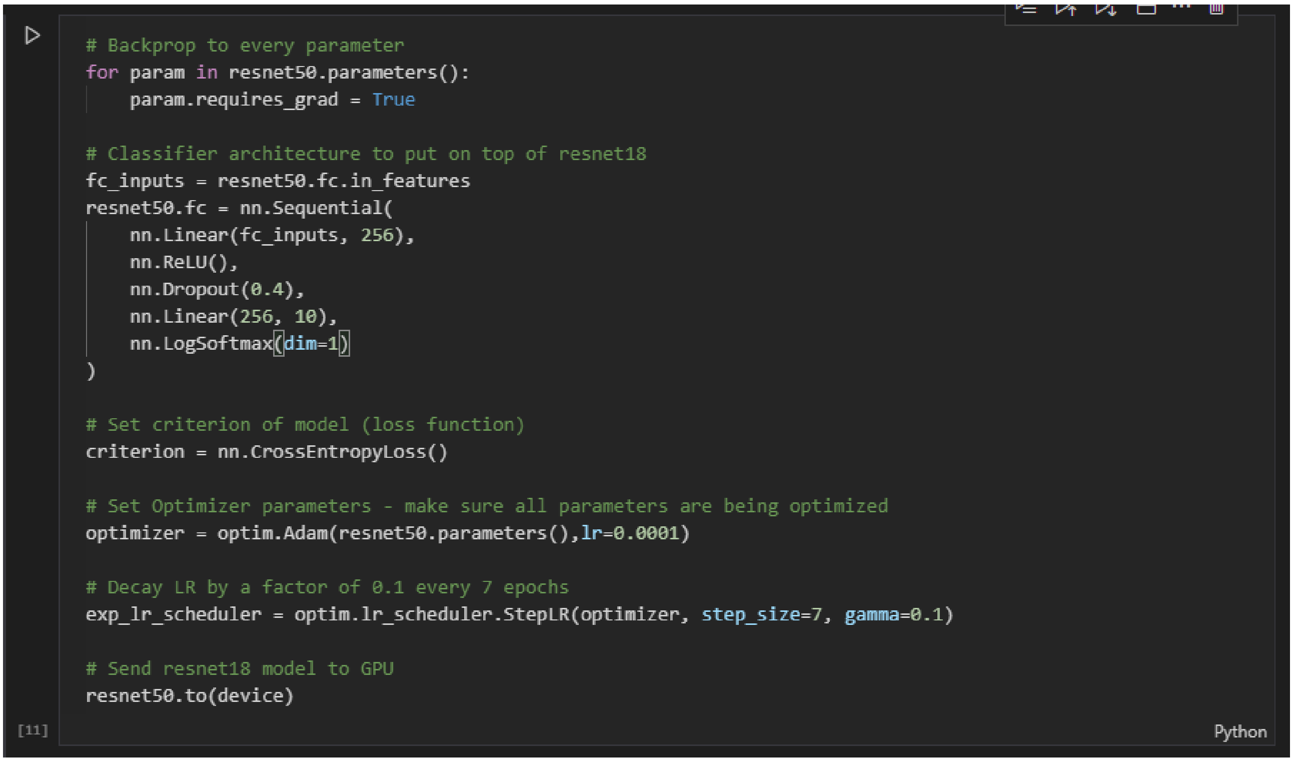
- GPU Usage. If client's current device supports GPU Usage by Pytorch, can run **CELL NUMBER 12** to switch training to GPU instead of CPU



```python
# Switch to GPU
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

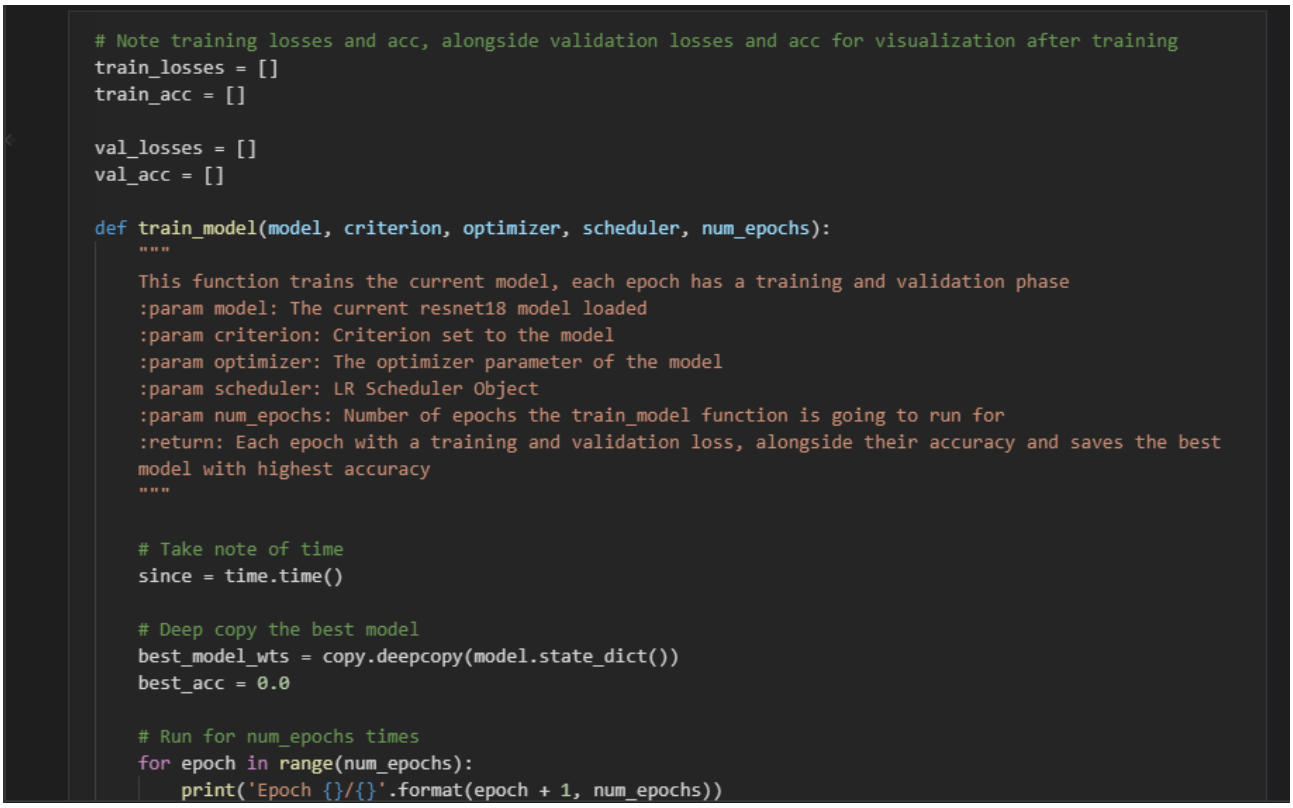
Training Model - Set Required Parameters

- The current trained model in the Jupyter Notebook is set to do no feature extraction i.e - It goes through every layer in the ConvNet. If User would like to do feature extraction, set param.requires_grad = False in **CELL NUMBER 13**
- To change the current classifier architecture, it is located in the variable called resnet50.fc
- The current loss function is using CrossEntropyLoss(). To set and change the loss function, modify criterion variable with the required loss function
- The current program uses Adam Optimizer. To change the optimizer and its parameters, modify the optimizer variable
- The scheduler function is also included in this program. It's located in the exp_lr_scheduler variable. Currently, the LR Scheduler is being decayed at a factor of 0.1 every 7 epochs
- Lastly, send the model to GPU



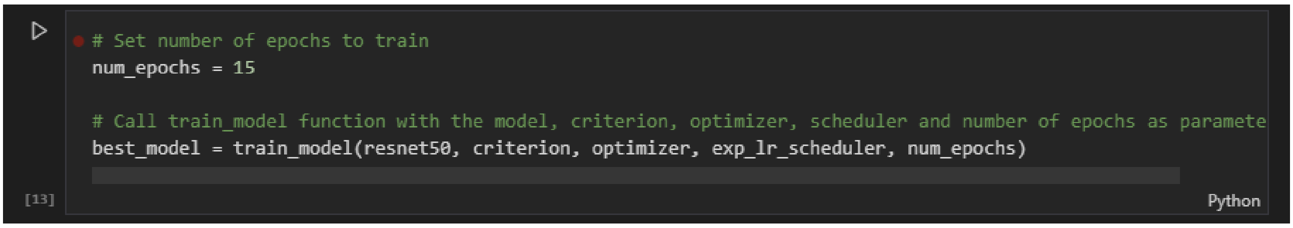
```python
# Backprop to every parameter
for param in resnet50.parameters():
    param.requires_grad = True

# Classifier architecture to put on top of resnet18
fc_inputs = resnet50.fc.in_features
resnet50.fc = nn.Sequential(
    nn.Linear(fc_inputs, 256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, 10),
    nn.LogSoftmax(dim=1)
)

# Set criterion of model (loss function)
criterion = nn.CrossEntropyLoss()

# Set Optimizer parameters - make sure all parameters are being optimized
optimizer = optim.Adam(resnet50.parameters(),lr=0.0001)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)

# Send resnet18 model to GPU
resnet50.to(device)
```

- The function to train the model is in a function called train_model, with required parameters of the model, criterion, optimizer, scheduler and the number of epochs. The function is located in **CELL NUMBER 14**, and this cell also notes the lists of training losses and acc alongside validation losses and acc



```
# Note training losses and acc, alongside validation losses and acc for visualization after training
train_losses = []
train_acc = []

val_losses = []
val_acc = []

def train_model(model, criterion, optimizer, scheduler, num_epochs):
    """
    This function trains the current model, each epoch has a training and validation phase
    :param model: The current resnet18 model loaded
    :param criterion: Criterion set to the model
    :param optimizer: The optimizer parameter of the model
    :param scheduler: LR Scheduler Object
    :param num_epochs: Number of epochs the train_model function is going to run for
    :return: Each epoch with a training and validation loss, alongside their accuracy and saves the best
    model with highest accuracy
    """

    # Take note of time
    since = time.time()

    # Deep copy the best model
    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    # Run for num_epochs times
    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch + 1, num_epochs))
```

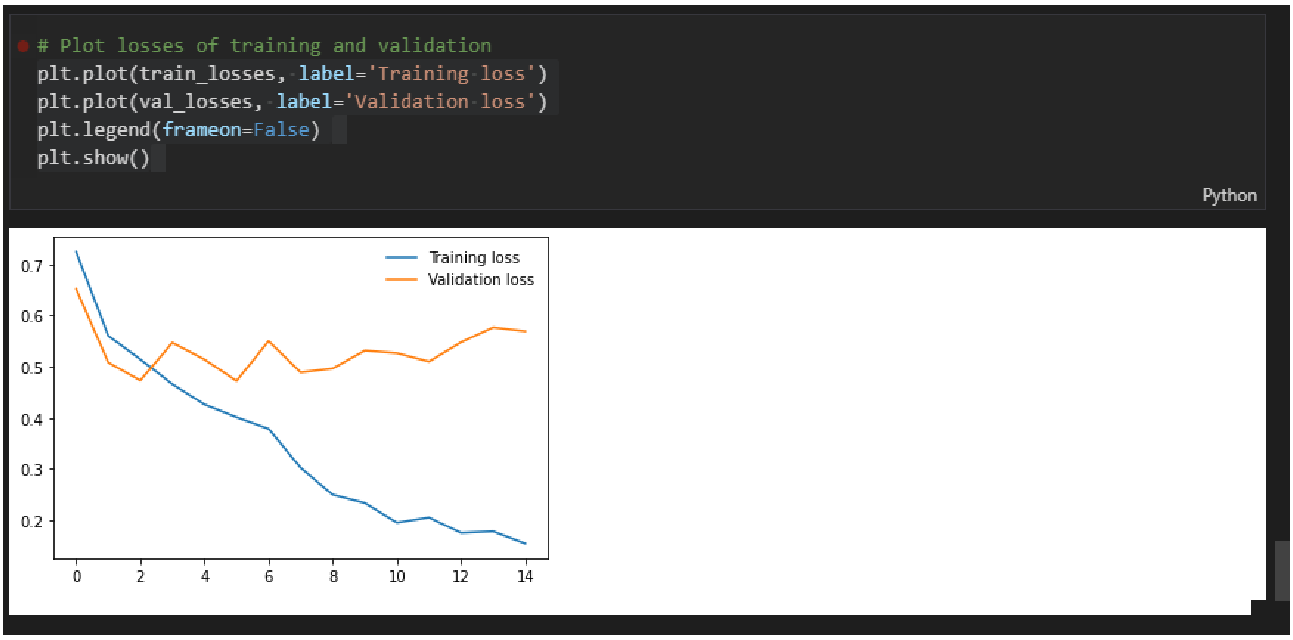
- To train the model, call the train_model function alongside with the number of epochs to train the model for in the num_epochs variable. An example can be seen below.



```
# Set number of epochs to train
num_epochs = 15

# Call train_model function with the model, criterion, optimizer, scheduler and number of epochs as paramete
best_model = train_model(resnet50, criterion, optimizer, exp_lr_scheduler, num_epochs)
```

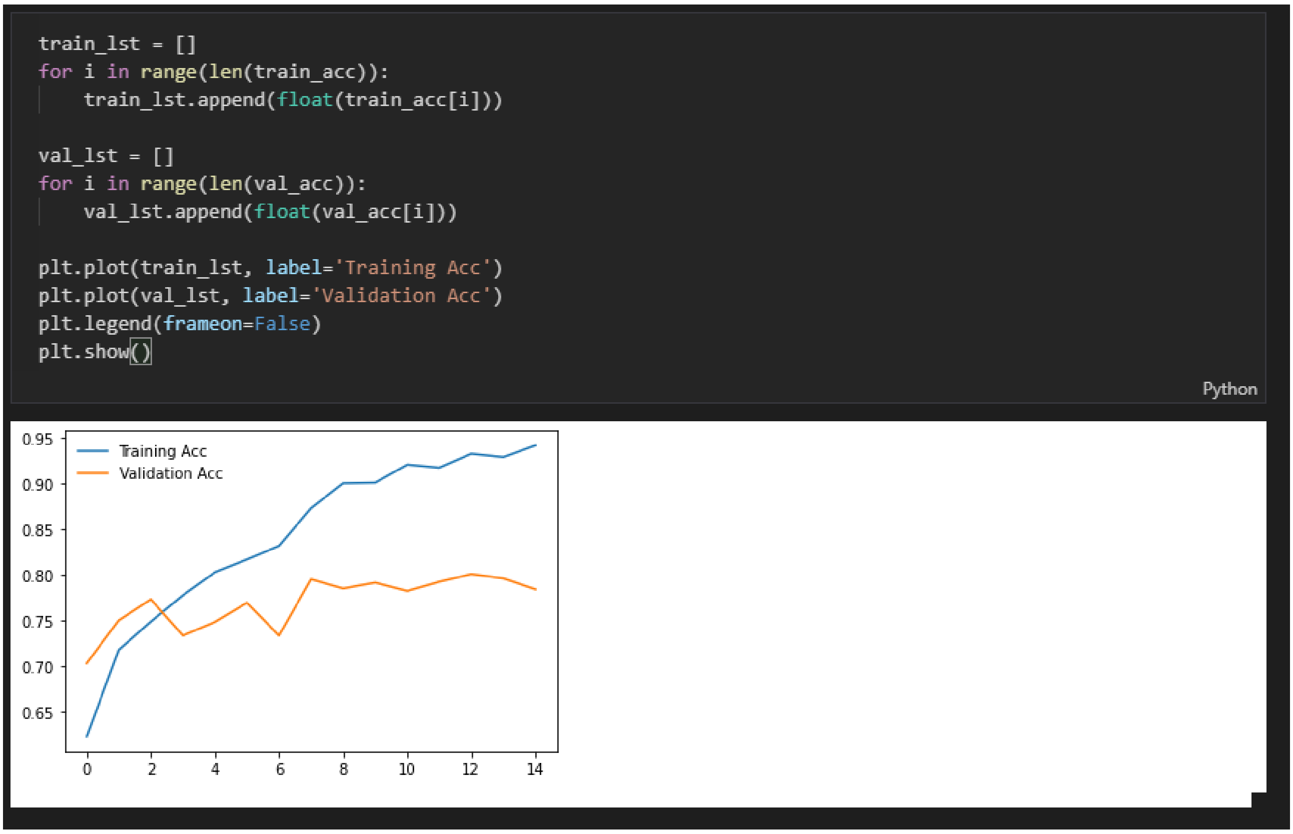
Plotting Results from Trained Model

- Run **CELL NUMBER 16** to plot the training and validation losses



```
# Plot losses of training and validation
plt.plot(train_losses, label='Training loss')
plt.plot(val_losses, label='Validation loss')
plt.legend(frameon=False)
plt.show()
```

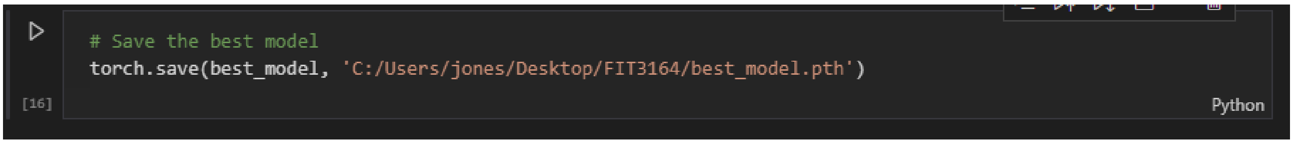
- Run **CELL NUMBER 17** to plot the training and validation accuracies



```
train_lst = []
for i in range(len(train_acc)):
    train_lst.append(float(train_acc[i]))

val_lst = []
for i in range(len(val_acc)):
    val_lst.append(float(val_acc[i]))

plt.plot(train_lst, label='Training Acc')
plt.plot(val_lst, label='Validation Acc')
plt.legend(frameon=False)
plt.show()
```

Save Model

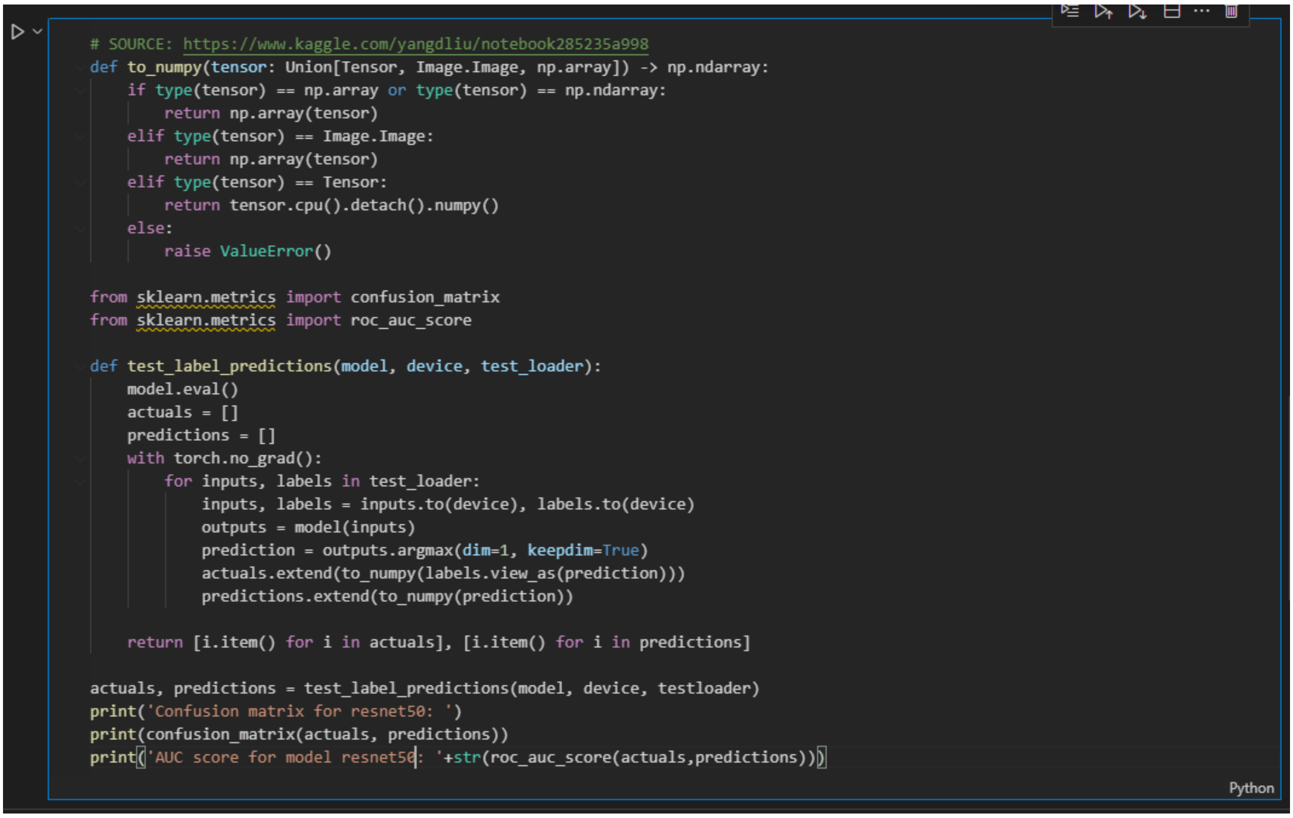
- To save the best model given by the training function above, Run **CELL NUMBER 18**. It saves the best model in .pth format. Remember to set the specified directory to where user want to save the model



```
# Save the best model
torch.save(best_model, 'C:/Users/jones/Desktop/FIT3164/best_model.pth')
```

Inference Notebook

- To plot Confusion Matrix and AUC, Load the best model saved earlier & Run code block shown below.

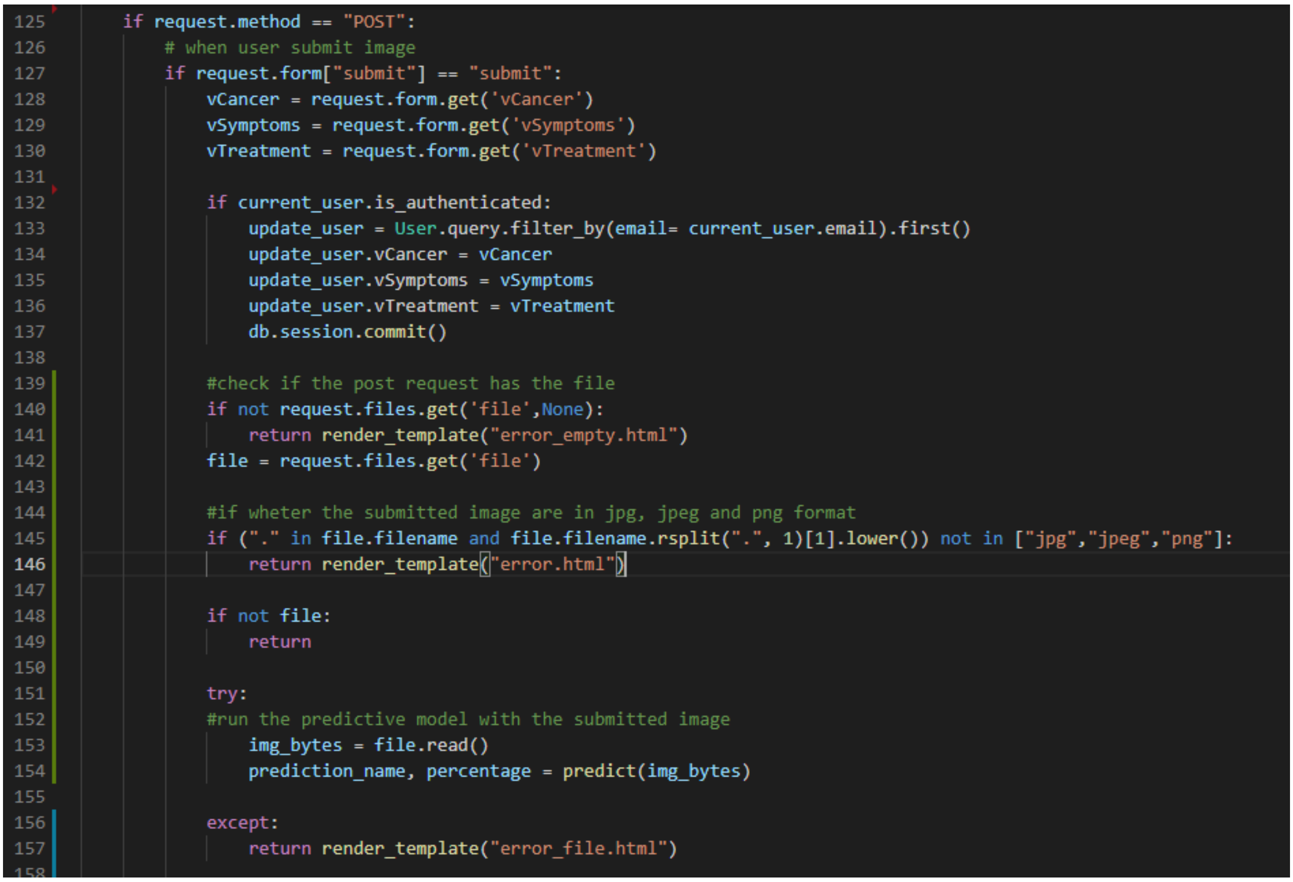


```python
# SOURCE: https://www.kaggle.com/yangdliu/notebook285235a998
def to_numpy(tensor: Union[Tensor, Image.Image, np.array]) -> np.ndarray:
    if type(tensor) == np.array or type(tensor) == np.ndarray:
        return np.array(tensor)
    elif type(tensor) == Image.Image:
        return np.array(tensor)
    elif type(tensor) == Tensor:
        return tensor.cpu().detach().numpy()
    else:
        raise ValueError()

from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score

def test_label_predictions(model, device, test_loader):
    model.eval()
    actuals = []
    predictions = []
    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            prediction = outputs.argmax(dim=1, keepdim=True)
            actuals.extend(to_numpy(labels.view_as(prediction)))
            predictions.extend(to_numpy(prediction))

    return [i.item() for i in actuals], [i.item() for i in predictions]

actuals, predictions = test_label_predictions(model, device, testloader)
print('Confusion matrix for resnet50: ')
print(confusion_matrix(actuals, predictions))
print('AUC score for model resnet50: '+str(roc_auc_score(actuals,predictions)))
```

Website

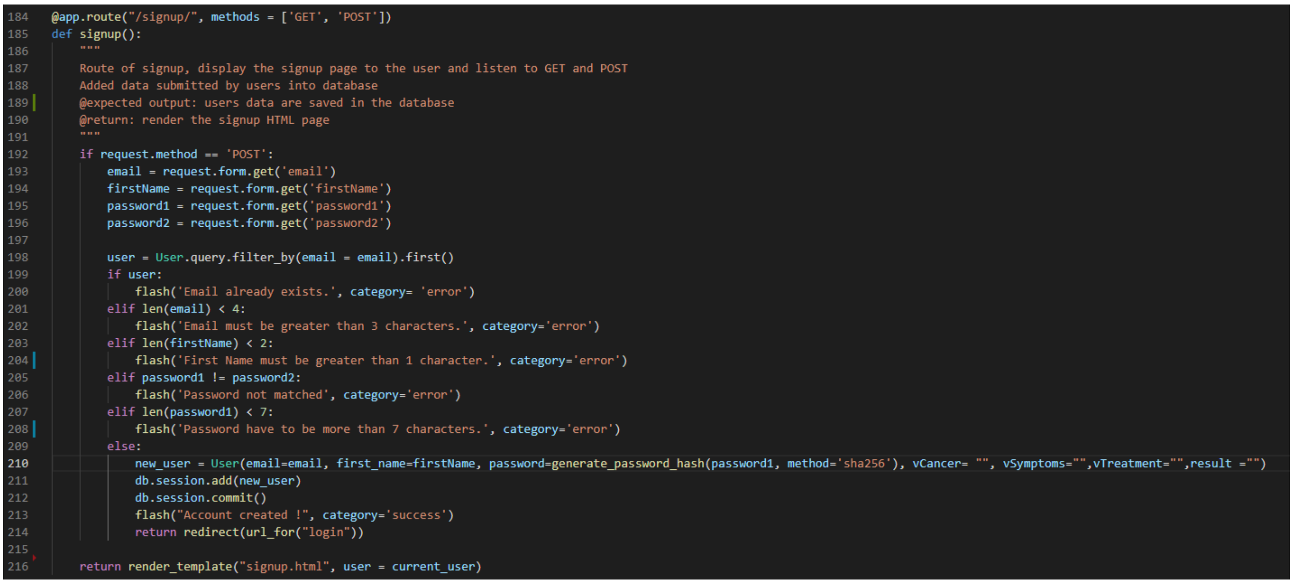
main.py

- To change the rendered HTML file, simply make changes on render_template("yourfile.html").
- To change the accepted users submitted file extension or type, make changes on **Line 145**.
- The software gets the responses from the questions (**Line 128 to 130**) and saves them in the database (**Line 133 to 137**). Then the image is process with the predictive model (**Line 151 to 157**)



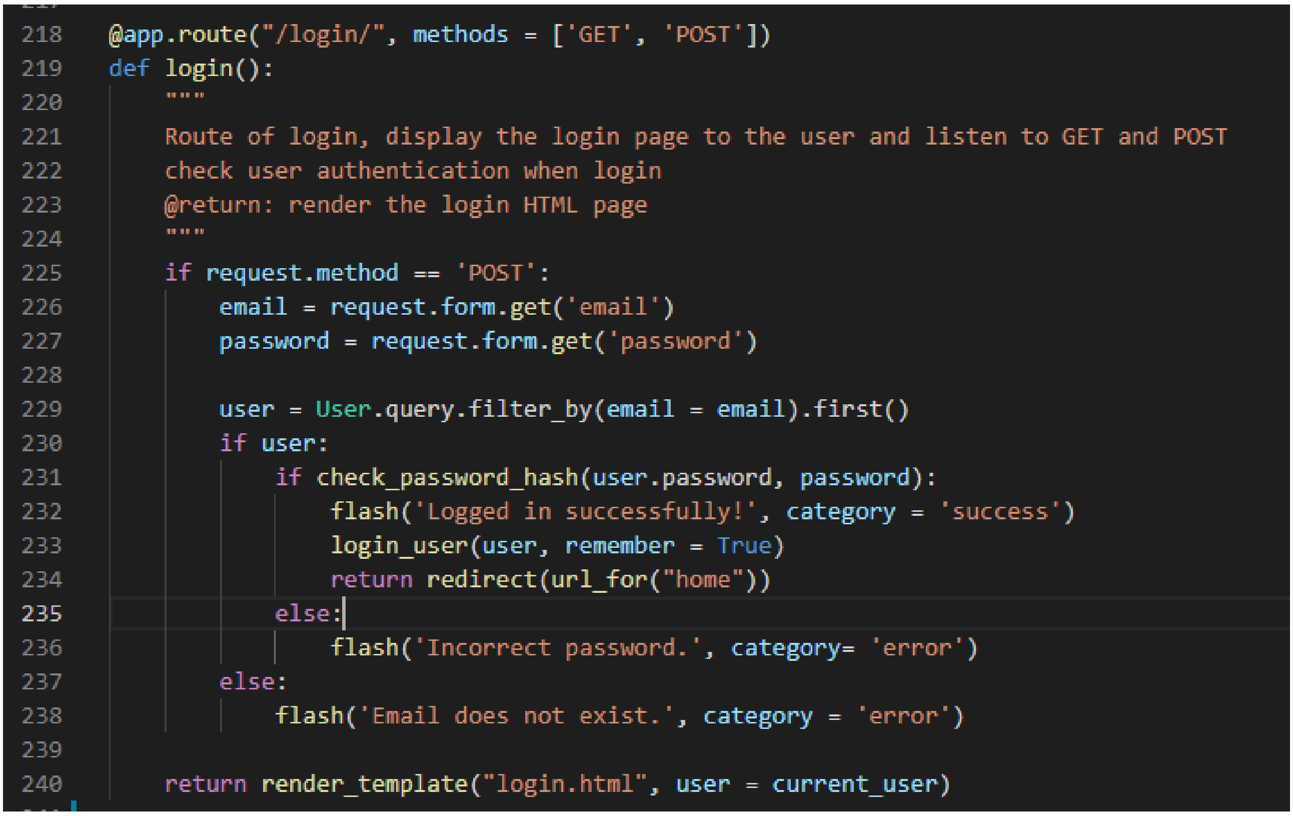
```
125     if request.method == "POST":
126         # when user submit image
127         if request.form["submit"] == "submit":
128             vCancer = request.form.get('vCancer')
129             vSymptoms = request.form.get('vSymptoms')
130             vTreatment = request.form.get('vTreatment')
131 
132             if current_user.is_authenticated:
133                 update_user = User.query.filter_by(email= current_user.email).first()
134                 update_user.vCancer = vCancer
135                 update_user.vSymptoms = vSymptoms
136                 update_user.vTreatment = vTreatment
137                 db.session.commit()
138 
139             #check if the post request has the file
140             if not request.files.get('file',None):
141                 return render_template("error_empty.html")
142             file = request.files.get('file')
143 
144             #if wheter the submitted image are in jpg, jpeg and png format
145             if ("." in file.filename and file.filename.rsplit(".", 1)[1].lower()) not in ["jpg","jpeg","png"]:
146                 return render_template("error.html")
147 
148             if not file:
149                 return
150 
151             try:
152             #run the predictive model with the submitted image
153                 img_bytes = file.read()
154                 prediction_name, percentage = predict(img_bytes)
155 
156             except:
157                 return render_template("error_file.html")
158 
```

- The software gets the signup information from users (**Line 193 to 196**) and saves them in the database (**Line 210 to 212**). To change the feedback message, make changes on flash("Message").



```
184  @app.route("/signup/", methods = ['GET', 'POST'])
185  def signup():
186      """
187      Route of signup, display the signup page to the user and listen to GET and POST
188      Added data submitted by users into database
189      @expected output: users data are saved in the database
190      @return: render the signup HTML page
191      """
192      if request.method == 'POST':
193          email = request.form.get('email')
194          firstName = request.form.get('firstName')
195          password1 = request.form.get('password1')
196          password2 = request.form.get('password2')
197
198          user = User.query.filter_by(email = email).first()
199          if user:
200              flash('Email already exists.', category= 'error')
201          elif len(email) < 4:
202              flash('Email must be greater than 3 characters.', category='error')
203          elif len(firstName) < 2:
204              flash('First Name must be greater than 1 character.', category='error')
205          elif password1 != password2:
206              flash('Password not matched', category='error')
207          elif len(password1) < 7:
208              flash('Password have to be more than 7 characters.', category='error')
209          else:
210              new_user = User(email=email, first_name=firstName, password=generate_password_hash(password1, method='sha256'), vCancer= "", vSymptoms="",vTreatment="",result ="")
211              db.session.add(new_user)
212              db.session.commit()
213              flash("Account created !", category='success')
214              return redirect(url_for("login"))
215
216      return render_template("signup.html", user = current_user)
```

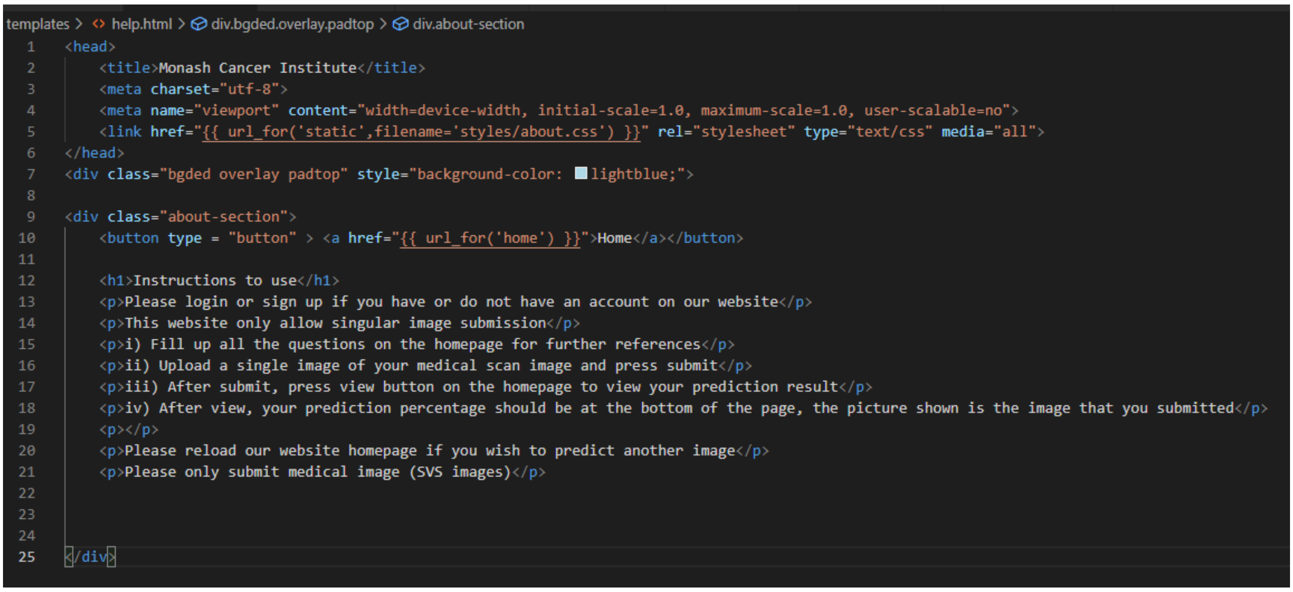
- The software checks the users' password by check_password_hash (**Line 231**) and remembers the session of the user by login_user (**Line 233**).



```
218  @app.route("/login/", methods = ['GET', 'POST'])
219  def login():
220      """
221      Route of login, display the login page to the user and listen to GET and POST
222      check user authentication when login
223      @return: render the login HTML page
224      """
225      if request.method == 'POST':
226          email = request.form.get('email')
227          password = request.form.get('password')
228
229          user = User.query.filter_by(email = email).first()
230          if user:
231              if check_password_hash(user.password, password):
232                  flash('Logged in successfully!', category = 'success')
233                  login_user(user, remember = True)
234                  return redirect(url_for("home"))
235              else:
236                  flash('Incorrect password.', category= 'error')
237          else:
238              flash('Email does not exist.', category = 'error')
239
240      return render_template("login.html", user = current_user)
```

help.html

- To change the stylesheet of the help page, change the reference link on **line 5**. To change the content of the help page, make changes from **line 12 to line 21**. To change the button navigation destination, make changes on **line 10**.

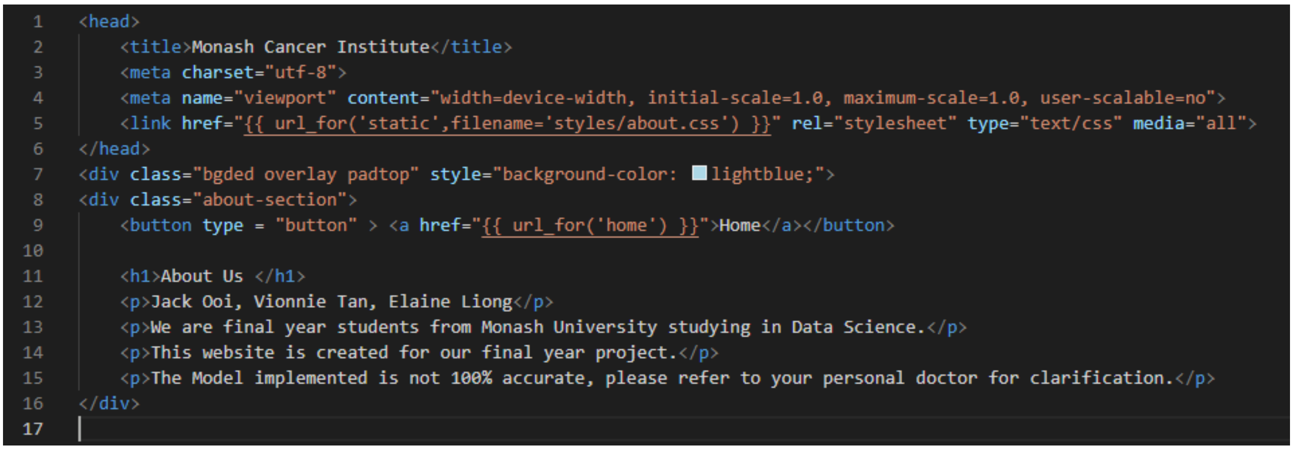


templates > help.html > div.bgded.overlay.padtop > div.about-section

```
1   <head>
2       <title>Monash Cancer Institute</title>
3       <meta charset="utf-8">
4       <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
5       <link href="{{ url_for('static',filename='styles/about.css') }}" rel="stylesheet" type="text/css" media="all">
6   </head>
7   <div class="bgded overlay padtop" style="background-color: lightblue;">
8
9   <div class="about-section">
10      <button type = "button" > <a href="{{ url_for('home') }}">Home</a></button>
11
12      <h1>Instructions to use</h1>
13      <p>Please login or sign up if you have or do not have an account on our website</p>
14      <p>This website only allow singular image submission</p>
15      <p>i) Fill up all the questions on the homepage for further references</p>
16      <p>ii) Upload a single image of your medical scan image and press submit</p>
17      <p>iii) After submit, press view button on the homepage to view your prediction result</p>
18      <p>iv) After view, your prediction percentage should be at the bottom of the page, the picture shown is the image that you submitted</p>
19      <p></p>
20      <p>Please reload our website homepage if you wish to predict another image</p>
21      <p>Please only submit medical image (SVS images)</p>
22
23
24
25  </div>
```

about.html

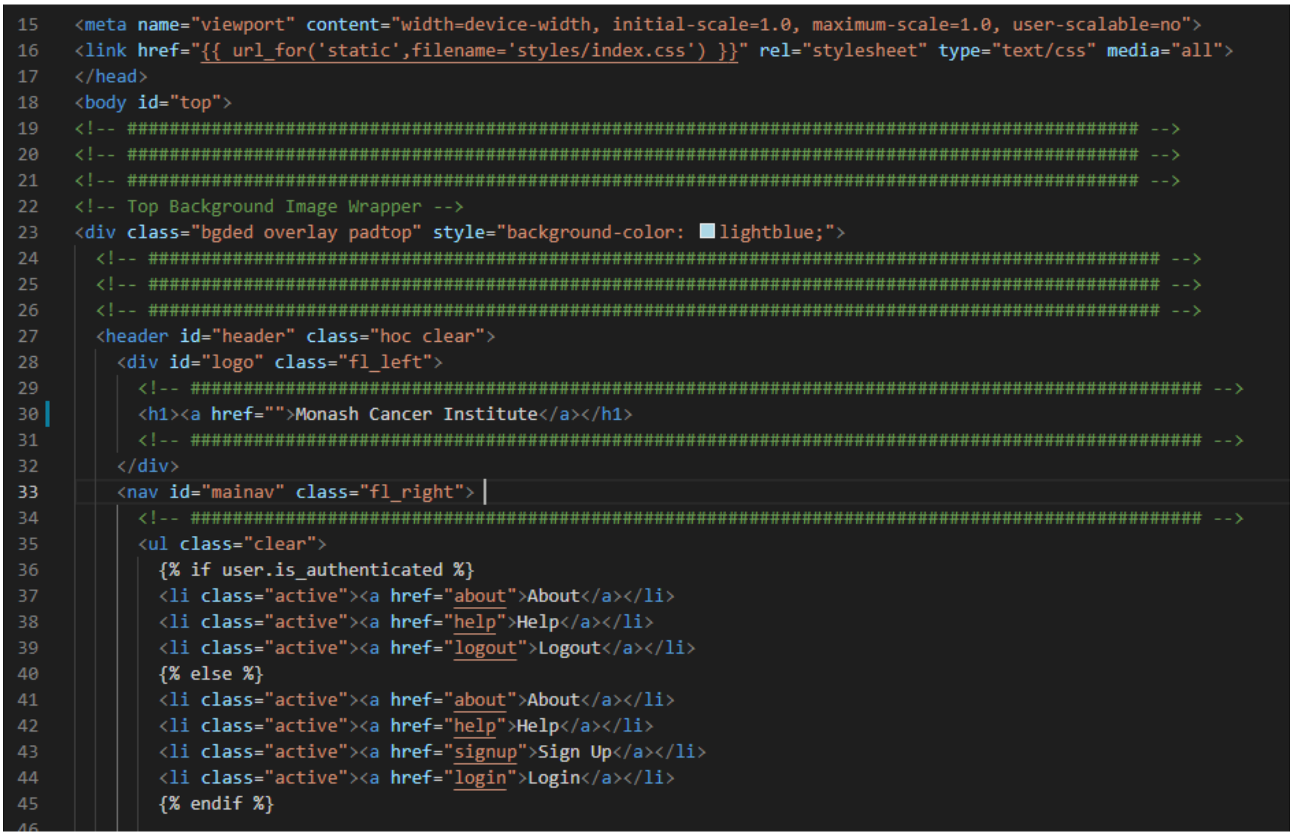
- To change the stylesheet of the about page, change the reference link on **line 5**. To change the content of the about page, make changes from **line 11 to 15**. To change the button navigation destination, make changes on **line 9**.



```
1   <head>
2       <title>Monash Cancer Institute</title>
3       <meta charset="utf-8">
4       <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
5       <link href="{{ url_for('static',filename='styles/about.css') }}" rel="stylesheet" type="text/css" media="all">
6   </head>
7   <div class="bgded overlay padtop" style="background-color: lightblue;">
8   <div class="about-section">
9       <button type = "button" > <a href="{{ url_for('home') }}">Home</a></button>
10
11      <h1>About Us </h1>
12      <p>Jack Ooi, Vionnie Tan, Elaine Liong</p>
13      <p>We are final year students from Monash University studying in Data Science.</p>
14      <p>This website is created for our final year project.</p>
15      <p>The Model implemented is not 100% accurate, please refer to your personal doctor for clarification.</p>
16  </div>
17
```

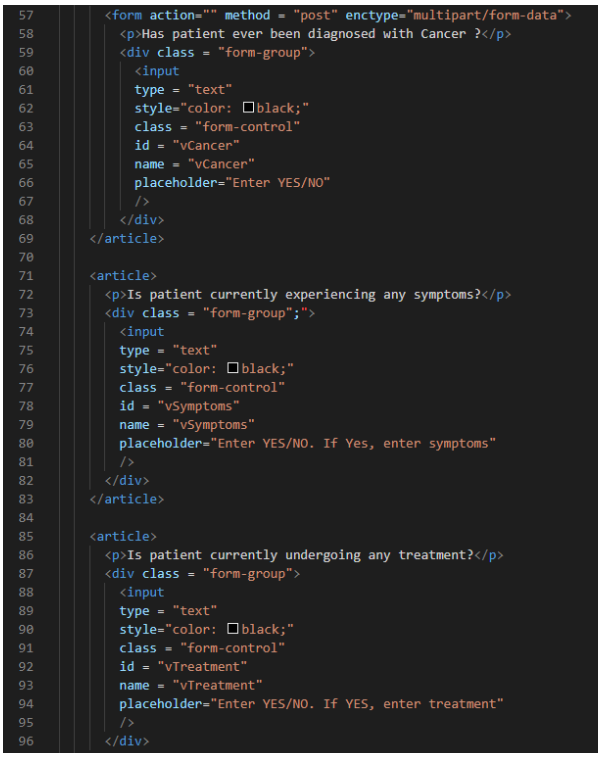
index.html

- To change the stylesheet of the index page, change the reference link on **line 16**. To change the background colour of the homepage, change **line 23**. **Line 36 to 45**, show the button when the user is logged in while **line 41 and 44** are when users are not logged in.



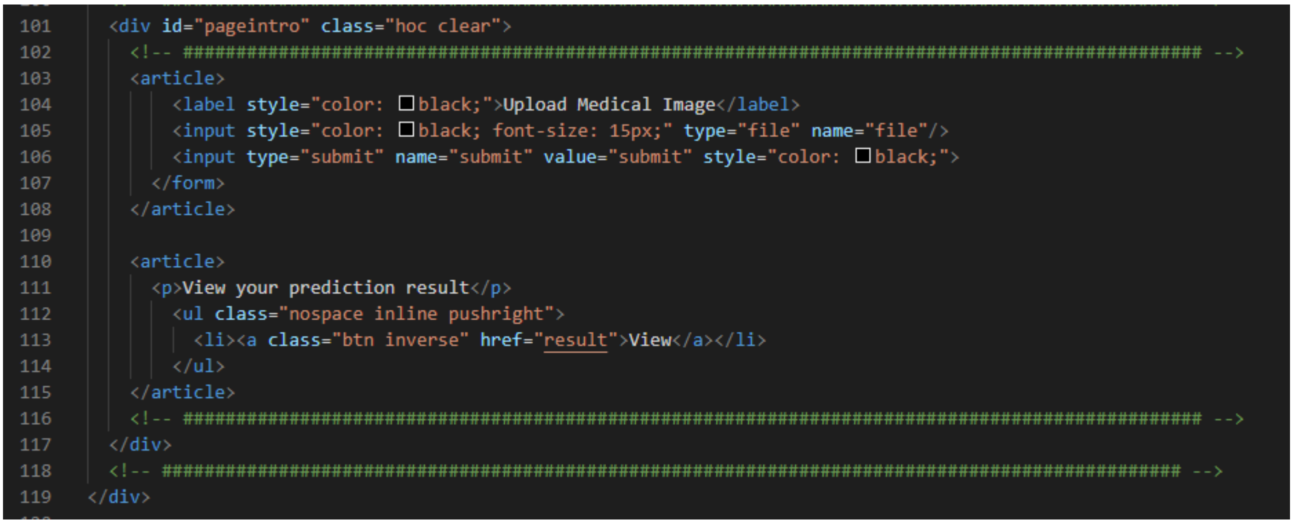
```
15  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
16  <link href="{{ url_for('static',filename='styles/index.css') }}" rel="stylesheet" type="text/css" media="all">
17  </head>
18  <body id="top">
19  <!-- ################################################################################################ -->
20  <!-- ################################################################################################ -->
21  <!-- ################################################################################################ -->
22  <!-- Top Background Image Wrapper -->
23  <div class="bgded overlay padtop" style="background-color: □lightblue;">
24    <!-- ################################################################################################ -->
25    <!-- ################################################################################################ -->
26    <!-- ################################################################################################ -->
27    <header id="header" class="hoc clear">
28      <div id="logo" class="fl_left">
29        <!-- ################################################################################################ -->
30        <h1><a href="">Monash Cancer Institute</a></h1>
31        <!-- ################################################################################################ -->
32      </div>
33      <nav id="mainav" class="fl_right">
34        <!-- ################################################################################################ -->
35        <ul class="clear">
36          {% if user.is_authenticated %}
37          <li class="active"><a href="about">About</a></li>
38          <li class="active"><a href="help">Help</a></li>
39          <li class="active"><a href="logout">Logout</a></li>
40          {% else %}
41          <li class="active"><a href="about">About</a></li>
42          <li class="active"><a href="help">Help</a></li>
43          <li class="active"><a href="signup">Sign Up</a></li>
44          <li class="active"><a href="login">Login</a></li>
45          {% endif %}
46
```

- **Line 57 to Line 94** show the questions input form.



```
57        <form action="" method = "post" enctype="multipart/form-data">
58          <p>Has patient ever been diagnosed with Cancer ?</p>
59          <div class = "form-group">
60            <input
61            type = "text"
62            style="color: ■black;"
63            class = "form-control"
64            id = "vCancer"
65            name = "vCancer"
66            placeholder="Enter YES/NO"
67            />
68          </div>
69      </article>
70
71      <article>
72        <p>Is patient currently experiencing any symptoms?</p>
73        <div class = "form-group";">
74          <input
75          type = "text"
76          style="color: ■black;"
77          class = "form-control"
78          id = "vSymptoms"
79          name = "vSymptoms"
80          placeholder="Enter YES/NO. If Yes, enter symptoms"
81          />
82        </div>
83      </article>
84
85      <article>
86        <p>Is patient currently undergoing any treatment?</p>
87        <div class = "form-group">
88          <input
89          type = "text"
90          style="color: ■black;"
91          class = "form-control"
92          id = "vTreatment"
93          name = "vTreatment"
94          placeholder="Enter YES/NO. If YES, enter treatment"
95          />
96        </div>
```

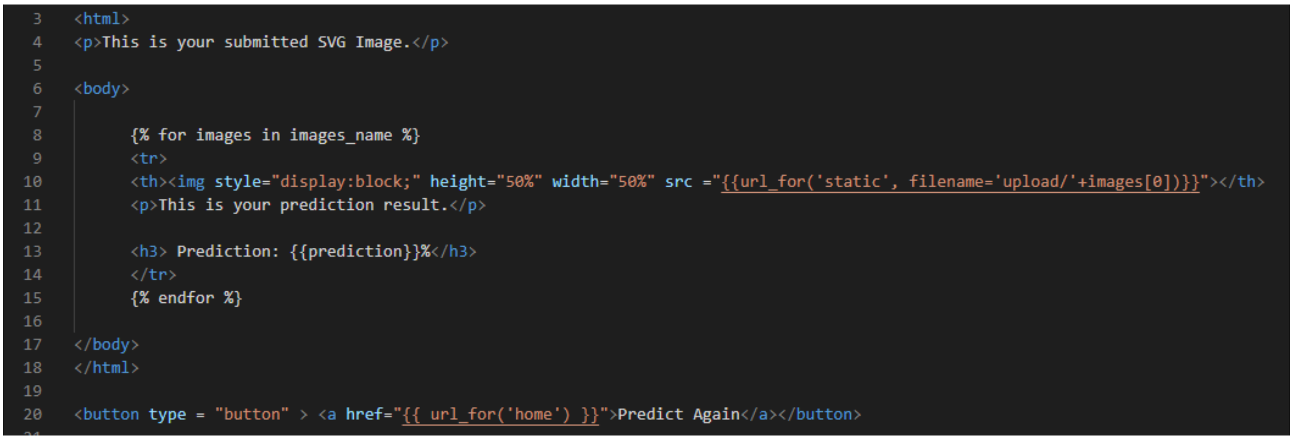
- **Line 101 to Line 108** indicate the form of image submission. **Line 111 to Line 113** is a button to let the user navigate to the result.html file.



```
    <div id="pageintro" class="hoc clear">
      <!-- ################################################################################################ -->
      <article>
          <label style="color: □black;">Upload Medical Image</label>
          <input style="color: □black; font-size: 15px;" type="file" name="file"/>
          <input type="submit" name="submit" value="submit" style="color: □black;">
        </form>
      </article>

      <article>
        <p>View your prediction result</p>
          <ul class="nospace inline pushright">
            <li><a class="btn inverse" href="result">View</a></li>
          </ul>
      </article>
      <!-- ################################################################################################ -->
    </div>
    <!-- ################################################################################################ -->
  </div>
```

result.html

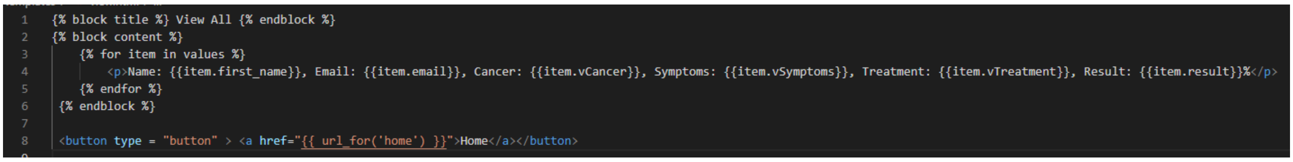
- **Line 8 to 10** show the user submitted image, while **line 11 to line 13** shows the result of our prediction model. For **line 10**, the url_for can be changed to display any image file in any directory. **Line 20** created a button to navigate users back to the homepage.



```
3   <html>
4   <p>This is your submitted SVG Image.</p>
5
6   <body>
7
8         {% for images in images_name %}
9         <tr>
10        <th><img style="display:block;" height="50%" width="50%" src ="{{url_for('static', filename='upload/'+images[0])}}"></th>
11        <p>This is your prediction result.</p>
12
13        <h3> Prediction: {{prediction}}%</h3>
14        </tr>
15        {% endfor %}
16
17  </body>
18  </html>
19
20  <button type = "button" > <a href="{{ url_for('home') }}">Predict Again</a></button>
```

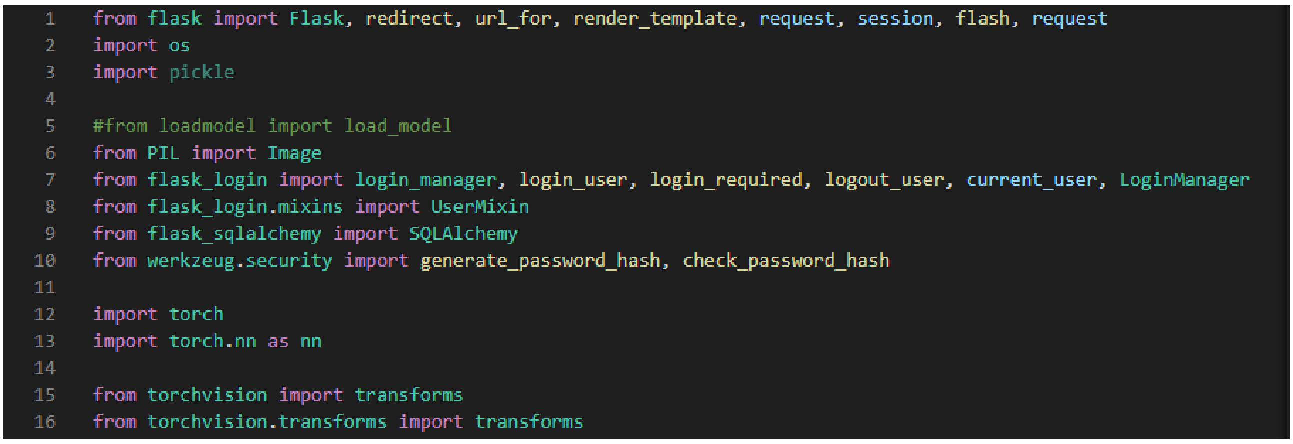
view.html

- **Line 1 to Line 5** show the data inside the database. **Line 8** created a button to navigate users back to the homepage. To change what is shown in view.html, simply make changes on **line 4**.



```
1  {% block title %} View All {% endblock %}
2  {% block content %}
3      {% for item in values %}
4          <p>Name: {{item.first_name}}, Email: {{item.email}}, Cancer: {{item.vCancer}}, Symptoms: {{item.vSymptoms}}, Treatment: {{item.vTreatment}}, Result: {{item.result}}%</p>
5      {% endfor %}
6   {% endblock %}
7
8   <button type = "button" > <a href="{{ url_for('home') }}">Home</a></button>
```

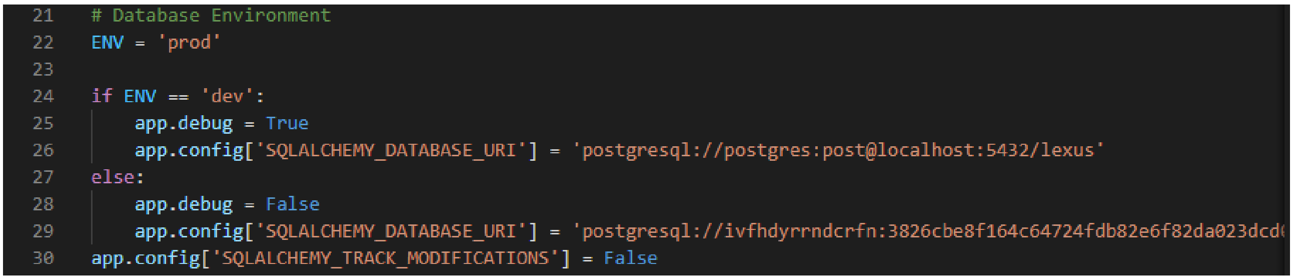
Import Required Libraries



```python
from flask import Flask, redirect, url_for, render_template, request, session, flash, request
import os
import pickle

#from loadmodel import load_model
from PIL import Image
from flask_login import login_manager, login_user, login_required, logout_user, current_user, LoginManager
from flask_login.mixins import UserMixin
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash

import torch
import torch.nn as nn

from torchvision import transforms
from torchvision.transforms import transforms
```

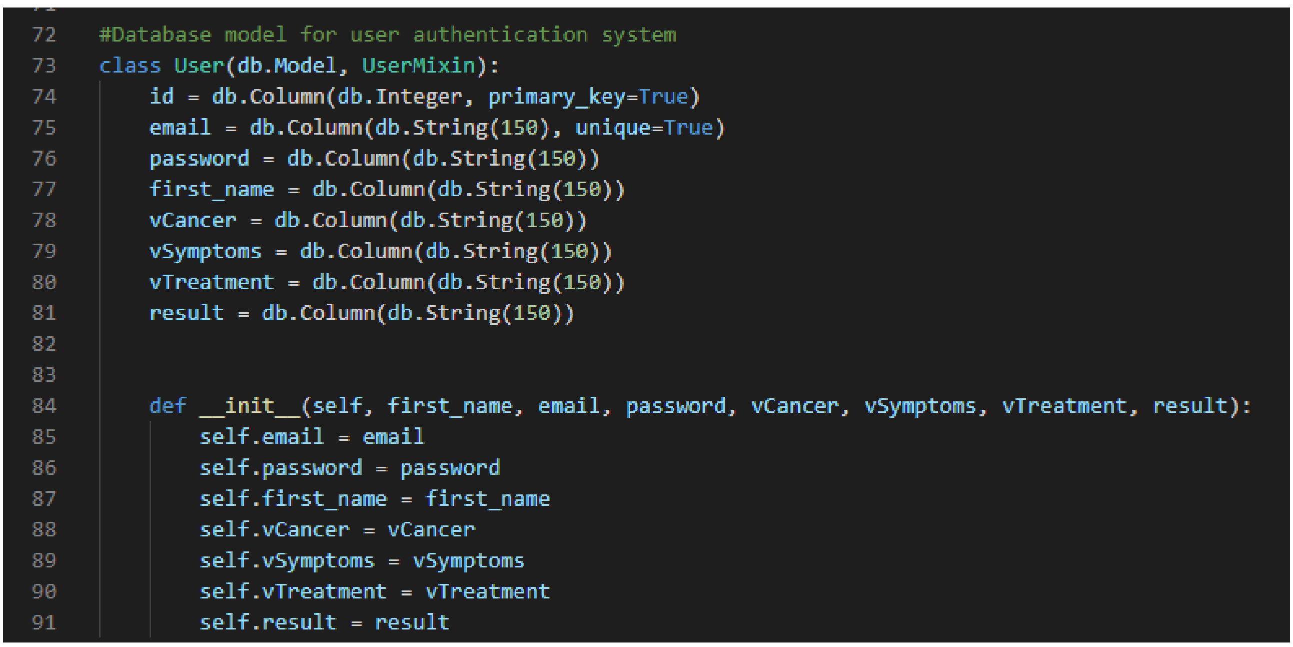
Database Environment (Heroku, 2021)

- If running on localhost, change **Line 22** in main.py to ENV = ‘dev’, to connect to local PostgreSQL server, otherwise let ENV = ‘prod’



```python
21  # Database Environment
22  ENV = 'prod'
23
24  if ENV == 'dev':
25      app.debug = True
26      app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:post@localhost:5432/lexus'
27  else:
28      app.debug = False
29      app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://ivfhdyrrndcrfn:3826cbe8f164c64724fdb82e6f82da023dcd
30  app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

- Our current database model, to delete or add a new column, makes changes under **Line 73** class function and initialization under function **Line 84**.

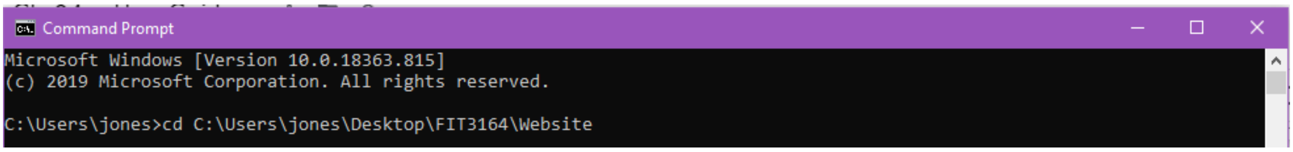


```python
72  #Database model for user authentication system
73  class User(db.Model, UserMixin):
74      id = db.Column(db.Integer, primary_key=True)
75      email = db.Column(db.String(150), unique=True)
76      password = db.Column(db.String(150))
77      first_name = db.Column(db.String(150))
78      vCancer = db.Column(db.String(150))
79      vSymptoms = db.Column(db.String(150))
80      vTreatment = db.Column(db.String(150))
81      result = db.Column(db.String(150))
82
83
84      def __init__(self, first_name, email, password, vCancer, vSymptoms, vTreatment, result):
85          self.email = email
86          self.password = password
87          self.first_name = first_name
88          self.vCancer = vCancer
89          self.vSymptoms = vSymptoms
90          self.vTreatment = vTreatment
91          self.result = result
```

Deploying Website (Goel, 2021; Nutan, 2020; Shawky, 2019)

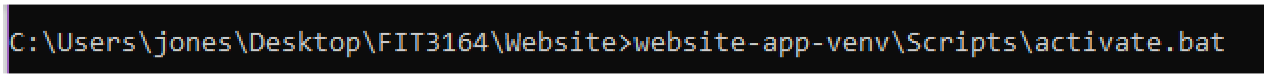
Create Virtual Environment in Website Directory

- Go to command prompt (if using Windows) or Terminal (in Mac)
- Locate to the Website folder in the local repository



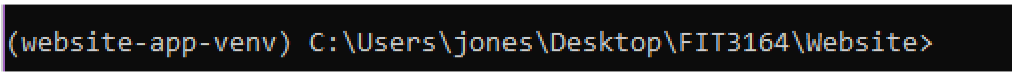
```
Microsoft Windows [Version 10.0.18363.815]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\jones>cd C:\Users\jones\Desktop\FIT3164\Website
```

- Install virtual environment and create a new virtual environment in the Website folder.
    - To install virtual environment, can use **pip install virtualenv**
- For this project, the virtual environment is located in the Website folder, under website-app-venv.
- To activate the virtual environment, can use the following code: **website-app-venv\Scripts\activate.bat**



```
C:\Users\jones\Desktop\FIT3164\Website>website-app-venv\Scripts\activate.bat
```

- To see if the virtual environment is activated, it would show the following on command prompt.



```
(website-app-venv) C:\Users\jones\Desktop\FIT3164\Website>
```

Create a Heroku Account (if deploying on Heroku)

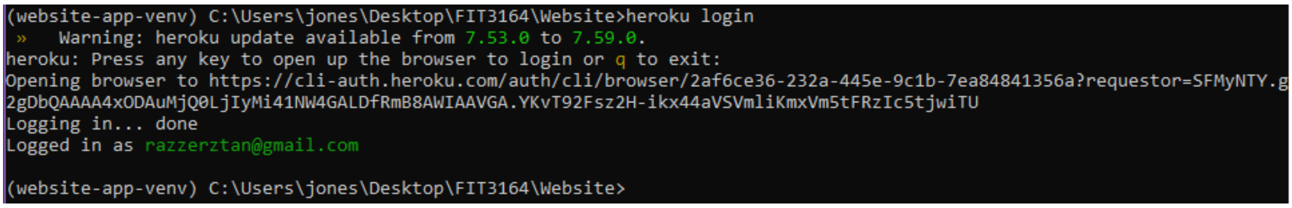
- As the project was deployed into Heroku, the following steps would be applicable if the user would also like to deploy into Heroku. If user would like to deploy using other services, such as AWS or Google Cloud, the steps would be mostly similar
- To create a free account on Heroku, we can use this link https://id.heroku.com/login

Download GIT

- Another requirement to deploy the website is to install GIT on the local machine.
- A fresh copy of GIT could be found here https://git-scm.com/downloads

Download Heroku CLI

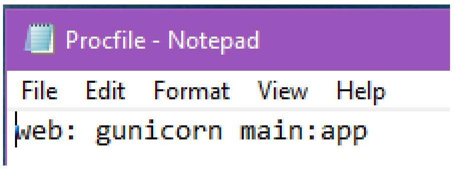
- To have ease of accessibility to the deployed website, it is recommended to use Heroku's command line interface as it can integrate directly with command prompt/terminal, making it easy to push updates to the website.
- Heroku's CLI can be downloaded from this link https://devcenter.heroku.com/articles/heroku-cli#download-and-install
- To login into your heroku account, type **heroku login** on the command prompt



```
(website-app-venv) C:\Users\jones\Desktop\FIT3164\Website>heroku login
 »   Warning: heroku update available from 7.53.0 to 7.59.0.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/2af6ce36-232a-445e-9c1b-7ea84841356a?requestor=SFMyNTY.g
2gDbQAAAA4xODAuMjQ0LjIyMi41NW4GALDfRmB8AWIAAVGA.YKvT92Fsz2H-ikx44aVSVmliKmxVm5tFRzIc5tjwiTU
Logging in... done
Logged in as razzerztan@gmail.com

(website-app-venv) C:\Users\jones\Desktop\FIT3164\Website>
```

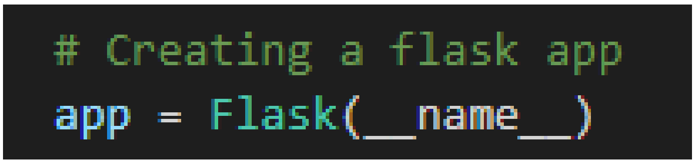
Procfile (Loeber, 2020)

- Another requirement for deployment of the website into Heroku is the presence of a Procfile.
- On the Python console, make sure gunicorn is installed. Otherwise, **pip install gunicorn**
- If no files have been renamed, the Procfile given in the local repository would be sufficient for deployment. However, if the main python script file has been renamed, then the Procfile would need to be changed as well



```
web: gunicorn main:app
```

- The format to rename the Procfile is given as follows:
    - **main** refers to the name of the python file.
    - **app** refers to the instance of Flask which is inside the **main.py** file.

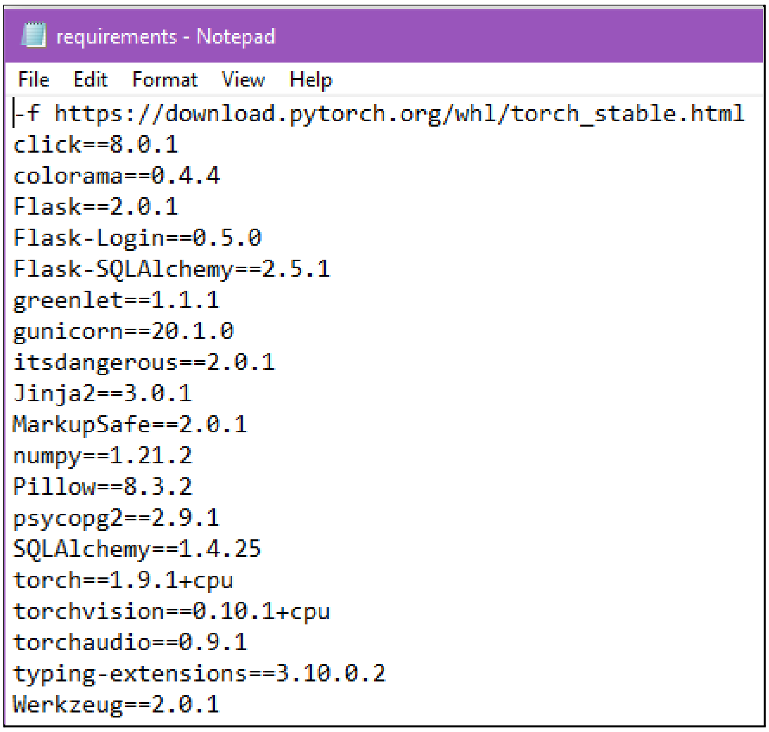


```
# Creating a flask app
app = Flask(__name__)
```

    - Change the above two parameters accordingly.

requirements.txt (Loeber, 2020)

- requirements.txt is a text file used to take note of all the packages the main python file is using.
- If no additional packages have been implemented, the current requirements.txt would be sufficient. Otherwise, we can use **pip freeze > requirements.txt** on the Python Console to get the new packages.



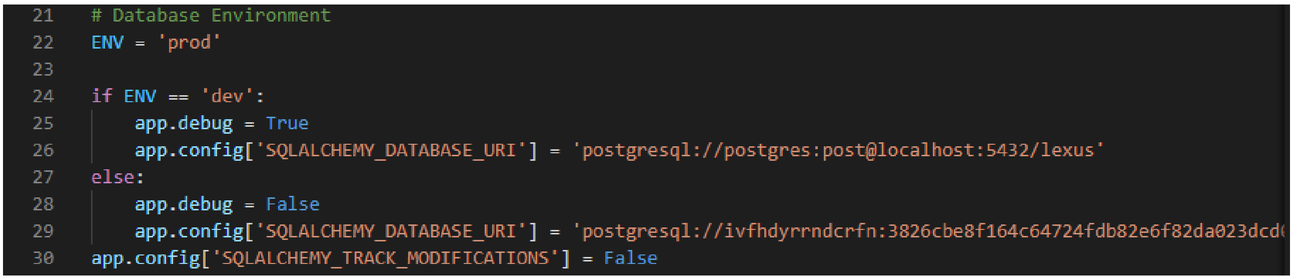
```
-f https://download.pytorch.org/whl/torch_stable.html
click==8.0.1
colorama==0.4.4
Flask==2.0.1
Flask-Login==0.5.0
Flask-SQLAlchemy==2.5.1
greenlet==1.1.1
gunicorn==20.1.0
itsdangerous==2.0.1
Jinja2==3.0.1
MarkupSafe==2.0.1
numpy==1.21.2
Pillow==8.3.2
psycopg2==2.9.1
SQLAlchemy==1.4.25
torch==1.9.1+cpu
torchvision==0.10.1+cpu
torchaudio==0.9.1
typing-extensions==3.10.0.2
Werkzeug==2.0.1
```

- Here, we need to edit the versions of the packages torch, torchvision and torchaudio to only include the CPU model - highlighted in yellow above.

- The current PyTorch version can be seen in this website: https://pytorch.org/get-started/locally/ (PyTorch, 2021)
- Choose Linux, Pip, Python and CPU

Download PostgreSQL (Heroku, 2021)

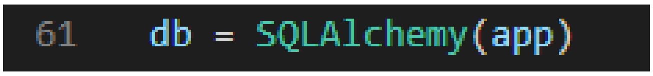
- PostgreSQL can be installed from this link: https://www.postgresql.org/download/
- Once the pgAdmin has been set up - it would ask to set up master password and passwords for the superuser. Client should set the password accordingly.
- Make sure the Heroku add-on for PostgreSQL is added. If not, insert the following command on the command prompt: **heroku addons:create heroku-postgresql:hobby-dev --app app_name**
  - Here, app_name refers to the name of the website to be deployed to Heroku
- Go back to the command prompt and enter the following command to get the database URL from Heroku: **heroku config --app app_name**
- Copy the link given by the output and paste it on the main.py file in the Website folder - **Line 29**



```
21  # Database Environment
22  ENV = 'prod'
23
24  if ENV == 'dev':
25      app.debug = True
26      app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:post@localhost:5432/lexus'
27  else:
28      app.debug = False
29      app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://ivfhdyrrndcrfn:3826cbe8f164c64724fdb82e6f82da023dcd
30  app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

Activating Database in Heroku (Loeber, 2020)

- To activate the database infrastructure, on the command prompt, type
  - **heroku run python**
  - **from app import db**



```
db = SQLAlchemy(app)
```

NOTE: app is the Flask instance in the **main.py,** and **db** is the SQLAlchemy instance

  - **db.create_all()**
  - **exit()**

Appendix

Goel, R. (2021, February 16). Heroku: Deploy your Flask App with a Database Online. Retrieved from https://medium.com/analytics-vidhya/heroku-deploy-your-flask-app-with-a-database-online-d19274a7a749

Heroku. (2021). Heroku Postgres | Heroku Dev Center. Retrieved from https://devcenter.heroku.com/articles/heroku-postgresql

Loeber, P. (2020, August 5). Create & Deploy A Deep Learning App - PyTorch Model Deployment With Flask & Heroku | Python Engineer. Retrieved from https://www.python-engineer.com/posts/pytorch-model-deployment-with-flask/

N. (2020, October 11). Deploy Machine Learning Model with Flask on Heroku - Nutan. Retrieved from https://medium.com/@nutanbhogendrasharma/deploy-machine-learning-model-with-flask-on-heroku-cd079b692b1d

PyTorch. (2021). PyTorch. Retrieved from https://pytorch.org/get-started/locally/

Shawky, M. (2019, March 15). How to deploy your trained PyTorch model on Heroku - Mohamed Shawky. Retrieved from https://medium.com/@mohcufe/how-to-deploy-your-trained-pytorch-model-on-heroku-ff4b73085ddd