INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

Department of Computing

"CANINE"

Module: - 5COSC009C Software Development Group Project (SDGP)

Module Leader: - Banu Athuraliya.

Team Members:

Name	IIT ID	UOW
Manoharan Arushan	2019208	W1761076
Kogula Kangaiveniyan	20191101	W1790354
Anojan Gnanasegram	2019474	W1761364
Thenuka Kengatharan	20191080	W1790342
Selvasothy Arulraj	2018103	W1761872
Marshal Johnsan	2019329	W1761286

Table of Contents

List of Tables

Table of Figures

1. Implementation

1.1 Chapter overview

This chapter will discuss the implementation aspects of the CANINE after completing the design phase. In this chapter, a detailed description of programming language used, and libraries selected have been added to provide framework, technology and tools selection as well as websites and problems during implementation and solution. To illustrate the implementation process, screenshots and code snippets are used.

1.2 Overview of the prototype

- Dog Breed identification – uploaded image will process into the system and it will extract features using four pretrained models. Then combining all the extracted features to create combined feature maps and fitting it into a neural network. After we will be getting our output.
- Dog Disease prediction – After getting the symptoms selected by the user it will go to the server in json format then server will get as an array then the prediction with train model after that prediction return the prediction value to the frontend and display the disease in the text field under the predict button.

1.3 Technology selections

Technology selections for CANINE are detailed below along with the language selection and libraries selection.

1.3.1 Language Selection

Python complies with the requirements of the CANINE system and offers more libraries, and it is good to use a language without the overhead of learning. Therefore, Python was then selected as the main language for developing its primary functionality.The reasons for the choosing of Python are:

- python is Object-oriented, supporting about 300 standard libraries and tools such as Jupyter.
- Simplicity and fast rate of user friendly.
- Due to the use of its data structure for repetitive work with short-length code, coding capabilities in this language can be identified.

- The development of modules, libraries and frames such as Scrapy, Scikit-Learn, pandas, and others for third parties.

The prototype has been agreed to publish as a web application for end-users through the web application process. The author has decided to use react native to develop the front end, but it is absurdly fast, fully loaded and reassurely safe.

1.3.2 Libraries selection

1.3.2.1 Selection of CNN Libraries

The identification of dog breed is another major component of the proposed solution. The author has therefore decided to use the CNN approach to implement the component, so finding the suited library is very important.

Comparison	TensorFlow	Keras	PyTorch
Primitive neural networks such as convolution are implemented	✓	✓	✘
Prototyping is fast, simple and easy to use.	High	High	Low
Helpful APIs are implemented, such as model saving & model training	High	High	Low
Results are much easier to read and concise.	✓	✓	✘

Table 1: Selection of CNN libraries

Keras is an abstract layer, which uses other deeply-learned libraries while TensorFlow is one of the libraries. In fact, TensorFlow now includes Keras as a default layer for abstraction.

The author has thus decided to build deep learning architectures with a few strings of Python code, using the Keras library on top of computational powerhouse TensorFlow, as a backend. It helps to build and deploy the Machine learning project easily.

1.3.2.2 Pandas

Pandas is an open-source library, and it was used for data analysis and manipulation. It is mainly used to manage the large dataset and save the time when import the dataset. It was used to bridge the gap between raw data, data modelling and reading data from CSV file.

1.3.2.3 Pickle

It is used to serializing and de-serializing the project structure. Pickle is used to convert a python object into a character stream. In this project, we used Data Frame for predict the dog diseases.

1.3.2.4 Flask

Flask is used for deploying the model for machine learning and processes requests from node js.

1.3.2.5 Matplotlib

Matplotlib is a data visualization and graphical plotting library for Python. In this project, Matplotlib is used to plot the labels in a straight line that called as line plot graph and that displayed the accuracy level of the disease prediction.

1.3.2.6 Scikit-Learn

Scikit-learn is a most useful Python Machine learning library and it will support various algorithm. In this project, it is support to Random forest Algorithm.

1.3.2.6.1 Random Forest classifier

Random Forest is most popular Machine learning algorithm. The random forest algorithm is given priority for implementing Canine for disease prediction because of its high accuracy and flexibility in both classification and regression problems. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification).

1.3.2.7 NumPy

It is an open-source numerical library. In this Project NumPy is used for reading the image and converting the image into an np array.

1.4 Implementation of the data science component

1.4.1 Dog breed identification

This CANINE aims to demonstrate how pretrained CNN models can be used as function extractors, one of the most effective techniques for the transfer learning.

A reasonable question arises, ‘why must we use this technique? Why should we not simply use transfer learning?’ If you try, you'll realize that it's quite hard for one model to handle the problem because you would get higher loss and less accuracy. It is even difficult to differentiate between 120 breeds of dogs. Hence, CNN image identification architecture in two major parts, ‘Extract features’ based on conv layers, and "classifier," usually based on fully connected layers. Furthermore, to use different feature extractors to identify the breed this dog belong to.

Step 1: Extract features from the dataset by using selected pretrained models.

Step 2: Extract features from row data and combine the features together.

Step 3: Use a simple deep net with one dense layer and a heavy dropout layer to figure out patterns in the feature extracted from the data by using all those combined features.

Step 4: GlobalAveragePooling2D to extract a pooled output from our selected pertained models.

First import TensorFlow and other libraries. In here we have a vector of probabilities for each class, the output of our predictor should be converted to the same format. This is a row vector of num_classes length with a 1 at the label's index and 0 everywhere else for each input (One-hot Encoding).

Authors select this shape (331, 331, 3) according to the documentation of the pertained model in keras application. They are InceptionV3, Xception, InceptionResNetV2, NasNetLarge.

```
input_shape = (331,331,3)

def images_to_array(directory, label_dataframe,target_size = input_shape):
  image_labels = label_dataframe['breed']
  images = np.zeros([len(label_dataframe),target_size[0], target_size[1], target_size[2]], dtype=np.uint8)
  y = np.zeros([len(label_dataframe),1],dtype=np.uint8)

  for ix, image_name in enumerate(tqdm(label_dataframe['id'].values)):
    img_dir = os.path.join(directory, image_name +'.jpg')
    img = load_img(img_dir, target_size= target_size)
    images[ix] = img
    del img
    dog_breed = image_labels[ix]
    y[ix] = class_to_num[dog_breed]

  #One hot encoder
  y = to_categorical(y)
  return images,y
```

Figure 1: Implementation of adding images into array

In here, giving the model's name (InceptionV3/Xception/InceptionResNetV2/NasNetLarge), processor, then taking input size and data. First data enter into input layer with dimension of input size then input layer applying model_preprocessor to create equality preprocessor input of the model. Model will now extract features from the process input. After that extract a pooled output of the model from our selected pertained models using GlobalAveragePooling2D.

Then creating the model to extract features from the image and it will return feature_map.

```
img_size = (331,331,3)

def get_features(model_name, model_preprocessor, input_size, data):

    input_layer = Input(input_size)
    preprocessor = Lambda(model_preprocessor)(input_layer)
    base_model = model_name(weights='imagenet', include_top=False,
                            input_shape=input_size)(preprocessor)
    avg = GlobalAveragePooling2D()(base_model)
    feature_extractor = Model(inputs = input_layer, outputs = avg)

    #Extract feature.
    feature_maps = feature_extractor.predict(data, verbose=1)
    print('Feature maps shape: ', feature_maps.shape)
    return feature_maps
```

Figure 2: Get features function

Combining all the extracted features to create final feature map.

```
final_features = np.concatenate([inception_features,
                                 xception_features,
                                 nasnet_features,
                                 inc_resnet_features,], axis=-1) #axis=-1 to concatinate horizontally

print('Final feature maps shape', final_features.shape)
```

Figure 3: Extract the final feature map

After that authors are creating two callbacks lrr and earlystop. The callbacks help a model to perform things like saving a progress of a model, checking a progression of models and stopping training early on if a model does not improve. lrr will monitor the validation accuracy and reduce the learning rate (ReduceLROnPlateau). EarlyStopoing will monitor the validation loss and it will stop the process of training if nothing significant is happening.

```
#Learning Rate Annealer
lrr= ReduceLROnPlateau(monitor='val_acc', factor=.01, patience=3, min_lr=1e-5,verbose = 1)

#Prepare call backs
EarlyStop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Figure 4: Callbacks

Building the neural network requires model layers to be configured, then compiling the model. During the model compile step :

1. loss function measures the model's accuracy during training. To "steer" the model in the right direction, you want to minimize this function.
2. Optimizer updates the model based on the data and the loss function of the model.
3. Metrics is used for training and testing monitoring. The example below uses accuracy, which is the right fractions of the images.

The loss and accuracy metrics are displayed when the model trains. The accuracy of the training data is approximately 0.9585(or 96%).

```
#Prepare Deep net

model = Sequential()
model.add(Dropout(0.7,input_shape=(final_features.shape[1],)))
model.add(Dense(n_classes,activation= 'softmax'))

model.compile(optimizer=adam,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

#Training the model.
history = model.fit(final_features, y,
            batch_size=batch_size,
            epochs=epochs,
            validation_split=0.2,
            callbacks=[lrr,EarlyStop])
```

Figure 5: Model training

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dropout (Dropout)            (None, 9664)              0
_________________________________________________________________
dense (Dense)                (None, 120)               1159800
=================================================================
Total params: 1,159,800
Trainable params: 1,159,800
Non-trainable params: 0
```

Figure 6: Model Summary

```
test_loss, test_acc = model.evaluate(final_features, y, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
320/320 - 1s - loss: 0.1288 - accuracy: 0.9585

Test accuracy: 0.9585208296775818
```

Figure 7: Test Accuracy

1.4.2 Canine Disease prediction

Data Science is a set of different tools, algorithm/calculations, and AI standards with the objective to find discover examples from the data collection of a source. The main target of using data science for image classification is to find the breed of dog by uploading an image and disease prediction of dog by getting symptoms with gender and age as an input from users.

Here will see about disease prediction, getting dataset for canine project from a book called "BlackWell's Five-Minute Veterinary Consultant" and also we have confirmed from Veterinary Surgeon Dr.Kathirvetpillai what are the details will be helping to do canine project, and collect data from the book.

Data science

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
#from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix
import seaborn as sns
from sklearn.datasets import load_digits
from sklearn.ensemble import RandomForestClassifier

data = pd.read_csv("Dog_Disease_Identification.csv")
sym = pd.read_csv("sym_count.csv")

# encode the dataset
data = data.fillna('NO')
cols = data.columns
vals = data.values
symptoms = sym['Symtom'].unique()
for i in range(len(symptoms)):
    vals[vals == symptoms[i]] = sym[sym['Symtom'] == symptoms[i]]['Count'].values[0]
```

Figure 8: Implementation of machine learning for disease prediction

Above code about the development of project was start by importing libraries for importing the data from the csv file and cleaning/encode the data which has null value.

```
d = pd.DataFrame(vals, columns=cols)
d = d.replace('NO', 0)
d = d.replace('Both', 5001)
d = d.replace('Male', 5002)
d = d.replace('Female', 5003)
d = d.replace('common ', 6001)
d = d.replace('common', 6001)
d = d.replace('Below Five', 6002)
d = d.replace('Below Ten', 6003)
d = d.replace('Below Fifteen', 6004)

df = d.iloc[:,1:].values
labels = d['Disease '].values

#split the dataset for training and testing
x_train, x_test, y_train, y_test = train_test_split(df, labels, shuffle=True, train_size = 0.85)
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)

from sklearn.ensemble import RandomForestClassifier
rondom = RandomForestClassifier()
rondom.fit(x_train, y_train)

#calculate accurecy
preds = rondom.predict(x_test)
accuracy_score(y_test, preds)
```

Figure 9: Encoding the data set

Above code is after encoding the dataset and split dataset for training and testing part after train the dataset using Random Forest algorithm to predict process we also try Decision Tree algorithm, but the random forest is better than to predict because decision trees are more prone to overfitting, especially when a tree is particularly gone deep so, project will be continued with random forest.

```
from sklearn.ensemble import RandomForestClassifier
rondom = RandomForestClassifier()
rondom.fit(x_train, y_train)
```

```
RandomForestClassifier()
```

Figure 10: Random forest Classifier

1.5 Implementation of the backend component

Disease Prediction

```
import json
from flask import Flask, request, jsonify
from flask_cors import CORS

import pandas as pd
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
rondom = RandomForestClassifier()


app = Flask(__name__)
CORS(app)

@app.route('/api/get/data/disese', methods=['PUT'])
def create_record():

    # Load csv file
    sym = pd.read_csv("../../Data-Science/Diseases/sym_count.csv")

    #get data array from react
    record = json.loads(request.data)

    #processing code to prepair
    # print(record)
    l2_dataframe = pd.DataFrame(record)
    list_f = []
    sample = (l2_dataframe.gender.unique())
    list_f.append(sample[0])
    sample = (l2_dataframe.age.unique())
    list_f.append(sample[1])
    sample = (l2_dataframe.Symptoms.unique())
    i = 1
    while i < len(sample):
        list_f.append(sample[i])
        i+= 1
```

Figure 11: Disease Implementation in Flask

In the above code using flask as backend to obtains the symptoms from user or client-side as an array, that array change as data frame type to predict.

```
    while(len(list_f) < 17) :
        list_f.append(0);

    print(list_f)

    l2_dataframe = pd.DataFrame(list_f)
    cols = l2_dataframe.columns
    vals = l2_dataframe.values
    symptoms = sym['Symtom'].unique()
    for i in range(len(symptoms)):
        vals[vals == symptoms[i]] = sym[sym['Symtom'] == symptoms[i]]['Count'].values[0]

    list_01 = pd.DataFrame(vals, columns=cols)
    list_01 = list_01.replace('NO',0)
    list_01 = list_01.replace('Both',5001)
    list_01 = list_01.replace('Male',5002)
    list_01 = list_01.replace('Female',5003)
    list_01 = list_01.replace('common',6001)
    list_01 = list_01.replace('Below Five',6002)
    list_01 = list_01.replace('Below Ten',6003)
    list_01 = list_01.replace('Below Fifteen',6004)
    print(list_01)

    #read modle
    print(" Read ---------------------------------------")
    with open('../../Data-Science/Diseases/disease_model.pkl','rb') as f:
        predict_model = pickle.load(f)
    print("Suc-----------------------------------------------------")
    #get prediction
    list_last = list_01[0].values
    print(list_last)
    disease = predict_model.predict([list_last])

    print("Your Dog Have ",disease[0],"Disease!!!!")

    #return value to react
    return jsonify(disease[0])
app.run()
```

Figure 12: Disease Prediction

And already trained model will assign to data frame to predict after getting the inputs output is predicted and return as an array index to react page.

Problem and solution for disease prediction

1. Data preprocessing: -

   Getting data from doctor recommendation and the dataset have more medical terms for disease and symptoms so that difficult to understand the terms.

2. Getting train model accuracy: -

   That different to get accuracy to every time we must run the train model it's hard to do.

3. Using some algorithm and got different value of accuracy.

   decision tree algorithm we try to get accuracy, but we had problem with that, so we gone with random forest algorithm.

4. while displaying the accuracy in the graph of the model.
   Try to print the graph clearly but have some problem with that.
5. Push github model for data science part because its large file: -
   After train the data we get model for disease prediction that model file has more than 100mb so the github doesn't work with large file that we hard to work with that.

Breed Identification

```
import os
from tqdm import tqdm
import pickle
import numpy as np
import pandas as pd
from keras import Sequential
from keras.callbacks import EarlyStopping
from keras.callbacks import ReduceLROnPlateau

from keras.optimizers import Adam, SGD
from keras.layers import Dense, Dropout
from flask import Flask, request
from flask_cors import CORS

from keras.layers import Lambda, Input, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from keras.preprocessing.image import load_img

app = Flask(__name__)
CORS(app)


# reading labels csv file
labels = pd.read_csv('Files\\Dataset\\labels.csv')
```

Figure 13: Breed Identification in Flask

Above figure 13 code using flask as backend to obtains the uploaded image from the client-side and it will convert to array.

```
with open('Files\\InceptionV3_s01', 'rb') as f:
    InceptionV3 = pickle.load(f)

with open('Files\\inception_features_s01', 'rb') as f:
    inception_features1 = pickle.load(f)

with open('Files\\inception_preprocessor_s01', 'rb') as f:
    inception_preprocessor = pickle.load(f)

with open('Files\\Xception_s01', 'rb') as f:
    Xception = pickle.load(f)

with open('Files\\xception_features_s01', 'rb') as f:
    xception_features1 = pickle.load(f)

with open('Files\\xception_preprocessor_s01', 'rb') as f:
    xception_preprocessor = pickle.load(f)

with open('Files\\InceptionResNetV2_s01', 'rb') as f:
    InceptionResNetV2 = pickle.load(f)

with open('Files\\inc_resnet_features_s01', 'rb') as f:
    inc_resnet_features1 = pickle.load(f)

with open('Files\\nasnet_features_s01', 'rb') as f:
    nasnet_features1 = pickle.load(f)

with open('Files\\inc_resnet_preprocessor_s01', 'rb') as f:
    inc_resnet_preprocessor = pickle.load(f)

with open('Files\\NASNetLarge_s01', 'rb') as f:
    NASNetLarge = pickle.load(f)

with open('Files\\nasnet_preprocessor_s01', 'rb') as f:
    nasnet_preprocessor = pickle.load(f)
```

Figure 14: Saved Models

In above figure 14 describes importing pickle library and then read the saved model files which already saved. This is the solution for the data science component which is mention in bellow figure 15.

```
# Extract features using InceptionV3
from keras.applications.inception_v3 import InceptionV3, preprocess_input
inception_preprocessor = preprocess_input
inception_features = get_features(InceptionV3,
                                  inception_preprocessor,
                                  img_size, X)

import pickle
with open('InceptionV3_s01','wb') as f:
    pickle.dump(InceptionV3,f)

with open('inception_preprocessor_s01','wb') as f:
    pickle.dump(inception_preprocessor,f)

with open('inception_features_s01','wb') as f:
    pickle.dump(inception_features,f)
```

Figure 15: Saving pertained model

In here, pretrained models were imported from the keras so that it took more massive time. To overcome the problem, save the pretrained by importing the pickle library. Anyhow, it took time for extract the data features. Below attached the evident screenshots.

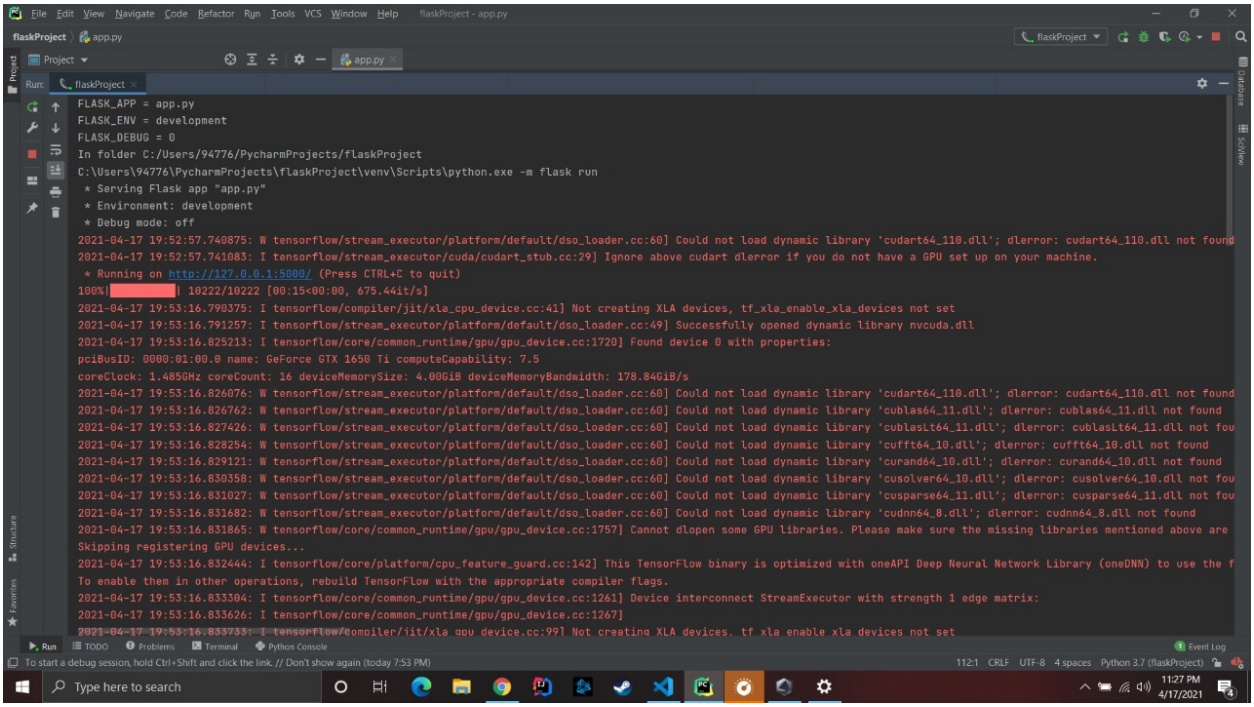

Figure 16: Screenshot 1

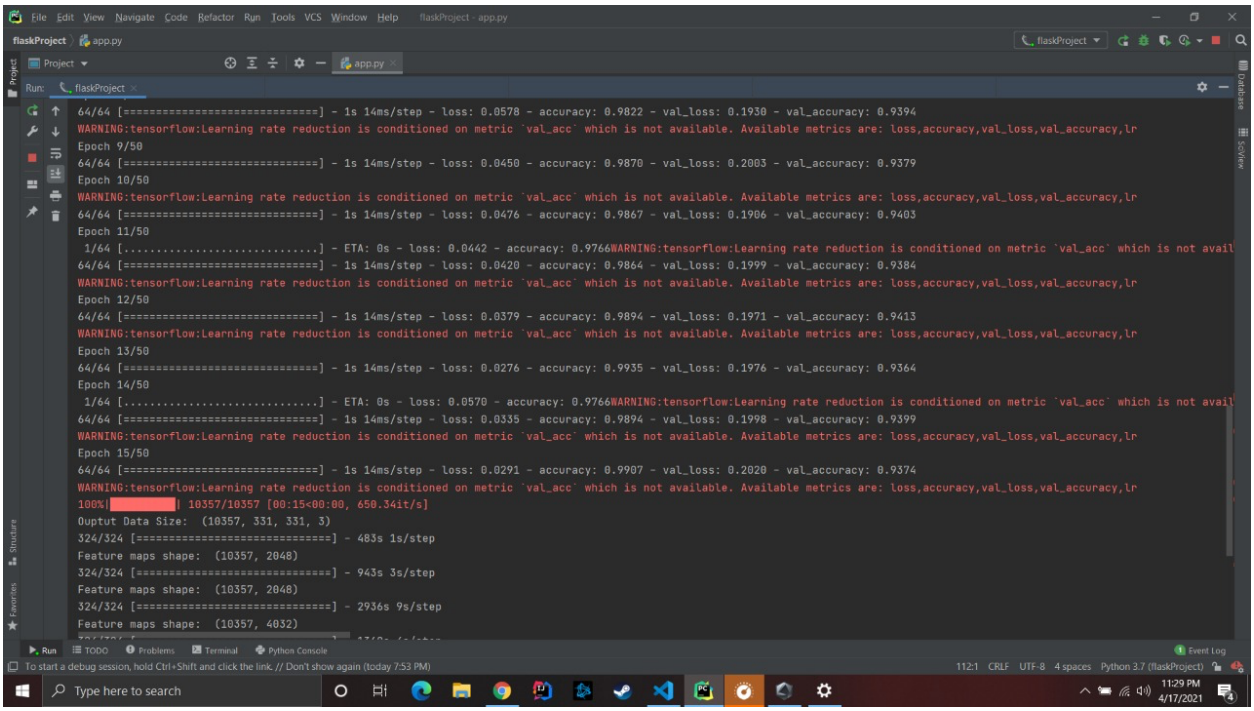

Figure 17: Screenshot 2

```
@app.route('/api/post/data', methods=['POST'])
def create_record():
    import base64
    imgdata = base64.b64decode(request.data['image'])
    filename = 'some_image.jpg'  # I assume you have a way of picking unique filenames
    with open(filename, 'wb') as f:
        f.write(imgdata)
    img_g = load_img("C:\\Users\\94776\\some_image.jpg", target_size=img_size)
    img_g = np.expand_dims(img_g, axis=0)
    print(f"Predicted label: {classes[np.argmax(img_g[0])]}")

    # processing code
    return classes[np.argmax(img_g[0])]


if __name__ == '__main__':
    app.run()
```

Figure 18: Breed Identification

User uploaded image of the URL will load in the variable called img_g according to that predict the dog breed.

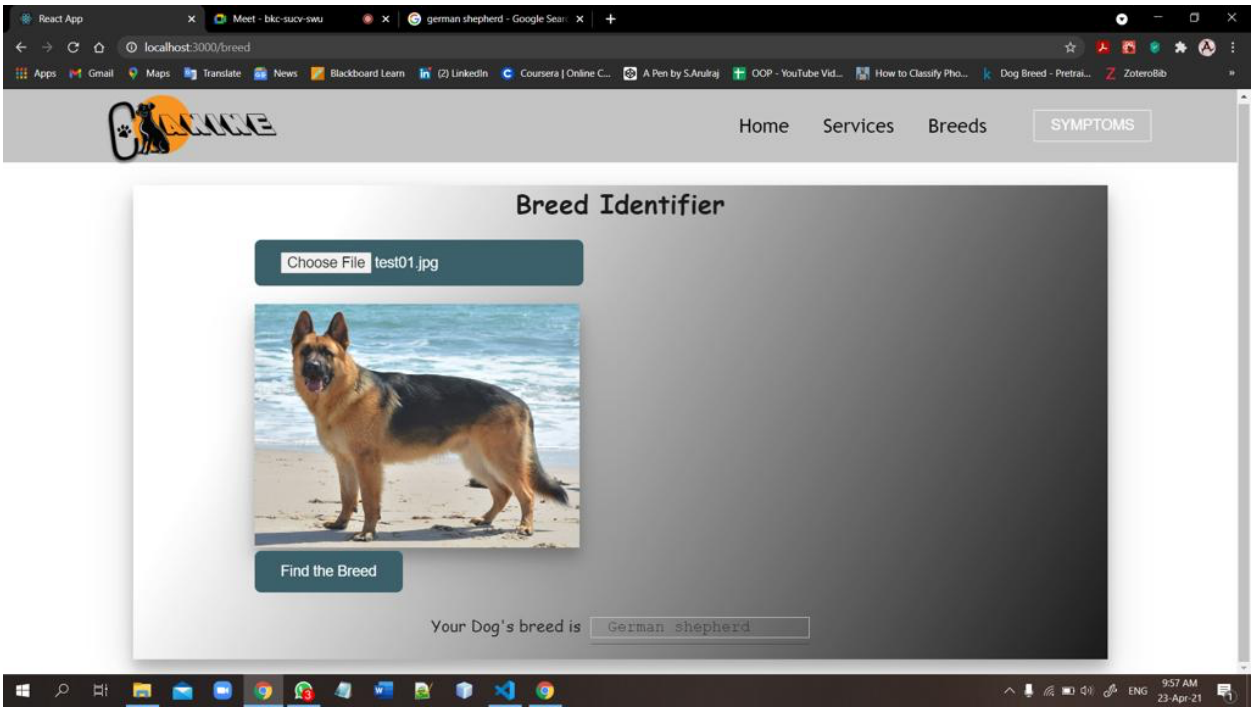

Figure 19: Predicted the dog Breed

1.6 Implementation of the front end component

CANINE web application mainly build for dog breed identification and disease identification. Further, the user can view the Dogs profile entered by other user and add their dog details too but if the login user can only do this in CANINE web application. Implemented by some components of pages namely Home, Service, Products, Breed, Symptoms, Signup, Login, Dog profile and Details of dog.

Home Page

The hero section component is implemented to display a home page to the user that includes with two button which Dog Profile and Get started button navigate to the Canine component that displays two options which is Breed identification and Disease identification. If user clicks Breed it navigate to breed component user can upload dog image from their machine and it will display the dog breed in the text box. If user clicks on the Disease user can add the symptoms of their dogs by selecting from option which reveal 9 search boxes its user friendly to select the symptoms by searching. Then system will display the disease in text box.

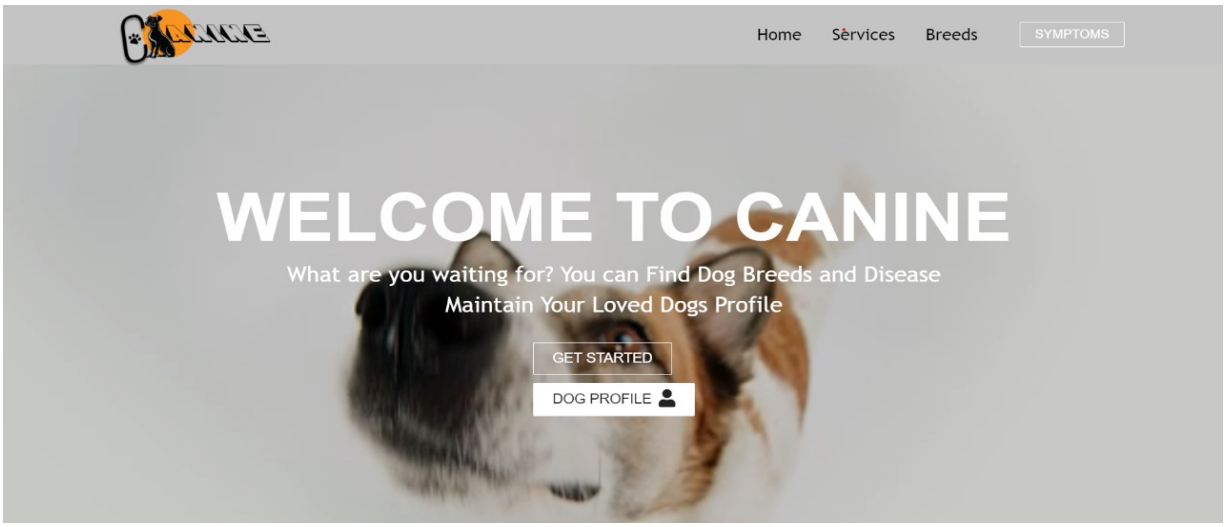

Figure 20: Home Page

Signup Page

The sign-in component will work to capture the user credentials which will be sent back to the NodeJS web server. A form group has been created to validate the Login component. Here we will authenticate the user with the users in the database. If the username and password match, we will generate a token that will be sent back to the sign-in page and then it Navigate to , otherwise the user will be responded with an error message.

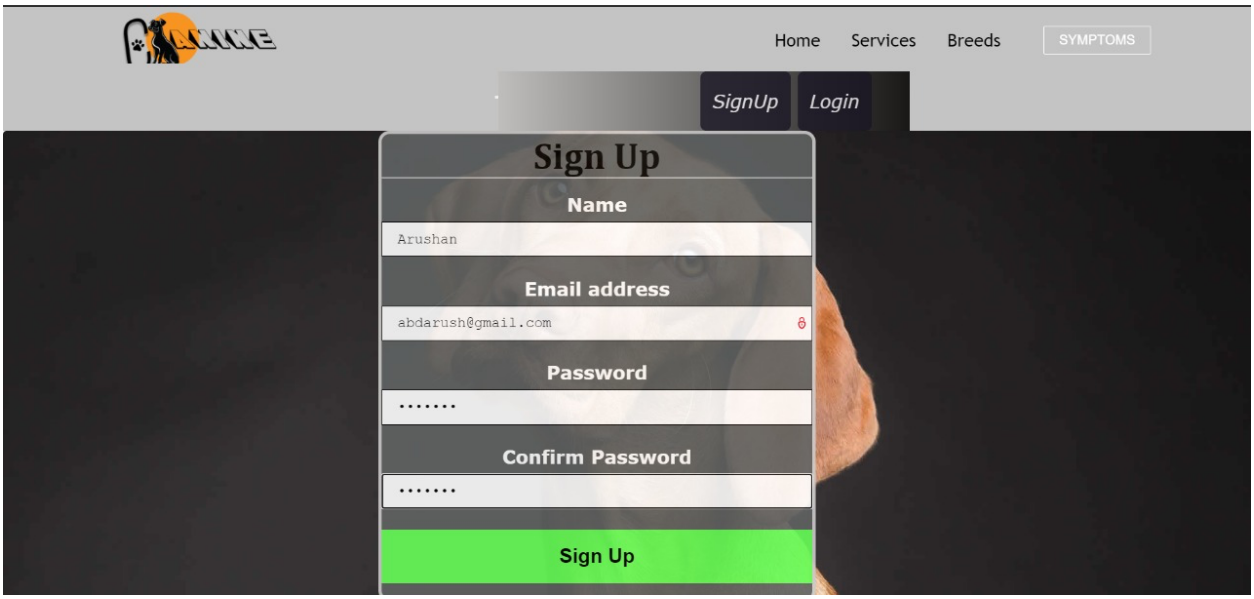

Figure 21: Signup Page

Login Page

In this component, a form group has been created to validate the Login component. The validator will check whether the entered username and the password are valid. Even though the user changing the username and the password values in the validators it will still validate the form belong to the entered value. The error message will be displayed only when there is an error occurs. A functionality created inside this component to pass the value to the NodeJS webserver to authenticate the user. When the authentication is successful the user should be navigated to the Dog Profile through the Add Service component. That's the working flow of the Login component functionalities. So, the following code snippets are the implementation of the Login component

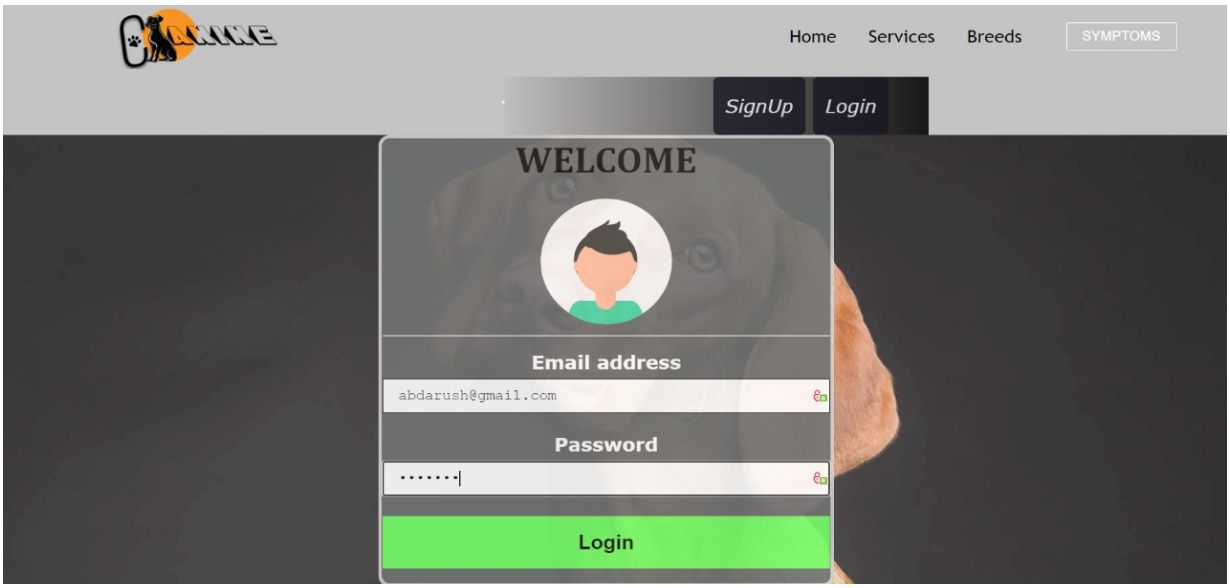

Figure 22: Login Page

Services Page

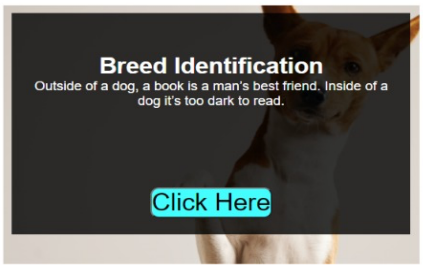

Figure 23: Service Page

Image upload for breed identification

Figure 24: Image upload for breed identification

In Breed component user can upload dog image. Used materialui library when user click on predict button it will pass as URL to the backend to predict the disease.

```
class Breed extends Component{
    constructor(props){
        super(props)
        this.state = {
          breed : "",
          file: null
        }
        this.handleChange = this.handleChange.bind(this)
        this.onSubmit = this.onSubmit.bind(this)
      }
      handleChange(event) {
        console.log('image : ', event.target.files[0]);
        let file = event.target.files[0];
        if(file) {
            const reader = new FileReader();
            reader.onload = this._handleReaderLoaded.bind(this);
            reader.readAsBinaryString(file);
        }
        this.setState({
          file: URL.createObjectURL(event.target.files[0])
        })
      }
```

Figure 25: Breed Component

Disease prediction

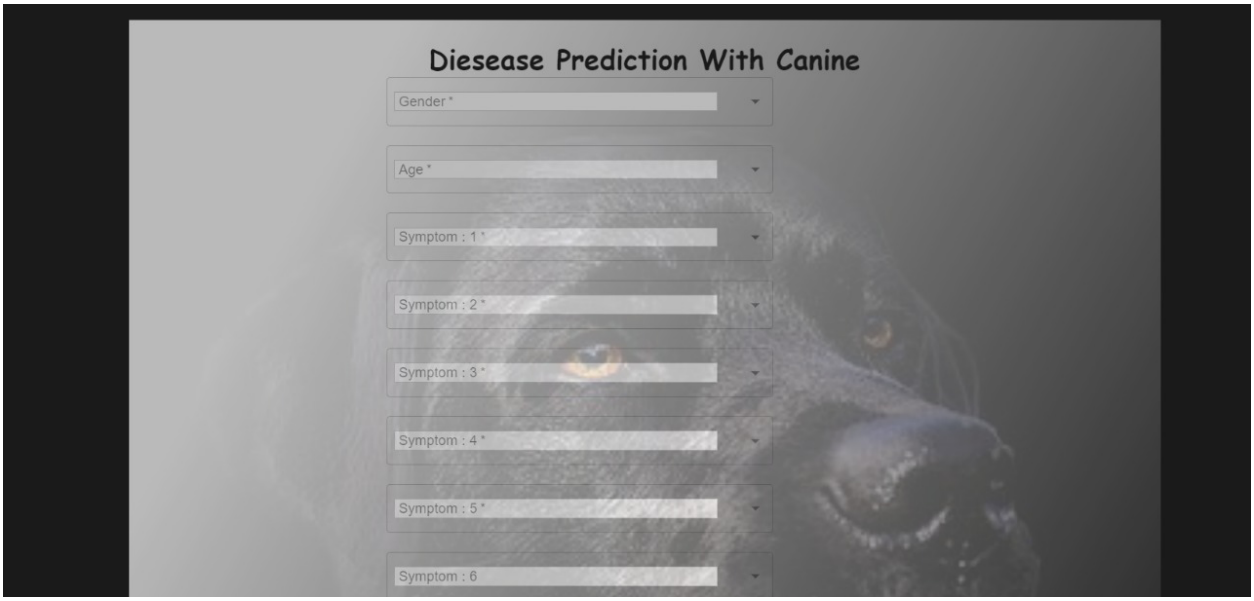

Figure 26: Disease prediction 1

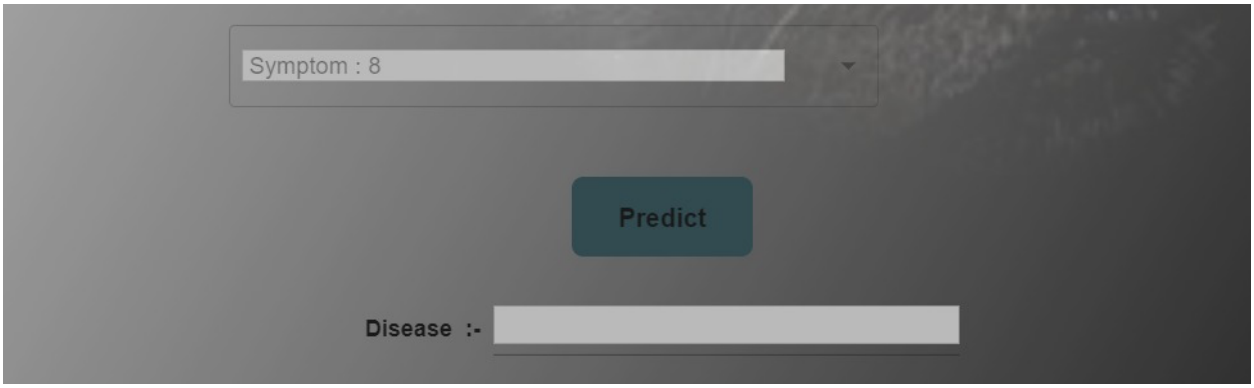

Figure 27: Disease prediction 2

```
Complexity is 77 Bloody hell...
class Symptoms extends Component{
    constructor(){
        super()
        this.state={
            gen:"",
            age:"",
            sym1:"",
            sym2:"",
            sym3:"",
            sym4:"",
            sym5:"",
            sym6:"",
            sym7:"",
            sym8:"",
            disease1:"",

            data:[]
        }
        this.onSubmit = this.onSubmit.bind(this);
    }

    //onSubmit function to passing user input to server
    Complexity is 15 You must be kidding
    onSubmit=(e)=>{
        e.preventDefault()
        //console.log("Work")
        if(this.state.sym6 !== '' && this.state.sym7 === '' && this.state.sym8 === ''){
          var formData = [this.state.gen,this.state.age,this.state.sym1,this.state.sym2,this.state.sym3,this.state.sym4,this.state.
          }
          else if(this.state.sym6 !== '' && this.state.sym7 !== '' && this.state.sym8 === ''){
           formData = [this.state.gen,this.state.age,this.state.sym1,this.state.sym2,this.state.sym3,this.state.sym4,this.state.sym
```

Figure 28: Symptoms Class

In Symptoms component creating a constructor to create the variable to input symptoms and to assign the disease get from the backend. User can add 8 symptoms to find disease but must need to add maximum 5 symptoms. If user didn't add maximum 5 error message will display in text box.

```
47          $.ajax({
48            url: 'http://127.0.0.1:5000/api/get/data/disese',
49            type: 'PUT',
50            dataType: 'json',
51            contentType: 'application/json',
52            data: JSON.stringify(formData),//predicted data assigned to the variable
53            success: function(data, textStatus, xhr) {
54                //alert("Your Dog have " + data +" Disease");
55                $('#txt').text(data + "- this disease your dog have");
56                this.setState({disease1:data}); //assign returning data from backend
57            }.bind(this),//identify its a functon ti reactjs
58            error: function(xhr, textStatus, errorThrown) {
59                console.log('Error in Operation');
60            }
61          });
62          }
63
```

Figure 29: Symptoms Component

After sending the symptoms selected by the user it will go to the server in json format after prediction return the prediction value to the component and predicted data assigned to the variable (disease1). If the prediction is not correct system will throw an error message.

```
<Autocomplete
value = {this.state.gen}
onChange={(event, val) => {
  this.setState({gen:val})
}}

  id="gender"
  options={gender}
  getOptionLabel={(option) => option.gender}
  style={{width:450,marginLeft:300}}
  renderInput={(params => <TextField {...params} label="Gender" variant="outlined" required={true}/>)}//validation
/><br/>
```

Figure 30: Materialui input library

For the text boxes Canine used materialui input library . for the symptom options converted the csv data set file to the json and assigned to the variable symptoms Disease. So, all the input text box can access the symptoms.

Other Pages

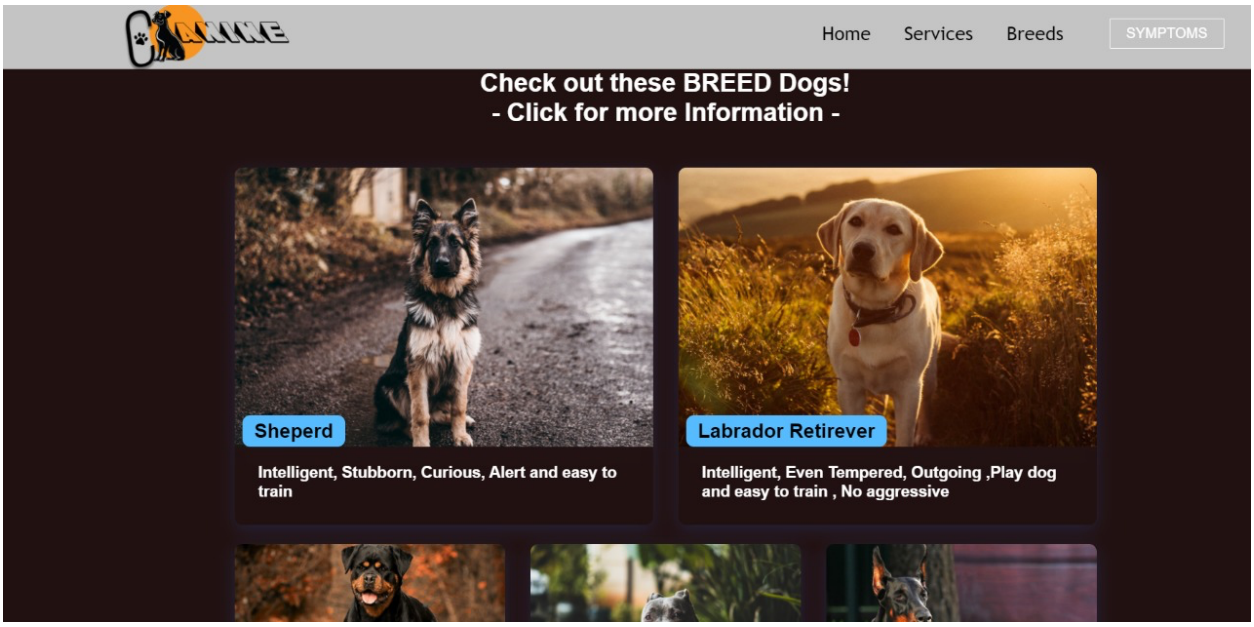

Figure 31: Dog Breed details Page

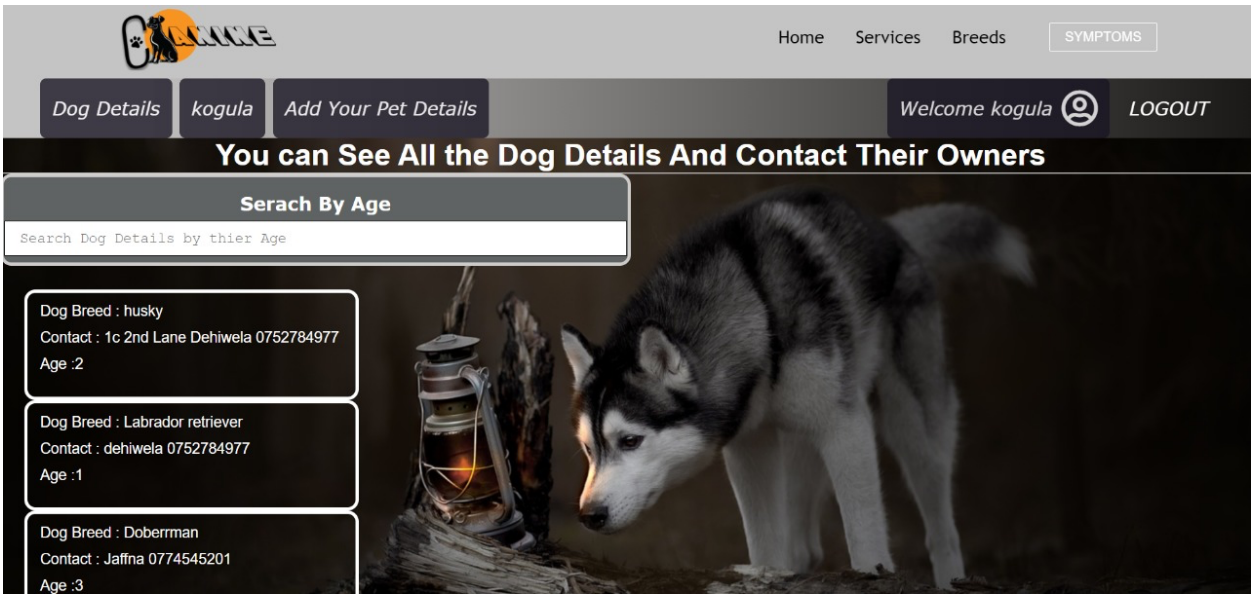

Figure 32: Dog and their owner details

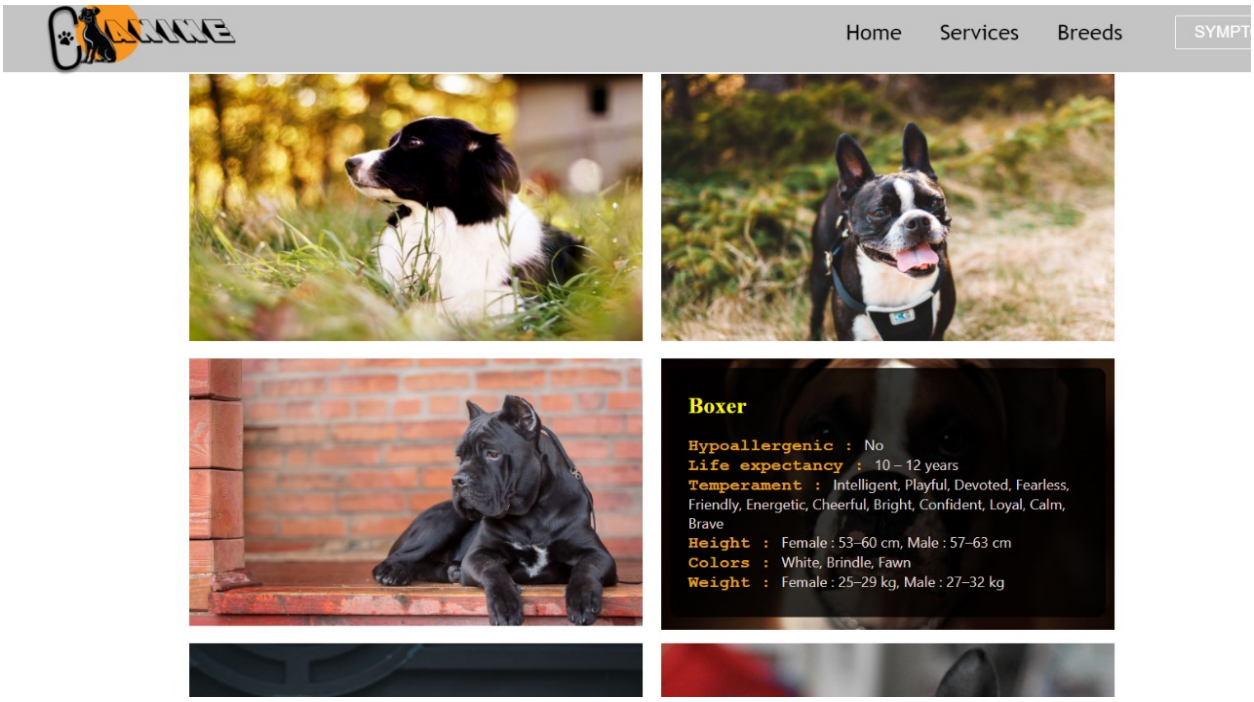

Figure 33: Dog details Page

1.6 Chapter Summary

The above chapter discussed about the implementation of the system and it provided the data science component that is used. The frontend and the backend that is been used in CANINE. The following chapter will discuss about the testing of the system that is been used

2. Testing

2.1 Chapter Overview

The previous chapter discussed the implementation of the System. This chapter describes the testing phase of CANNIE. Section 2.2 discuss about the testing criteria. Section 2.3,2.4 discuss about the testing the functional and non-functional requirements of CANINE. Other section also illustrates the test cases with their outcomes.

2.2 Testing Criteria

The CANNIE system has a set of functional and non-functional requirement. It is very important to test and validate the functional and non-functional standards against the business and quality requirements of the system.

2.3 System Testing

System testing has been used to check CANNIE, as discussed previously. System testing is the final test step in which the device is normally evaluated in functional testing. After the device tests are completed, the program is considered a successful system. Bellow details the results of most important system test cases.

ID	Test Cases	Priority	Result
ST01	User should be able to access the system with login	High	Pass
ST02	The system saves the dog profile in the database	High	Pass
ST03	The system display the dog details	Medium	Pass
ST04	Predict the diseases for dog	High	Pass
ST05	Identify dog breed	High	Pass

Table 2: System Testing Table

2.4 Testing functional requirements

A Black Box approach was used for the functional testing of the system. This was done to test all the requirements of the solution proposed.

Functional Requirement	Status
User should be able to upload a dog image.	Pass
User should be able to view the breed of the dog.	Pass
User can view the symptoms entered by the previous user.	Pass
User should be able to view the predicted diseases for dog.	Pass
User should be able to view other user's dog details.	Pass
User should be able to view the breed details.	Pass

Table 3: Functional requirements Testing

2.5 Testing non-functional requirements

The system's non-functionality is described in SRS report section 4.7 and will be tested for accuracy, performance, reliability, usability, and scalability.

Test case No	Feature tested	Test case description	Test case condition	Expected result	Actual result	Status
1.0	NF1	Accuracy	Random Forest classifier	Good accuracy	86.97%	Pass
1.1			CNN	Good accuracy	95.85%	Pass
2.0	NF2	Performance	Load Testing	100%	100%	Pass
2.1			Overall Testing	100%		
3.0	NF3	Usability	User friendly design	Good UI/UX design	The system has good image and colors	Pass
4.0	NF4	Scalability	Code comment standards	Good code comments	Comments were used in code	Pass

Table 4: Non-functional requirements Testing

2.6 Accuracy Testing

Accuracy is an important factor to fulfill the functional requirements of the system. The system's accuracy determines the success of the application as well as the success of this research project. For breed identification accuracy relies mostly on the test images and test labels. These two processes determine the accuracy of the summary and it is important that both processes produce accurate results based on the two processes.

Given bellow results are test accuracy of the two main models. To compare the model's performance on a test data set to evaluate accuracy.

Disease prediction accuracy

```
#calculate accurecy
preds = rondom.predict(x_test)
accuracy_score(y_test, preds)
```

```
0.8697478991596639
```

Figure 34: Disease prediction accuracy

Breed prediction accuracy

```
test_loss, test_acc = model.evaluate(final_features, y, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
320/320 - 1s - loss: 0.1288 - accuracy: 0.9585

Test accuracy: 0.9585208296775818
```

Figure 35: Breed prediction accuracy

Our CNN has achieved a test accuracy of over 95%. The accuracy of the test dataset is somewhat less than the accuracy of the training dataset. This difference between training accuracy and test accuracy is overfit.

```
320/320 - 0s - loss: 0.1288 - accuracy: 0.9585
```

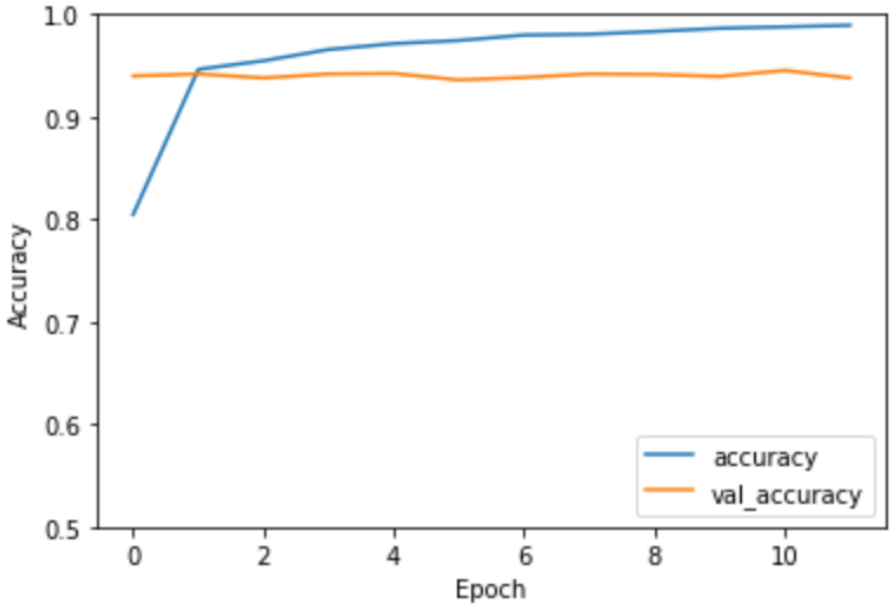

Figure 36: Validation Accuracy Graph

Bellow code is for test the validation accuracy.

```
#evaluate the model
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(final_features, y, verbose=2)
```

Figure 37: Validation Accuracy Code

2.7 Performance testing

Performance testing was utilized to test CANINE. It is an important aspect of the testing process.

2.7.1 Load testing

Apache Benchmark tool was used following load tests.

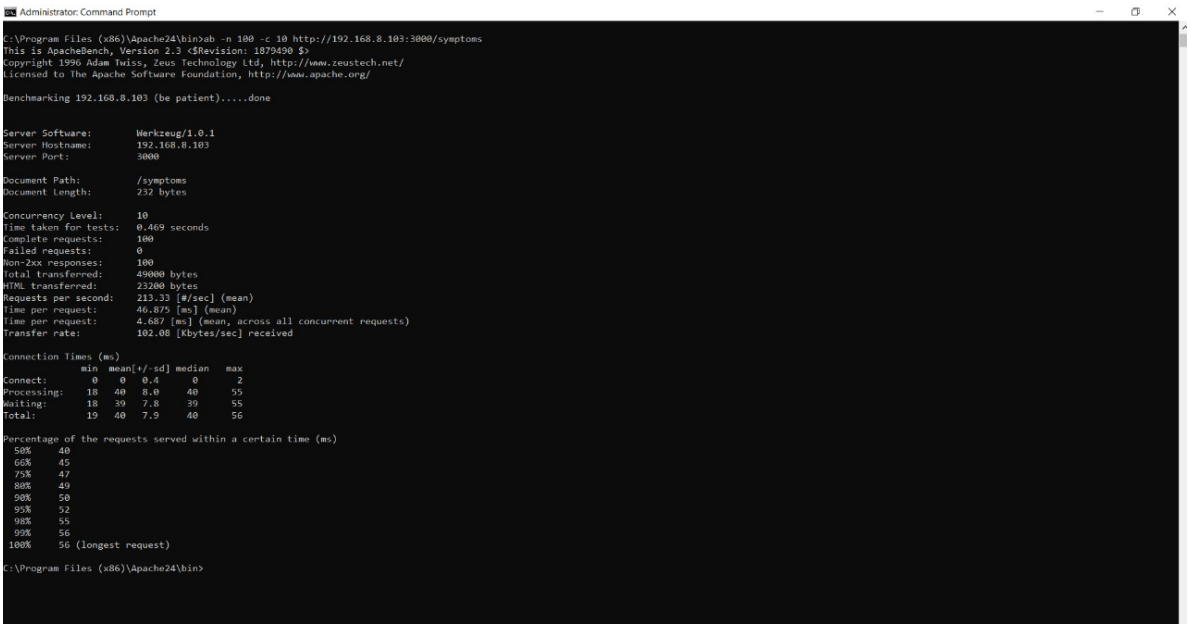

Figure 38: 100 Request for Disease Prediction

These figures measure website load management performance. In the entire case, 100 users have simultaneous access to the website and 100 requests are forwarded. To achieve this competitive rate, website control is quite fast and therefore this number of userss can be handled at once easily.

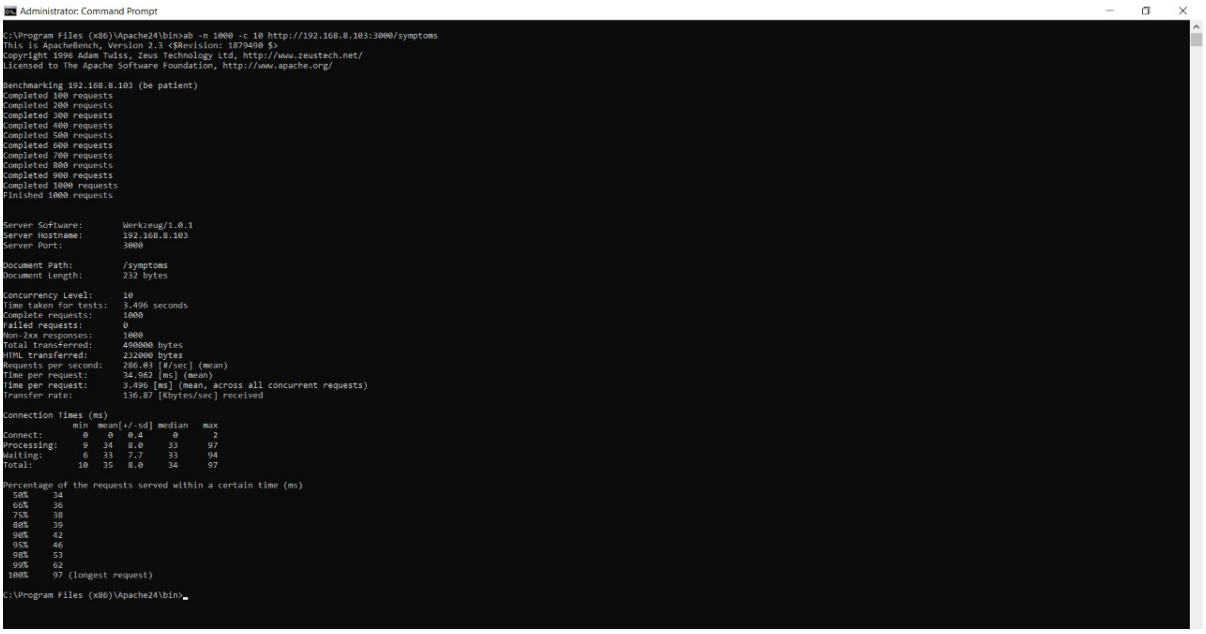

Figure 39: 1000 Request for Disease Prediction

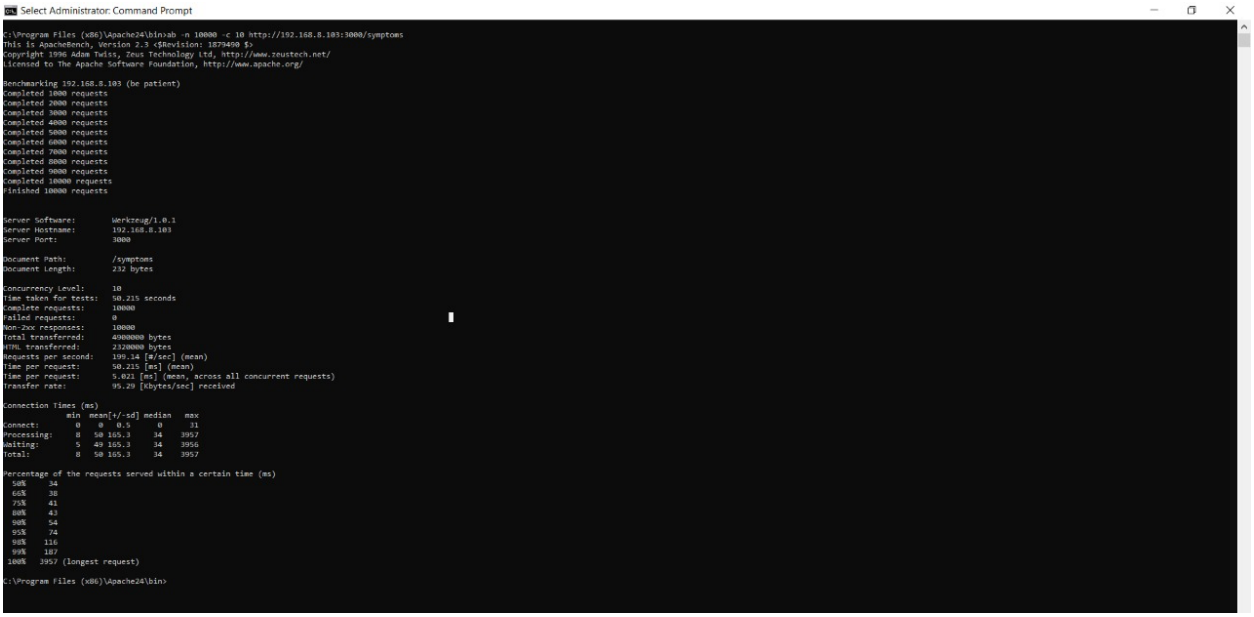

Figure 40: 10000 Request for Disease Prediction

The performance of web load handling is measured in this figure. 100 users have simultaneous access to the website and 10000 requests each are forwarded. To achieve this competitive rate, the above figure is its performance and this performance test does not require failed requests.

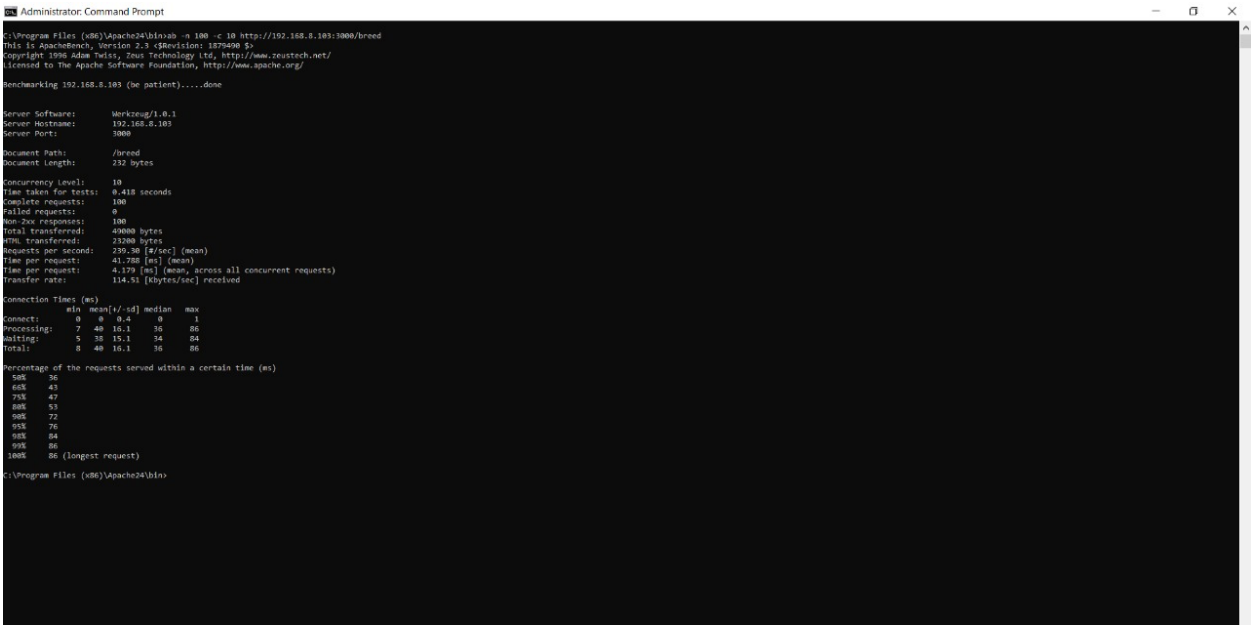

Figure 41: 100 Request for Breed Identification

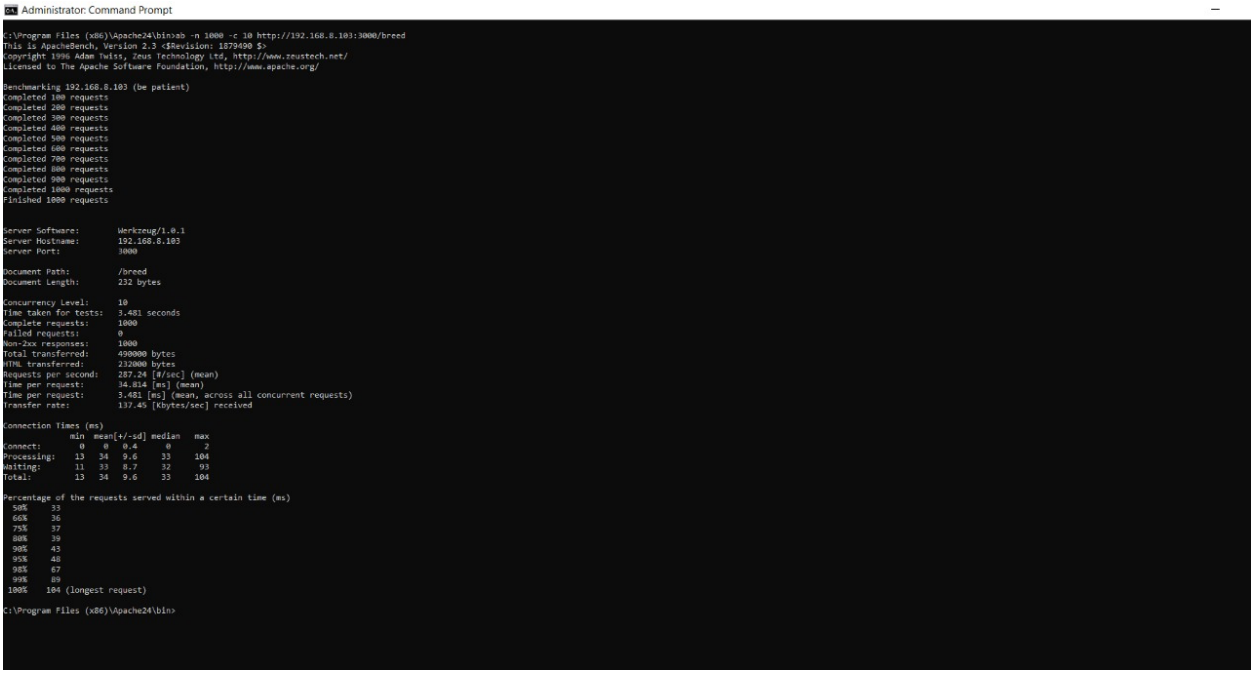

Figure 42: 1000 Request for Breed Identification

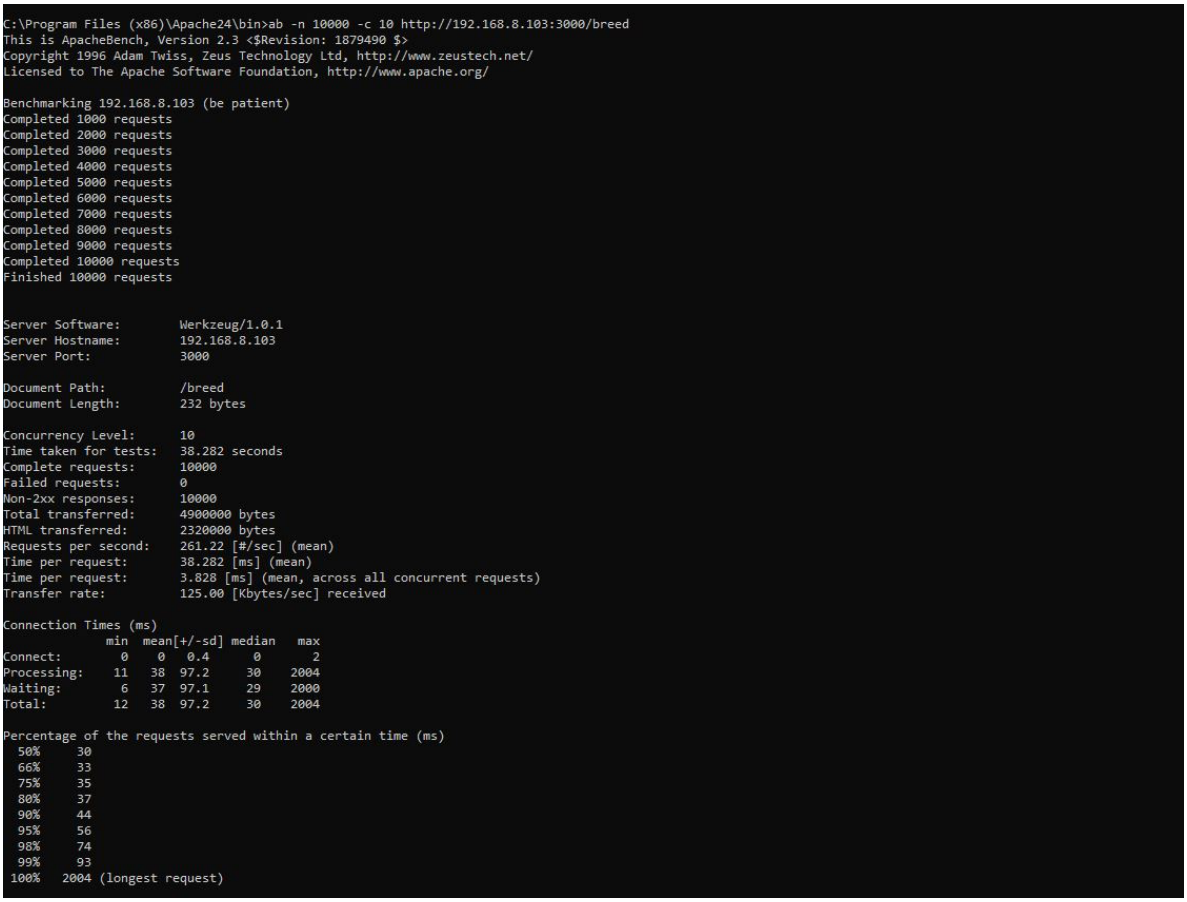

Figure 43: 10000 Request for Breed Identification

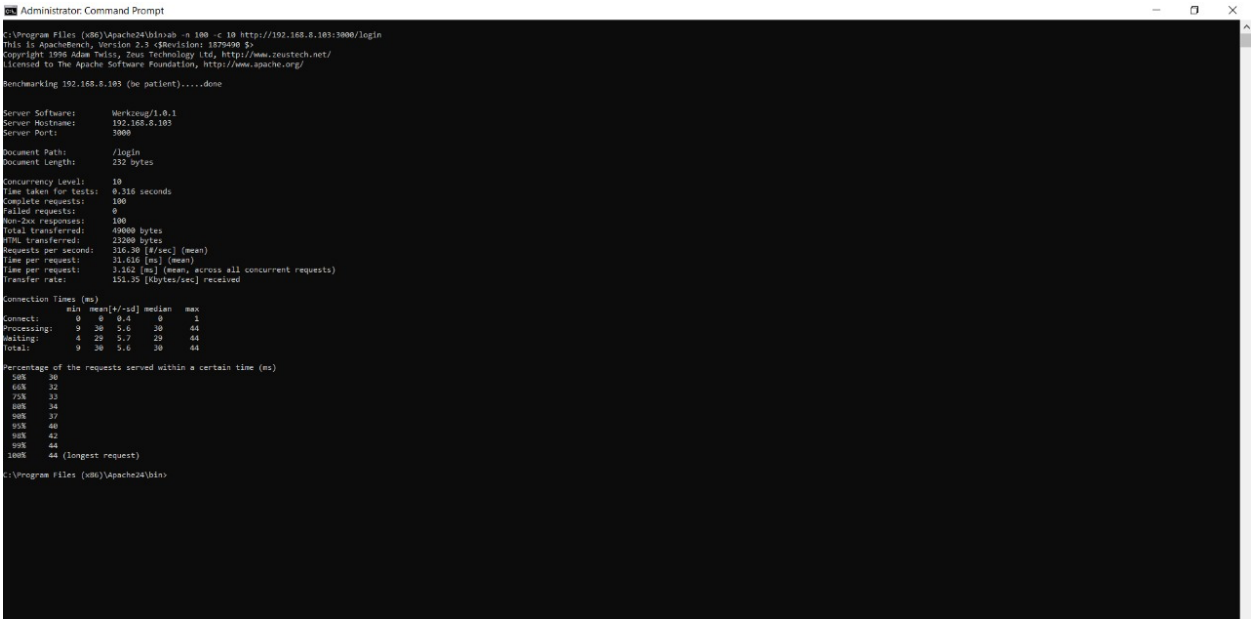

Figure 44: 100 Requests for Login

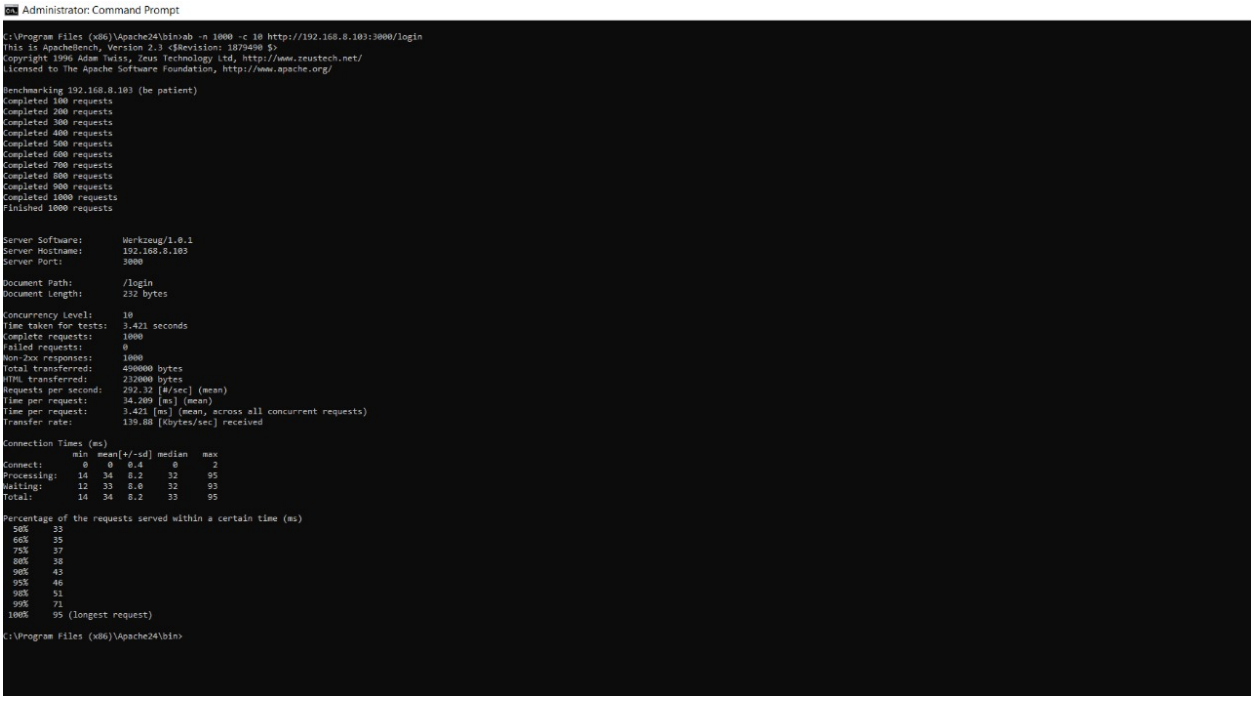

Figure 45: 1000 Requests for Login

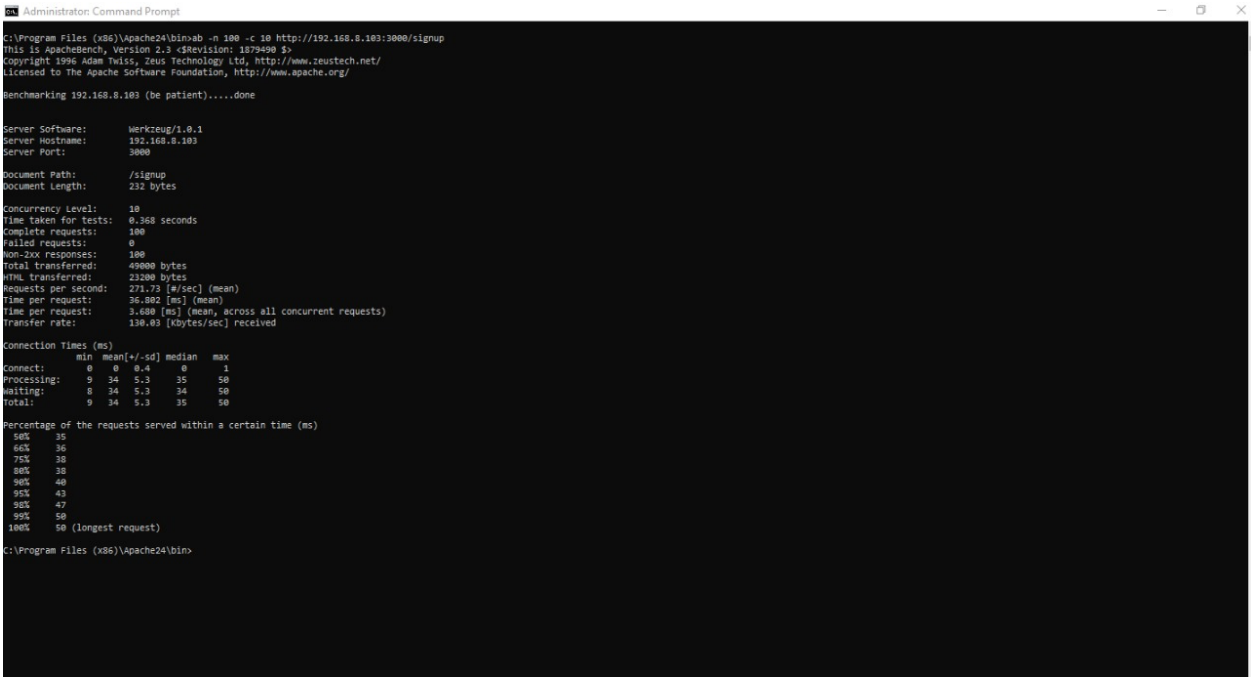

Figure 46: 100 Requests for Signup

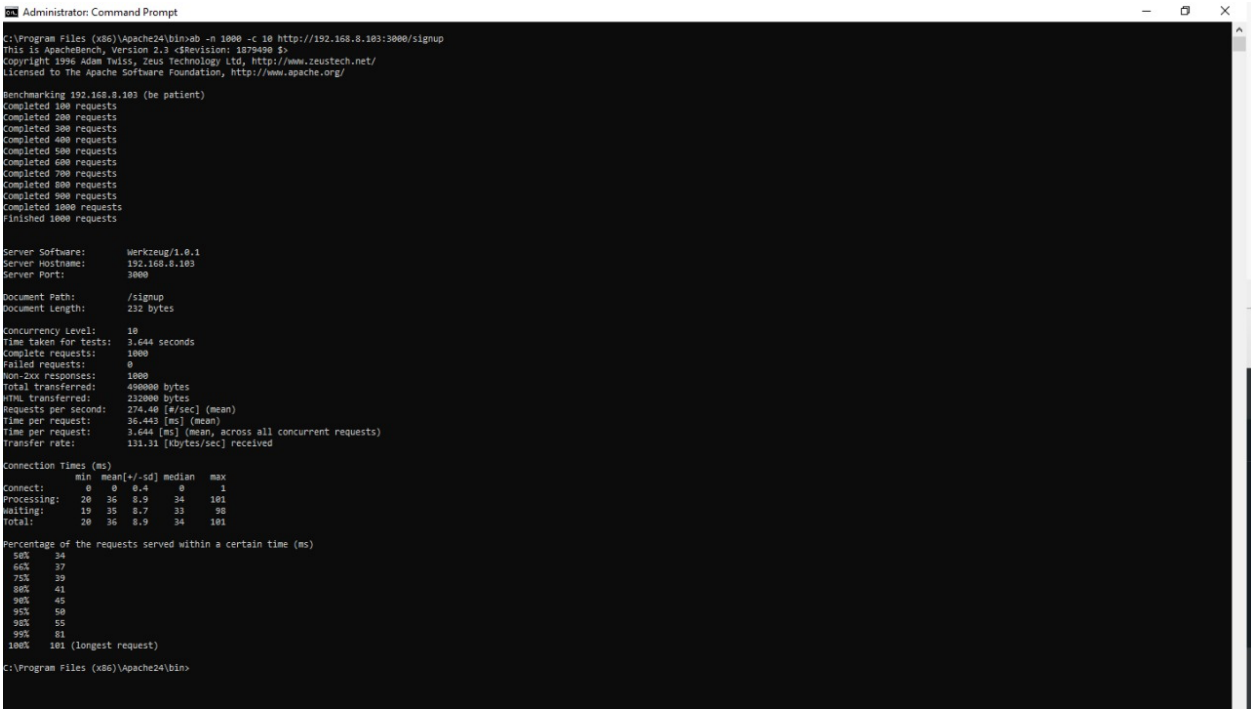

Figure 47: 1000 Requests for Signup

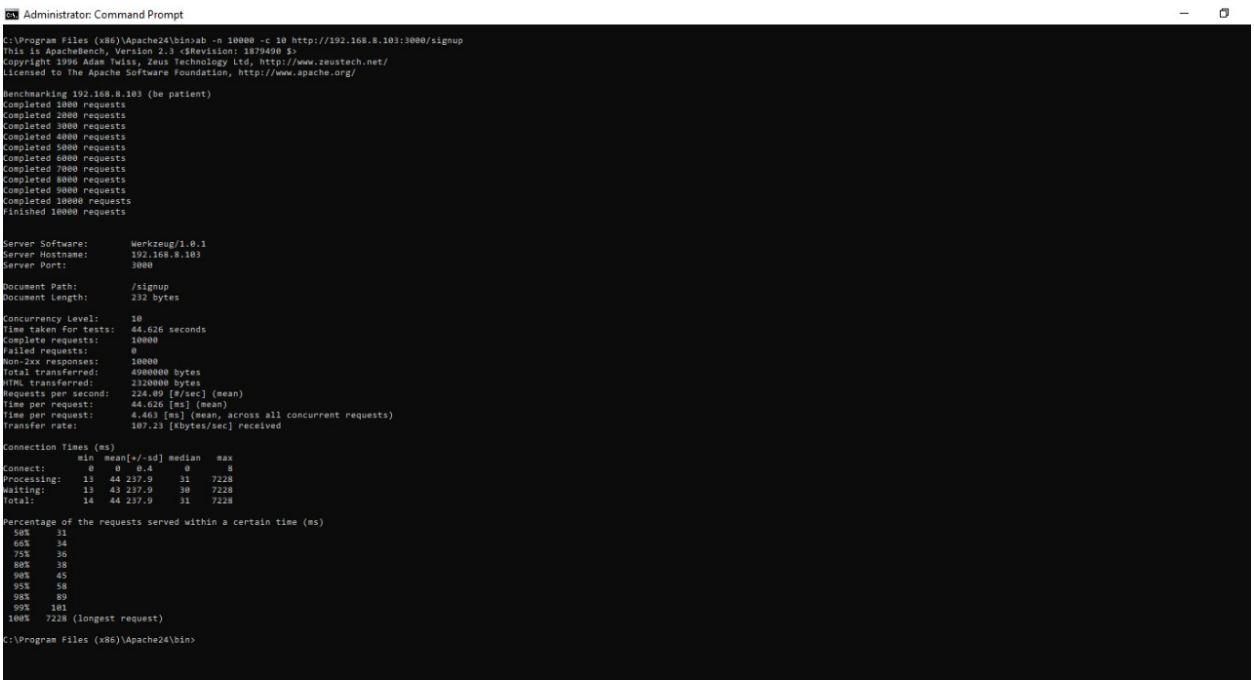

Figure 48: 10000 Requests for Signup

2.7.2 Overall Performance Testing

The overall Google Lighthouse tool performance testing was analyzed. This report included performance, accessibility, best practices, and SEO with a percentage using this tool. Bellow figure will describe

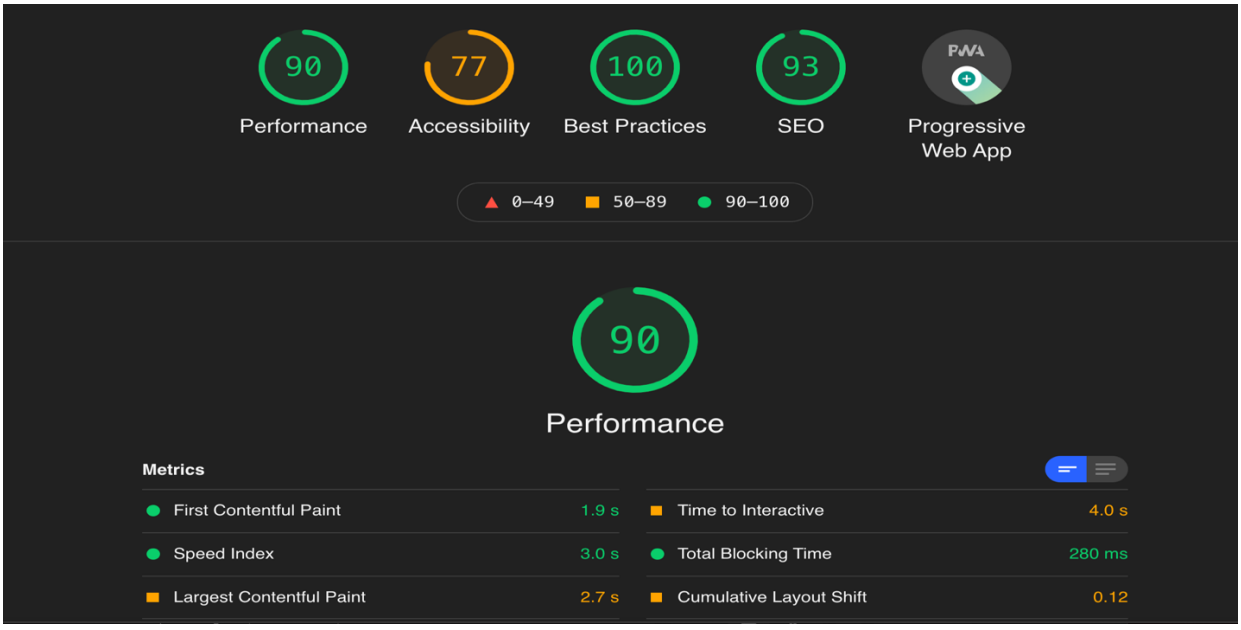

Figure 49: Overall Performance Testing

2.8 Usability testing

Figure 50: Navigation Bar

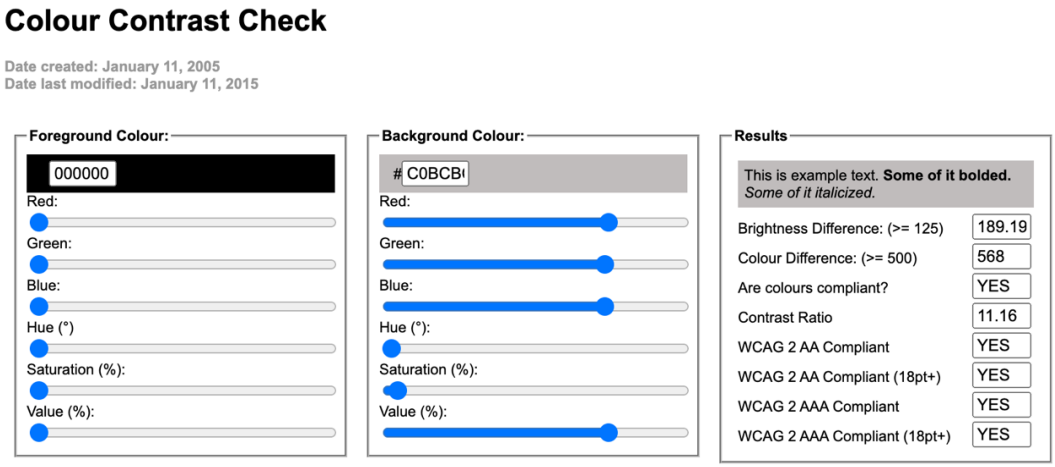

Figure 51: Navigation Bar color contrast checker

2.9 Compatibility testing

Compatibility ensures that all browsers are running well. Google Chrome, Safari and Mozilla Firefox were used to test the Canine Web application. In all browsers, all the features worked well. It is very important to check the website for several browsers, as different clients use various web browsers. The following chart shows the test status performed on various web browsers.

Browser	Version	Status	Comments
Google Chrome	90.0	Pass	Shows all pages that it intends to display and Styles, Page's performance is not changed.
Mozilla Firefox	88.0	Pass	Displays all pages without a design

			differentiation. Slightly different from the fonts used in other browsers, but no change to performance.
Safari	14.0.3	Pass	Shows all pages without any change.

Table 5: Compatibility Testing

React native and CSS 3 are used for the design of the website. Only their browsers are tested as passed.

2.10 Chapter Summary

The above chapter discussed on testing the CANINE and reporting the test outcomes. Accuracy testing, Performance testing, Usability testing and Compatibility testing were done to test the functional requirements. The following chapter will discuss about the Evaluation of the system.

3. Evaluation

3.1 Chapter Overview

This chapter explain about how the application's mastery and target crowd assessed the different parts of the framework like significance, ease of use. Following their assessment, the creators' self-assessment is additionally given. On the off chance that the underlying goals of the framework were met, are additionally archived.

3.2 Evaluation Methods

Nobody knows about the area of this task about how it functions at all. The assessment period has become the hardest period due to the pandemic situation (COVID - 19). So, we chose to share the application guidelines and the functionalities by means of video to the chose evaluators. Our target audience and everyday citizens.

The assessment couldn't go through our fundamental objective crowd on the grounds that of the pandemic circumstance yet from the specialized side we were gone through certain engineers and students. The assessment results are centered around amount and the quality also. According to our assumption didn't fall flat. The Evaluation results have the quality and the amount. The accompanying measures will additionally clarify what we have done in this part.

3.3 Quantitative evaluation

Quantitative evaluation performed to getting accuracy level, other evaluation metrices was arranging the confusion matrix. Every step was unreal within the code and graphs arranged using the matplotlib library to analyze the result for prediction. The further details or result and testing about prediction under the testing chapter 2.

3.4 Qualitative evaluation

Person	Feedback
Thanex Dog owner/lover (End User)	The website is really amazing with dazzling and simple interface. I could get some idea about my dog's prolonged disease and shows the output quickly. It would be nice if more features get added to the website. I am extremely happy with the product.

K.Premnath Techical Lead Smartzi (Pvt) Ltd (domain expert)	It's a great business idea for the online market and technically well planned. The requirement level seems to be intermediate and approach for diseases identification is fine but need more data to train the model to increase the accuracy. I admire that the team added the requirement as reproduction strategy which will help the people directly.
N.Yanarthanan Senior Engineer Smartzi (Pvt) Ltd (industry expert)	Much appreciated. brilliant idea and required RND for the industry. I suggest improving the breed finding technique as it more depend on multiple factors.

Table 6: Evaluation Feedback

3.5 Self evaluation

The prototype pointed the concern details into the problem domain chapter under SRS. The project prototype has built that can be used in any domains, and modifications. The project will be changed according to certain things order to achieve. For our second year SDGP project, the scope of project seems like satisfactory.

The languages, libraries and other methodologies were select correctly for this canine project. The implementation finished after using and handle of techniques and tools for the project. The algorithm helped to identify a problem, and accuracy part. In the due time constrains, we only implemented important features were implemented in the prototype. The implemented system works well and functionally in testing part result prove that.

3.6 Chapter Summary

This part was about the few assessments by specialists of various kinds. It began with the assessment procedure and approach. The assessment standards were reported. The evaluators chose and their reasonableness to assess the venture was introduced. The assessment brings about terms of idea, scope, plan, engineering, execution, arrangement, model and restrictions were assessed. The useful prerequisites fruition lastly the creator’s self-assessment was introduced.

4. Conclusion

4.1 Chapter Overview

The previous chapter discussed the Evaluation of the project. This is conclusion chapter that include the summary of the project. Achievement of Aim and Objectives were discussed in this chapter. Legal, social, professional and ethical issues also documented.

4.2 Achievements of aims and objectives

4.2.1 Aim

Aim of this project is implement a web application to identify the dog breed and identify the diseases from the symptoms. Most of the dog lovers do not know the dog breed what they have. Identifying the dog diseases also another problem. This project will help to solve those problems.

4.2.2 Objectives

ID	Objective Description	Status
OBJ1	**Project Introduction** • The Problem background, aim, objectives, scope, Resource requirements and features of the prototype of this project are defined.	Completed
OBJ2	**Literature Review** • Existing systems, Research on Approaches and Techniques are defined.	Completed
OBJ3	**Project Management** • Process Model, Project Management Method, Data Gathering Method and Gannt Chart are defined.	Completed
OBJ4	**System Requirements Specification** • Functional and non-functional requirements, Requirement Gathering, Use Case diagrams are defined.	Completed
OBJ5	**Design** • High-Level architecture Diagram, Class Diagram, Sequence Diagram, Activity Diagram are defined.	Completed
OBJ6	**Implementation** • Implementation of the data science component, backend component and frontend component are implemented.	Completed

OBJ6	**Testing** • Unit testing, Performance testing, Usability testing, Compatibility testing are conducted.	Completed

Table 7: Objectives

4.3 Legal, social, ethical and professional issues

- **Legal issues**

When developing the CANINE, high regard was given to data protections law. The terms and conditions published on Kaggle were checked when using the Kaggle dataset for the CANINE Project. Dataset was not used for any illegal activities and not misused.

- **Social issues**
  There will be no social problems since this project is entirely focused on dog diseases and their remedies. The project was developed in only English Language, which could have an effect on people those who don't know English language.

- **Ethical issues**
  The dataset for this project was obtained from a government-approved veterinarian through a questionnaire and the Kaggle platform. The dataset was given by the doctor based on his medical history and gave the permission to use the dataset for CANINE project.

- **Professional issues**
  Diseases Dataset we gathered from Doctor and we informed about how to use this and why we need this dataset for our project.

4.4 Limitations of the research

- This project is only support for English Language.
- Testing tools limitation.

4.5 Future enhancements

- **Multilanguage support** - This prototype is only support for English Language. In the future, there is a chance to include Multilanguage support to our prototype. This might be help for those who don't understand English.

- **Add some features**-Finding the nearest veterinary hospital, Food tips, Fitness Activities will be included in future.

- Identify the dog breed from DNA test report.

4.6 Concluding remarks

This project presents a solution for identify the dog breed and predict the diseases from the symptoms. CANINE has been evaluated and documented with outcomes. New tools, technologies and techniques were discovered by doing this project.