CSCI 6660 - ARTIFICIAL INTELLIGENCE

FINAL TERM PROJECT

CHATBOT FOR BUSINESS

TEAM MEMBERS:

KUNAL DHANAITKAR

SAMARA SIMHA REDDY CHINNAGOUNI

SUBMITTED TO:

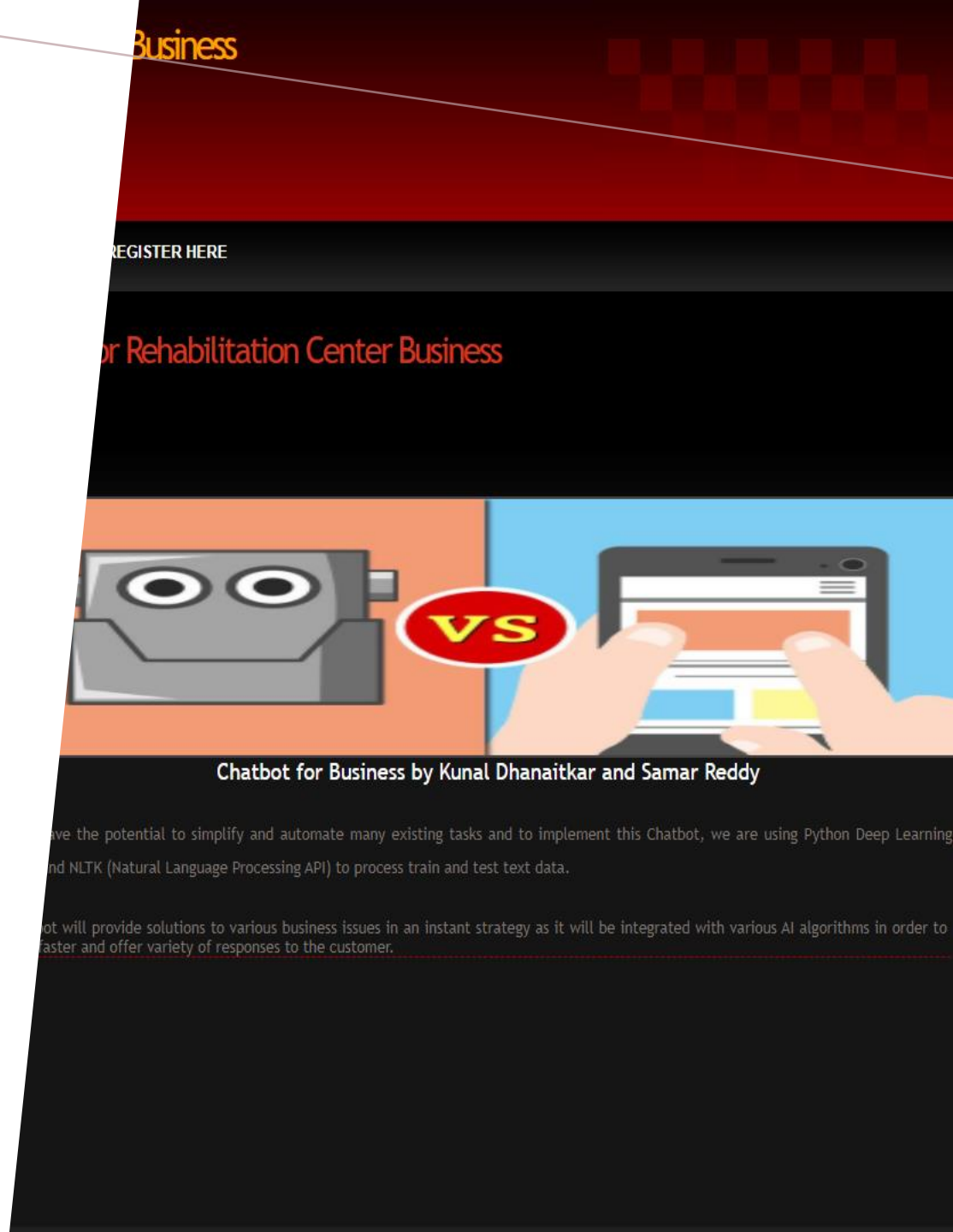
DR. SHIVANJALI KHARE

OVERVIEW

- Introduction
- Project Requirement
- Python Libraries Requirements
- Building the AI Model for the Chatbot
- Functioning of Chatbot
- User Interaction with Chatbot

INTRODUCTION

- Chatbots have the potential to simplify and automate many existing tasks and to implement this Chatbot, we are using Python Deep Learning Neural Networks and NLTK (Natural Language Processing API) to process train and test text data.



PROJECT REQUIREMENTS

OPERATING SYSTEM – WINDOWS

CODING LANGUAGE – PYTHON

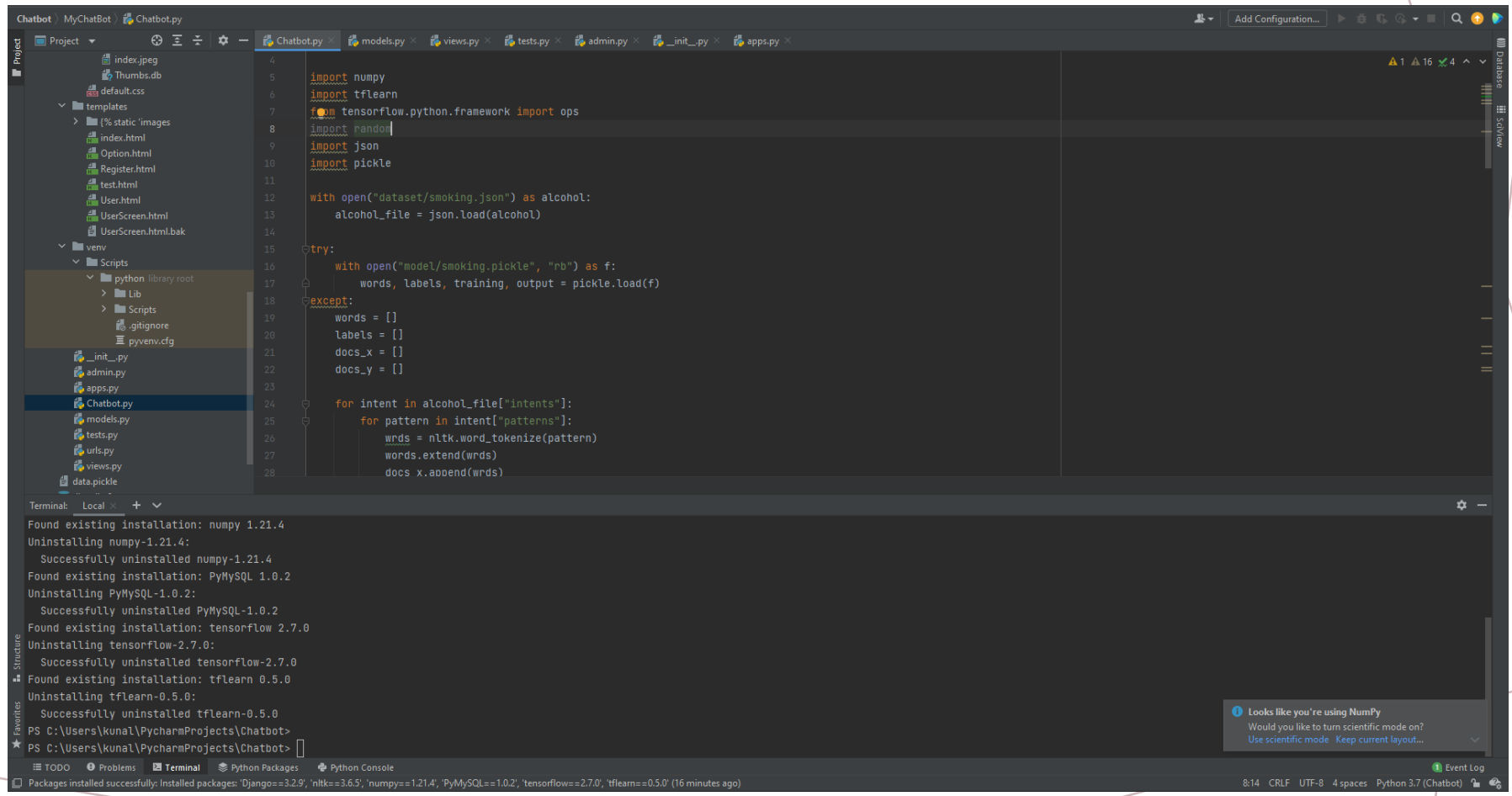
FRAMEWORK – DJANGO

DATABASE – MYSQL

IDE - PYCHARM

PYTHON LIBRARIES USED TO IMPLEMENT CHATBOT:

- Pandas
- Numpy
- Matpltolib
- Keras
- Tensorflow
- NLTK
- Scopy
- Sklearn



BUILDING THE AI FOR CHATBOT

- **Training Data:**

For dataset we are using JSON file. The JSON file is creating a bunch of messages that the user is likely to type in and mapping them to a group of appropriate responses. The tag on each dictionary in the file indicates the group that each message belongs too. With this data we will train a neural network to take a sentence of words and classify it as one of the tags in our file. Then we can simply take a response from those groups and display that to the user. The more tags, responses, and patterns you provide to the chatbot the better and more complex it will be.

```
import json
import pickle

with open("dataset/smoking.json") as alcohol:
    alcohol_file = json.load(alcohol)


try:
    with open("model/smoking.pickle", "rb") as f:
        words, labels, training, output = pickle.load(f)
```

BUILDING THE AI FOR CHATBOT

- **Extracting Data**

We need all the patterns and which class/tag they belong to. We also want a list of all the unique words in our patterns. For each pattern we will turn it into a list of words using nltk.word_tokenizer, rather than having them as strings. We will then add each pattern into our docs_x list and its associated tag into the docs_y list.

BUILDING THE AI FOR CHATBOT

- **Word Stemming**

Stemming a word is attempting to find the root of the word. We will use this process of stemming words to reduce the vocabulary of our model and attempt to find the more general meaning behind sentences.

```
for x, doc in enumerate(docs_x):
    bag = []

    wrds = [stemmer.stem(w.lower()) for w in doc]

    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)

    output_row = out_empty[:]
    output_row[labels.index(docs_y[x])] = 1

    training.append(bag)
    output.append(output_row)


training = numpy.array(training)
output = numpy.array(output)

except
```

BUILDING THE AI FOR CHATBOT

- **Bag of Words**

Neural networks and machine learning algorithms require numerical input. So out list of strings won't cut it. We need some way to represent our sentences with numbers and this is where a bag of words comes in.

```
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

BUILDING THE AI FOR CHATBOT

- **Developing a Model**

Now that we have preprocessed all our data, we are ready to start creating and training a model. For our purposes we will use a standard feed-forward neural network with two hidden layers.

The goal of our network will be to look at a bag of words and give a class that they belong too (one of our tags from the JSON file). We will start by defining the architecture of our model.

```
print(str(len(training[0]))+" "+str(len(output[0])))
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)

model = tflearn.DNN(net)

#try:
#    model.load("model/drugsmodel.tflearn")
#except:
model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
model.save("model/smokingmodel.tflearn")
bag_of_words()
```

FUNCTIONING OF CHATBOT

- Download the Code.
- pip install virtualenv on your Desktop using CLI.
- Open the source directory of the Project.
- Create a virtual enviornment using the command

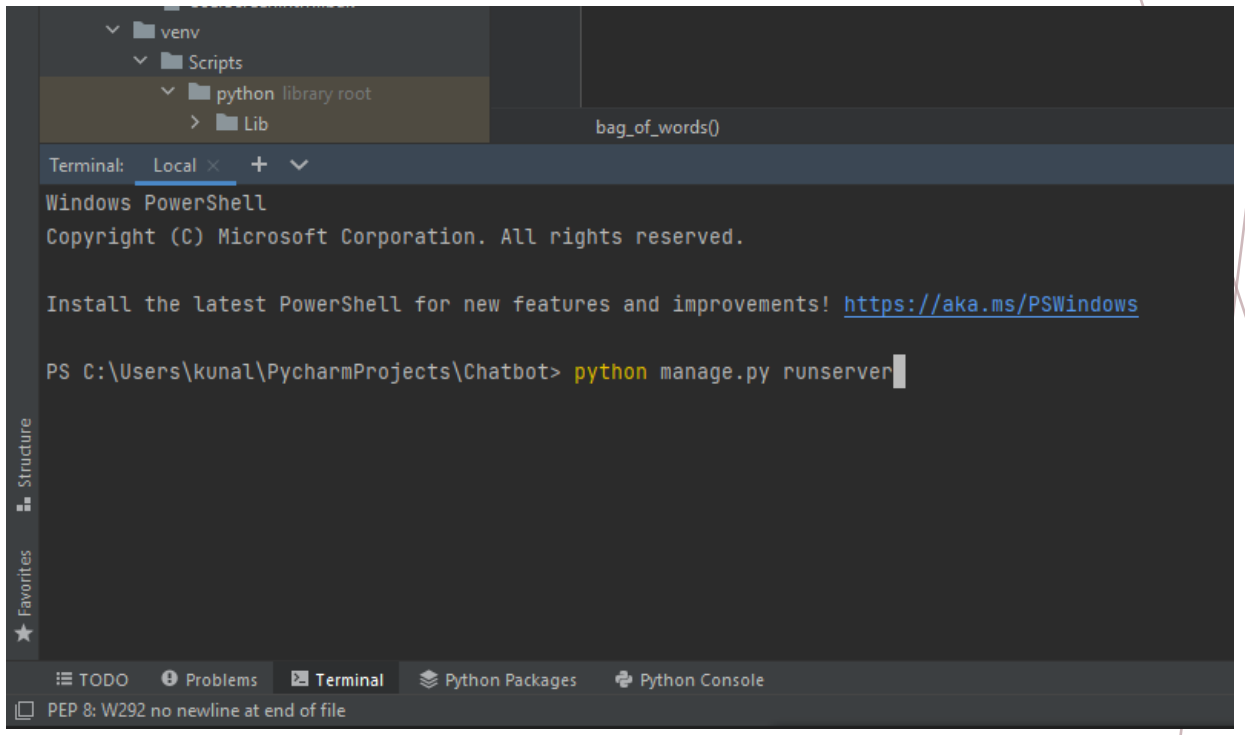
  virtualenv newenv

- Install all the dependecies using the command

  pip install -r requirements.txt

- Execute the Code using the following command

  python manage.py runserver

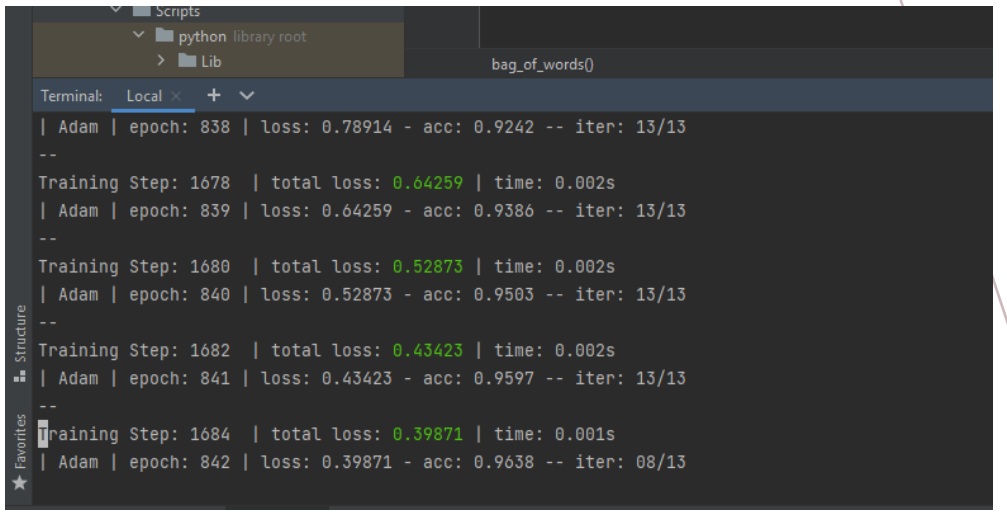


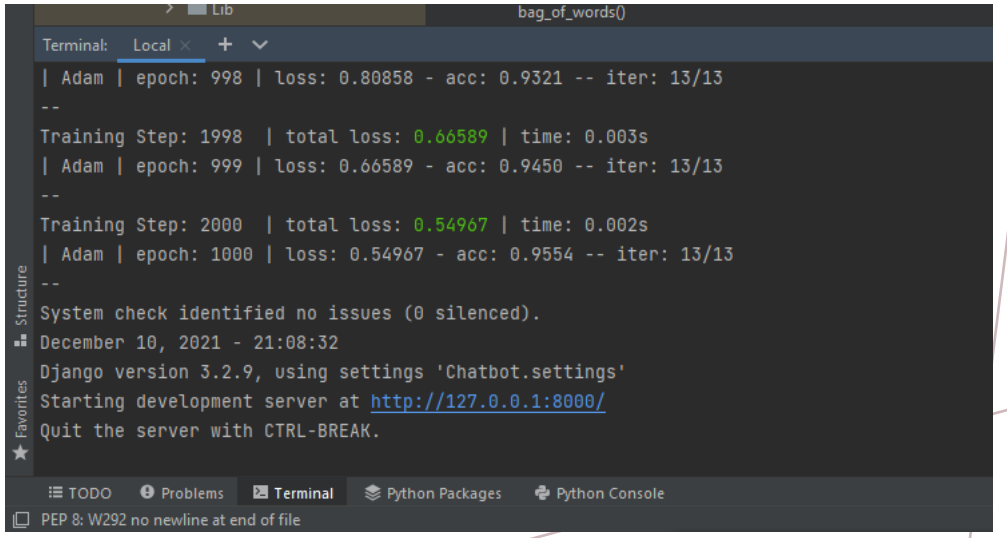
FUNCTIONING OF CHATBOT

- **Loading the Trained Model and Passing it to the GUI**

To predict the sentences and get a response from the user, we will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve us a random response from the list of responses.

We import the necessary packages and load the pickle files which we have created when we trained our model.

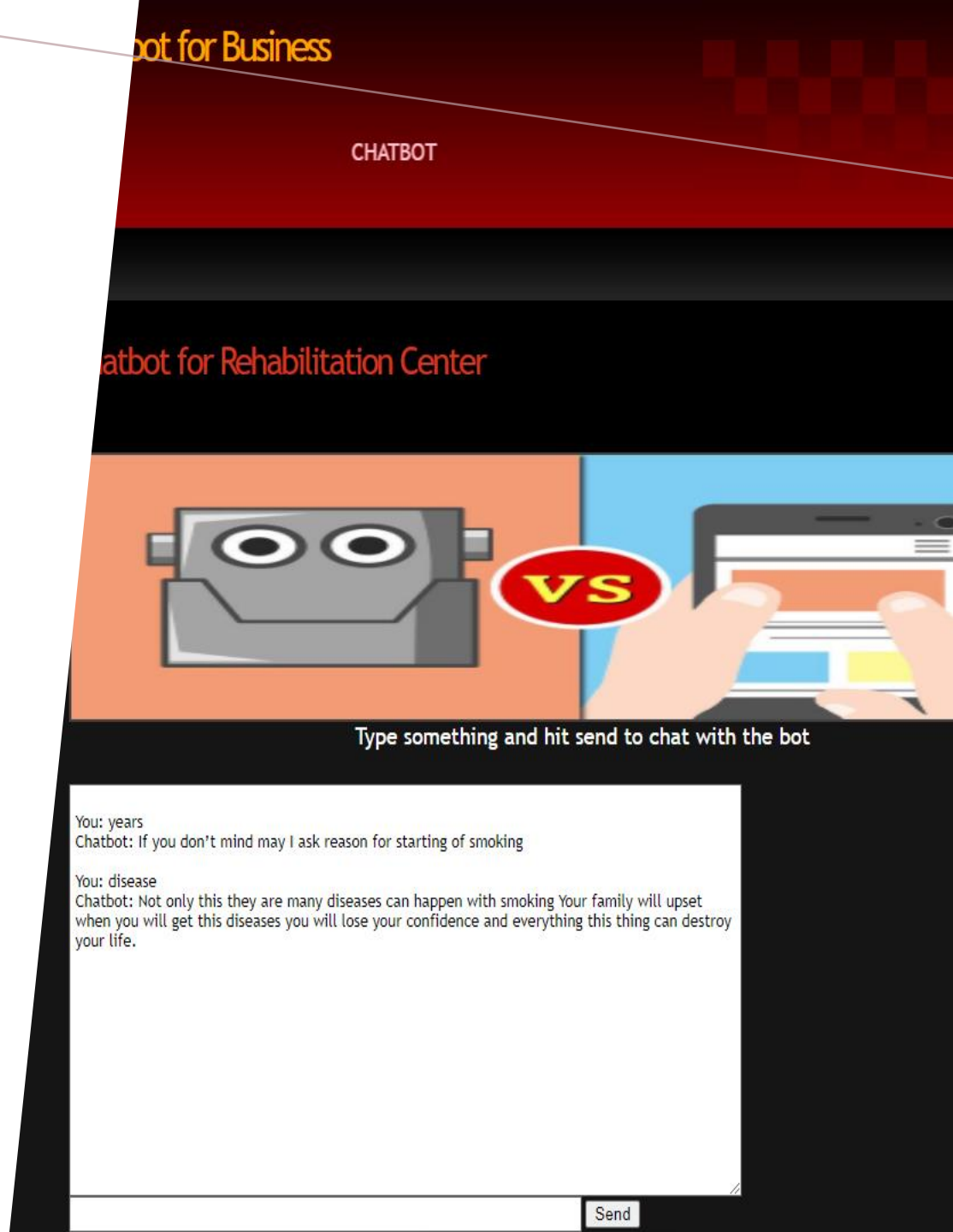




USER INTERACTION WITH CHATBOT

- Now that the Project has been loaded by the Browser using the server link provide by our Django project. We can chat with the bot after creating a new registration login and choosing a bot to chat with.

- Currently, we were able to train only 3 models. Due to time constraints, this release can be termed as an Alpha Release.



THANK YOU