# 

# PROJECT REPORT

# On

**iBOT**

Submitted in partial fulfilment for the award of

**BACHELOR OF TECHNOLOGY**

**In**

**Computer Science & Engineering (2022)**

By

**AAMNA AKHTAR (1801012003)**

**ARSHIYA DILSHAD (1801012026)**

**ATHAR ZAHID USMANI (1801012032)**

Under the Guidance

Of

**Mr. Aftab Yaseen**

(Assistant professor)

****

**INTEGRAL UNIVERSITY, LUCKNOW (INDIA)**

**COMPUTER SCIENCE & ENGINEERING**

**INTEGRAL UNIVERSITY,**

**LUCKNOW**

**CERTIFICATE**

Certified that the project entitled **“iBot”** submitted by Aamna Akhtar (1801012003), Arshiya Dilshad (1801012026), Athar Zahid Usmani (1801012032) in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering, **Integral University, Lucknow (INDIA)** is a record of students owns work carried under our supervision and guidance. The project report embodies results of original work and studies carried out by students and the contents do not form the basis of the award of any other degree to the candidate or to anybody else.

(Signature of Project Guide) (Signature of HOD)

MR. AFTAB YASEEN MRS. KAVITA AGARWAL

(Assistant Professor) (Head of Department)

**DECLARATION**

We hereby declare that the project entitled “**iBot”** submitted by us in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science &Engineering of Integral University, Lucknow, is record of our own work carried under the supervision and guidance of **Mr. AFTAB YASEEN** (Assistance Professor).

To the best of our knowledge this project has not been submitted to Integral University, Lucknow or any other University or Institute for the award of any degree.

### **AAMNA AKHTAR ARSHIYA DILSHAD ATHAR ZAHID USMANI**

**1800102657 1800100420 1800100606**

**ACKNOWLEDGEMENT**

Thanking everybody is such a crucial task and without performing that, any good work remains incomplete.

Every constructive work needs some fuel as its driving force. The fuel in our case was every little stone of help and appreciation that we received from our **Project Supervisor- Mr. Aftab Yaseen.** We sincerely thank him for his guidance and constant motivation during the development phase of this project. Without his constant support and motivation this project would never had been come to its completion. We also wish to express our gratitude to our friends who helped us with resolving the problems and difficulties we faced building the project. We also thank Integral University for providing me the opportunity to embark on this project, which will gradually help us in becoming a better human resource as a Computer Science professional and will help us in our future endeavours.

Thanking you

Aamna Akhtar

Arshiya Dilshad

Athar Zahid Usmani

**PREFACE**

It is a great opportunity for us to have the Bachelor of Technology in Computer Science & Engineering in Integral University, Lucknow. In the partial fulfilment of the requirements for the award of this degree we are submitting a project report on **“iBot”**. Subject to the limitation of time efforts and resources every possible attempt has been made to complete the project successfully.

The whole project has been divided into five chapters.

* Introduction
* Feasibility Study
* Project Requirement/Proposed methodology
* Result Analysis and Future Work
* Conclusion

**ABSTRACT-RESUME OF THE PROJECT**

**iBot** is the name of our chatbot. A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. These days, chatbots are used almost everywhere, from customer interaction to marketing on social network sites and instantly messaging the client. A chatbot is a computer program that simulates and processes human conversation either written or spoken, allowing humans to interact with digital devices as if they were communicating with a real person. Chatbots can be as simple as rudimentary programs that answer a simple query with a single-line response, or as sophisticated as digital assistants that learn and evolve to deliver increasing levels of personalization as they gather and process information.

Chatbots allow businesses to connect with customers in a personal way without the expense of human representatives. For example, many of the questions or issues customers have been common and easily answered. That’s why companies create FAQs and troubleshooting guides. Chatbots provide a personal alternative to a written FAQ or guide and can even triage questions, including handing off a customer issue to a live person if the issue becomes too complex for the chatbot to resolve. Chatbots have become popular as a time and money saver for businesses and an added convenience for customers.

Chatbots are frequently used to improve the IT service management experience, which delves towards self-service and automating processes offered to internal staff. With an intelligent chatbot, common tasks such as password updates, system status, outage alerts, and knowledge management can be readily automated and made available 24/7, while broadening access to commonly used voice and text based conversational interfaces.

**PROJECT PROPOSAL**

**“iBot”** is the name of our chatbot. A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. A chatbot is a computer program that simulates and processes human conversation either written or spoken, allowing humans to interact with digital devices as if they were communicating with a real person.

**OBJECTIVES AND AIMS: -**

The project will serve the following Objectives: -

* Our chat will solve the queries of the people. It can be as simple as rudimentary programs that answer a simple query with a single-line response
* With an intelligent chatbot, common tasks such as password updates, system status, outage alerts, and knowledge management can be readily automated and made available 24/7,
* It improves the IT service management experience.
* Money saver for businesses and an added convenience for customers.
* Chatbots allow businesses to connect with customers in a personal way without the expense of human representatives

**RELEVENCE: -**

* **Fast and Simple** – It will answer the query of the customer in seconds
* **Everything is stored in the database** - Since everything will be stored in the database, you will have a proper record of it and it can be accessed whenever needed.
* **Automation** – Answer will be automated which saves time.
* **Cost** - Chatbots have become popular as a time and money saver for businesses and an added convenience for customers.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **TITLE** | **PAGE NO** |
| 1. | COVER PAGE | i |
| 2. | CERTIFICATE | ii |
| 3. | DECLARATION | iii |
| 4. | ACKNOWLEDGEMENT | iv |
| 5. | PREFACE | v |
| 6. | ABSTRACT- RESUME OF PROJECT | vi |
| 7. | PROJECT PROPOSAL | vii |
| 8. | TABLE OF CONTENTS | viii |
| 9. | INTRODUCTION | 1-7 |
| 10. | FEASIBILITY STUDY | 8 |
| 11. | REQUIREMENT ANALYSIS | 9-18 |
| 12. | PROJECT DEVELOPMENT | 19-42 |
| 13. | IMPLEMENTATION | 43-50 |
| 14. | RESULT ANALYSIS AND FUTURE WORKS | 51-55 |
| 15. | CONCLUSION | 56 |
| 16 | LIST OF FIGURES | ix |
| 17. | REFERENCES AND BIBLIOGRAPHY | x |

**CHAPTER -1**

**INTRODUCTION**

**“iBot”** is the integrated chatbot, aims to solve the queries of the people. simulates and processes human conversation either written or spoken, allowing humans to interact with digital devices as if they were communicating with a real person.

**1.1 CHATBOT**

Chatbots is also called chatterbots. It is a form of artificial intelligence (AI) used in messaging apps. This tool helps add convenience for customers they are automated programs that interact with customers like a human would and cost little to nothing to engage with.



Figure 1 - Chatbot



Figure 2 - illustration

**1.2 HOW CHATBOT WORKS**

The most typical chatbot interaction occurs on a business site. These customer service bots usually pop up after a human user navigates around a site for a few minutes or exhibits behaviors that show that they have become “lost” or are having trouble connecting with the information they need.

Once the chatbot window presents itself, the user can enter their question in plain, syntactical English. [The bot’s language recognition functions break down the question](https://www.sofbang.com/utility-chatbot/) and, at the speed of light, compares the query to its data bank of previously asked questions to look for ways customers have achieved satisfying results in similar situations.

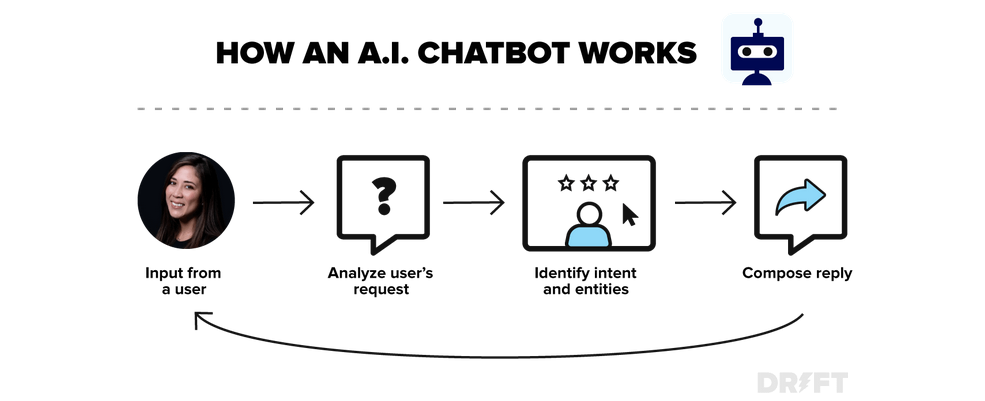


Figure 3 – How a chatbot works

**1.3 STRUCTURE OF THE PROJECT**

Our project "iBot" is based on the concept of NLP and Deep Neural Network. Natural Language Processing (NLP) is concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. A Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers.

Natural language processing strives to build machines that understand and respond to text or voice data and respond with text or speech of their own in much the same way humans do.

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

Deep learning also known as deep structured learning is part of a broader family of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) methods based on [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) with [representation learning](https://en.wikipedia.org/wiki/Representation_learning). Learning can be [supervised](https://en.wikipedia.org/wiki/Supervised_learning), [semi-supervised](https://en.wikipedia.org/wiki/Semi-supervised_learning) or [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning).

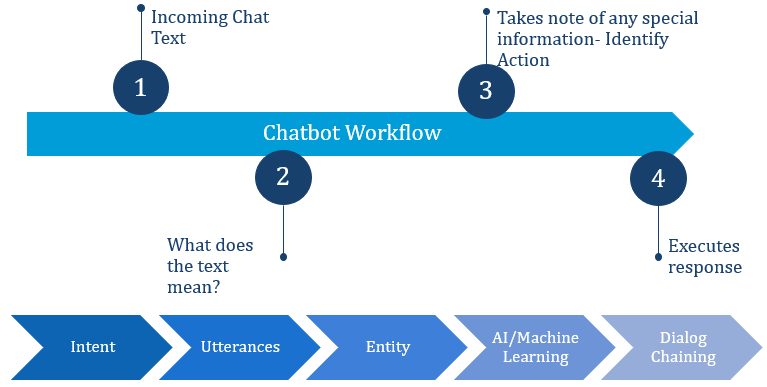
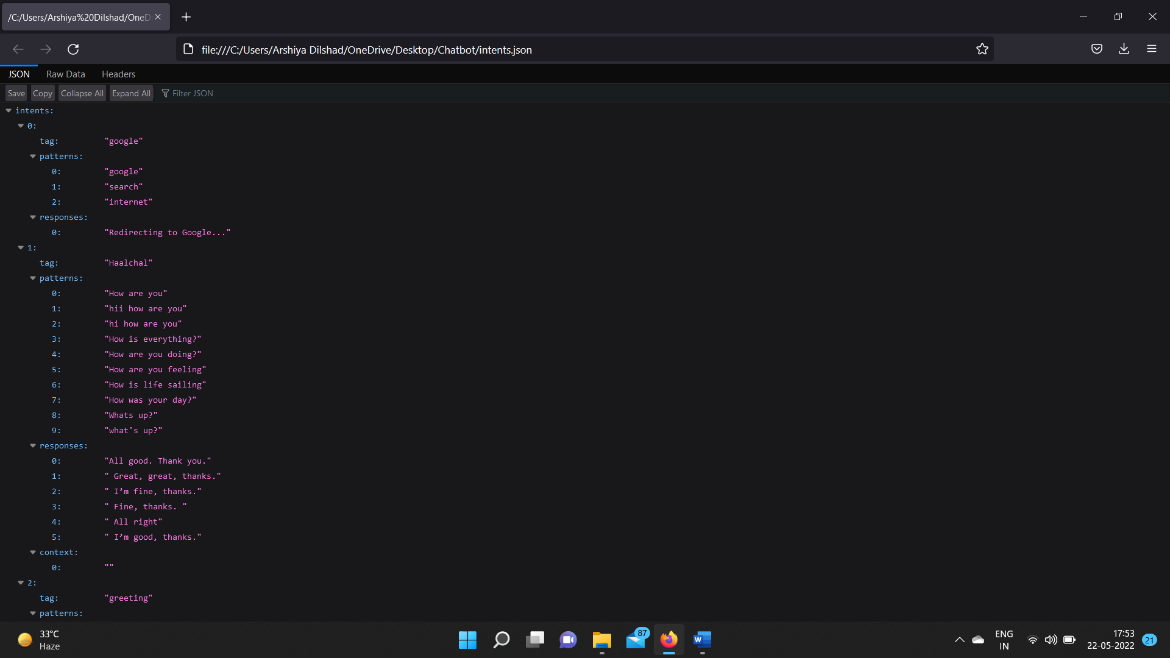


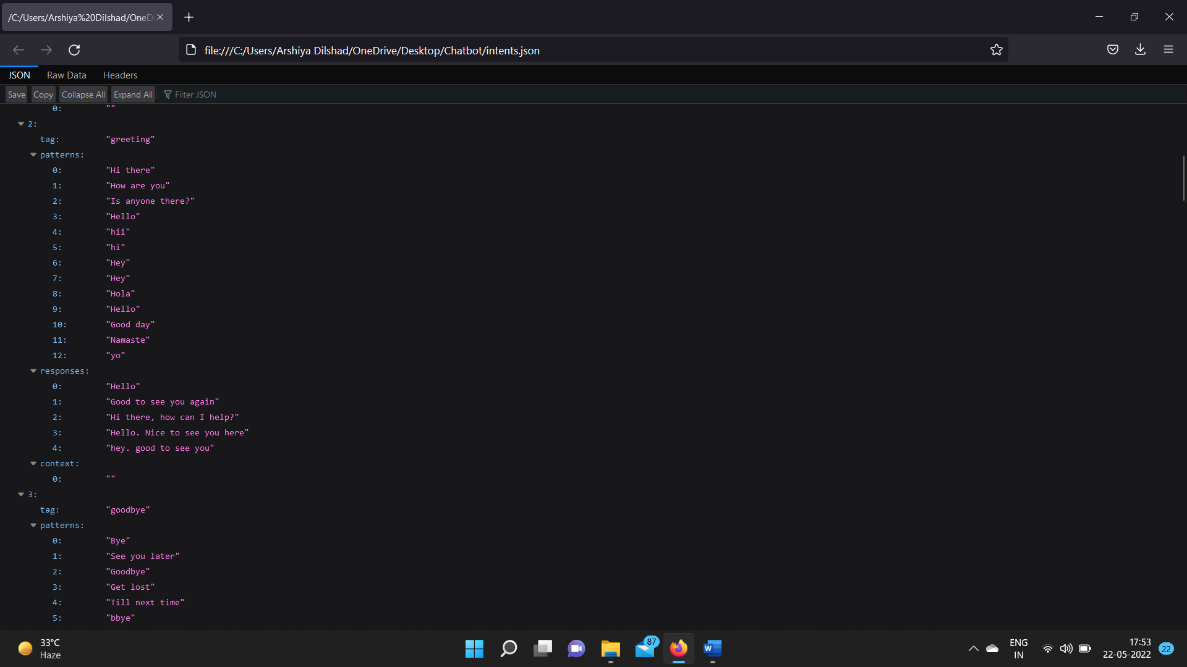
Figure 4 – Chatbot workflow

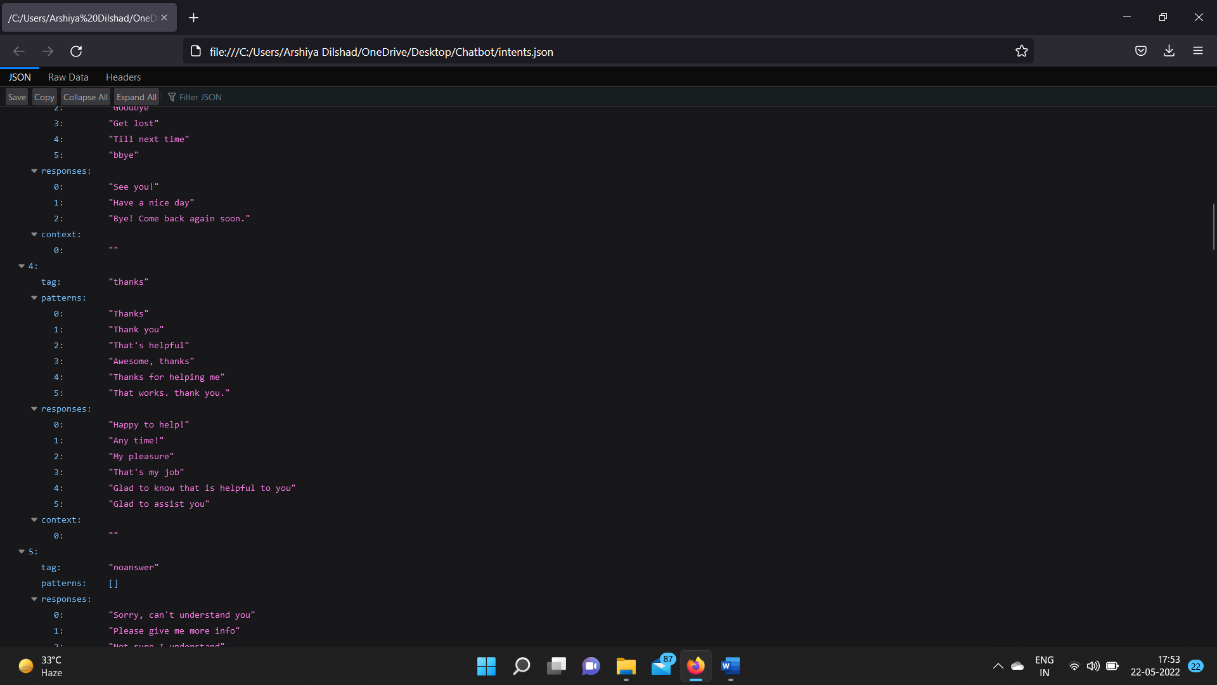
**DATASET OF THE PROJECT**

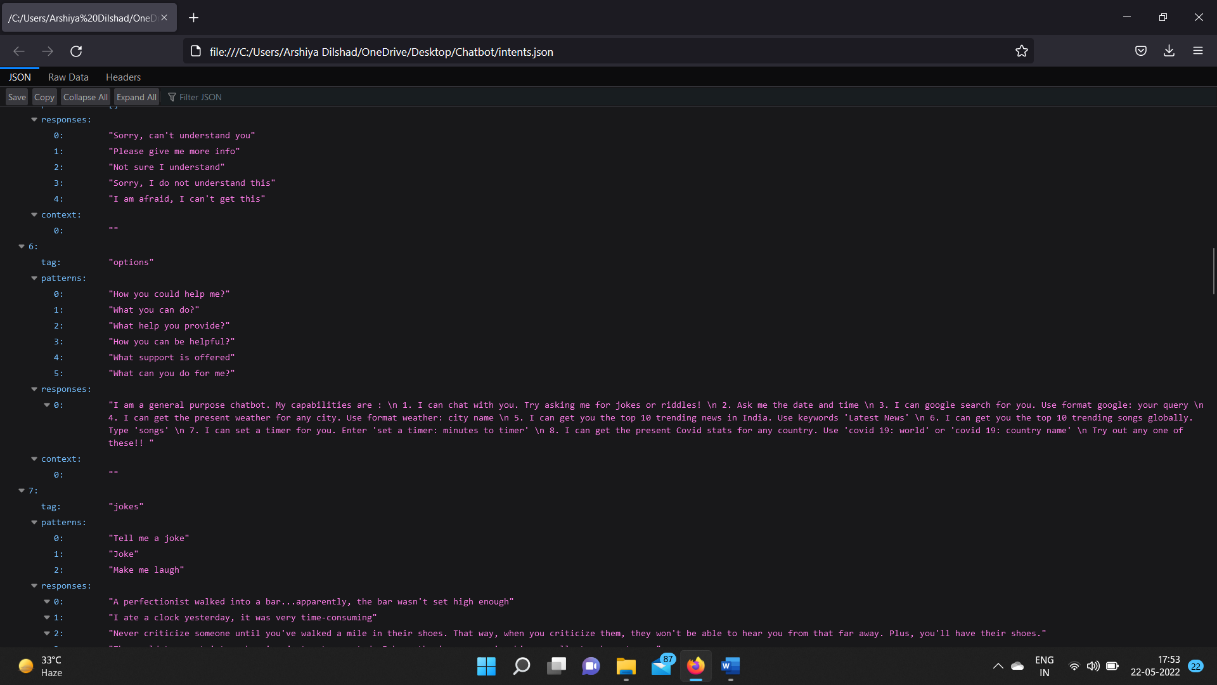
To build a perfect bot, all we needed was a large amount of data. But finding the right dataset for our project was quite a challenging task. Finally, we found an intents.json file from Kaggle. The iBot dataset is a JSON file that has disparate tags like welcome message, goodbye, greetings, covid related information, jokes, riddles etc. Every tag has a list of patterns that a user can ask, and the iBot will respond according to that pattern. The dataset is perfect for understanding how iBot data works.

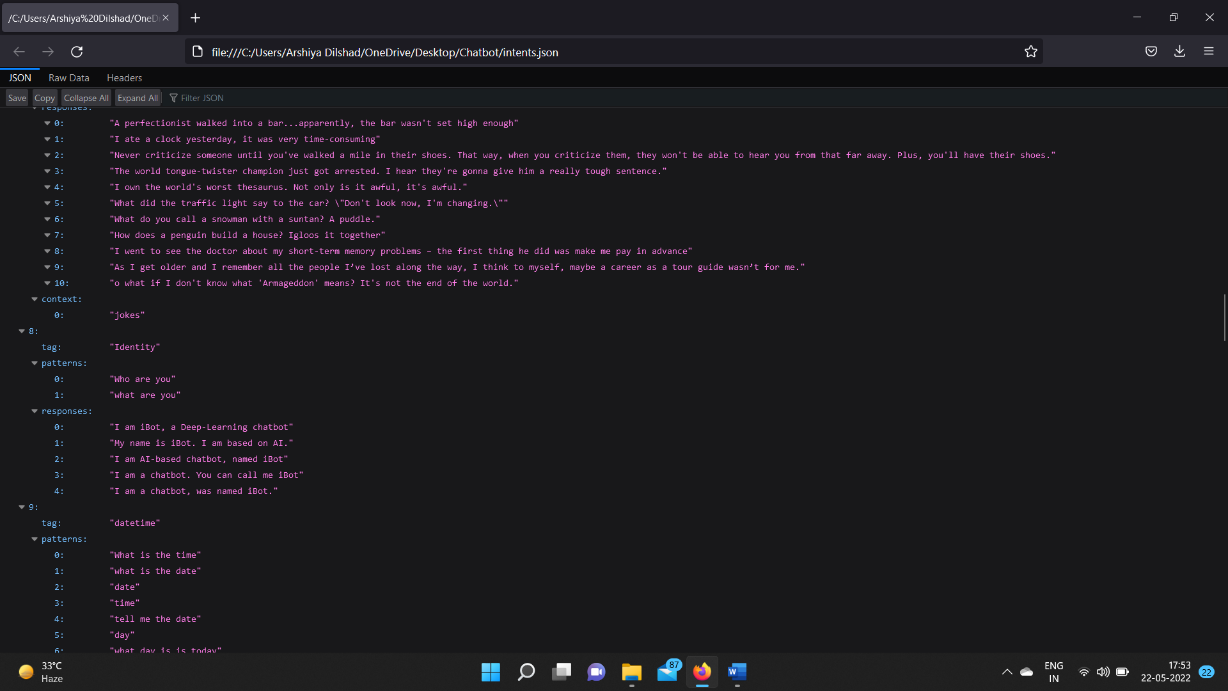
**1.4 Dataset which is used in our chatbot: -**

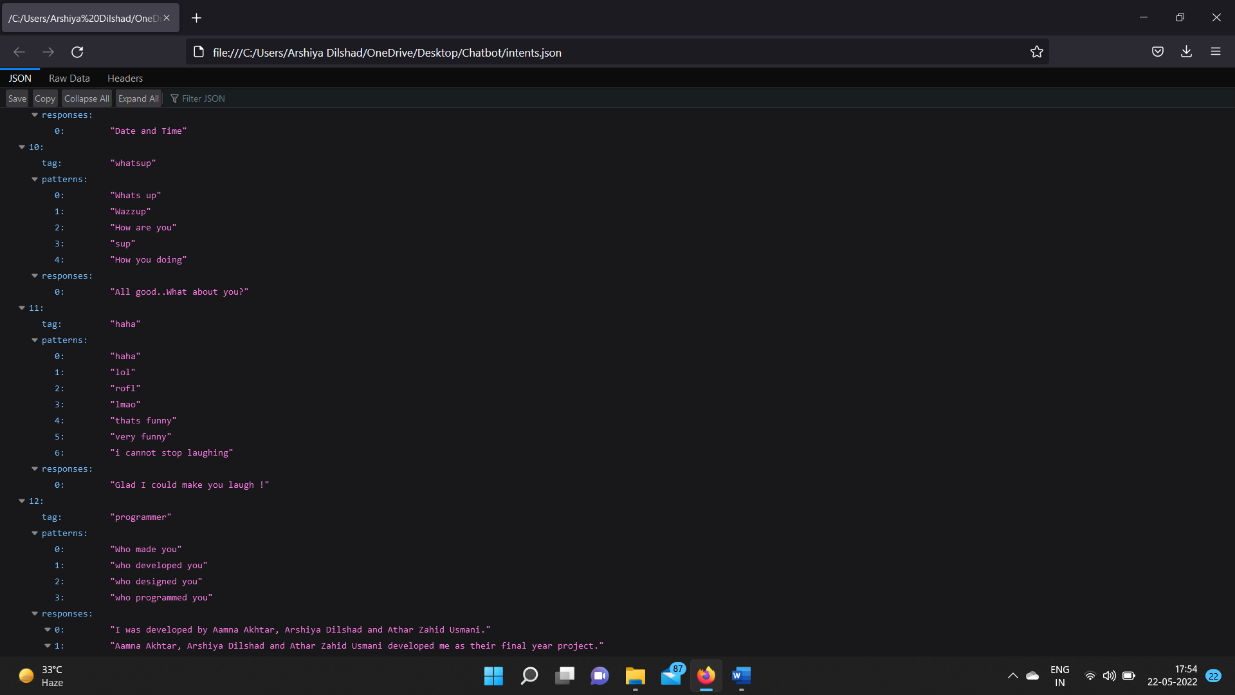


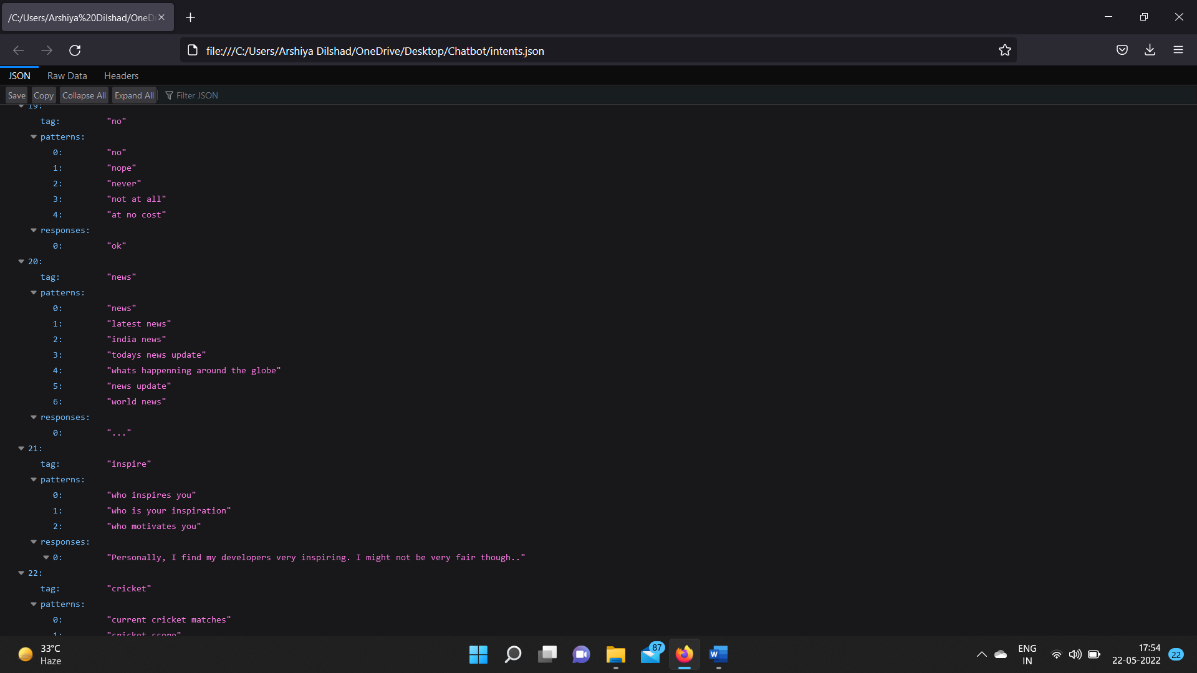


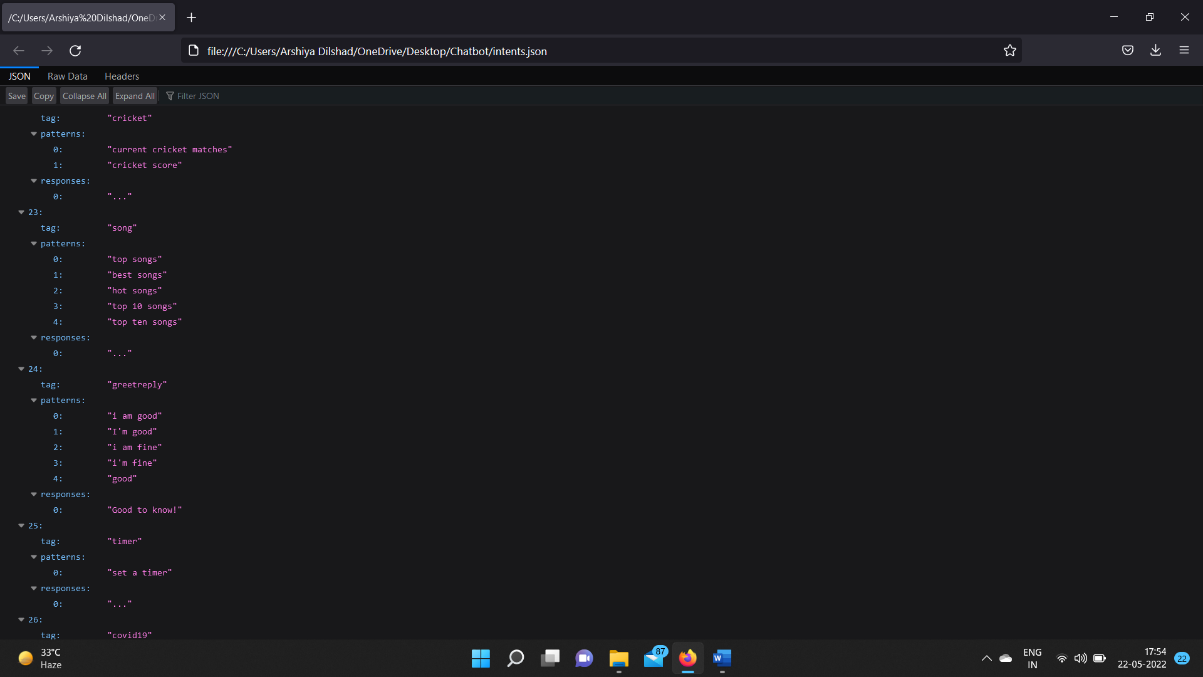


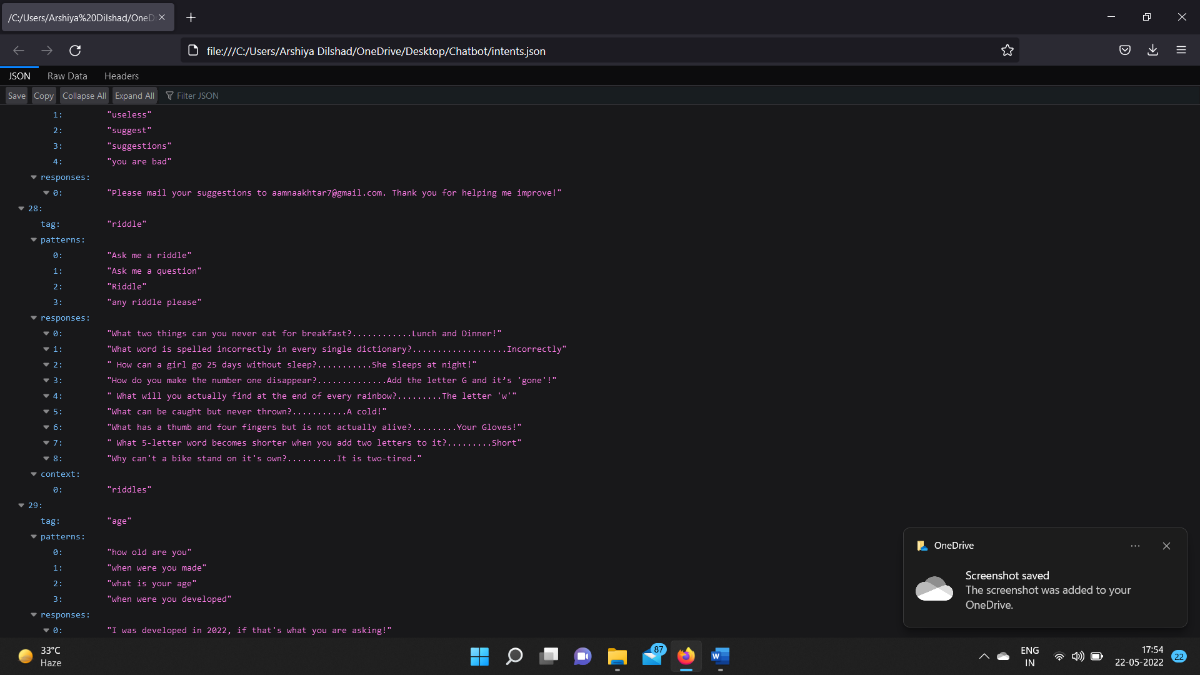












**1.5 DATA FLOW DIAGRAM**

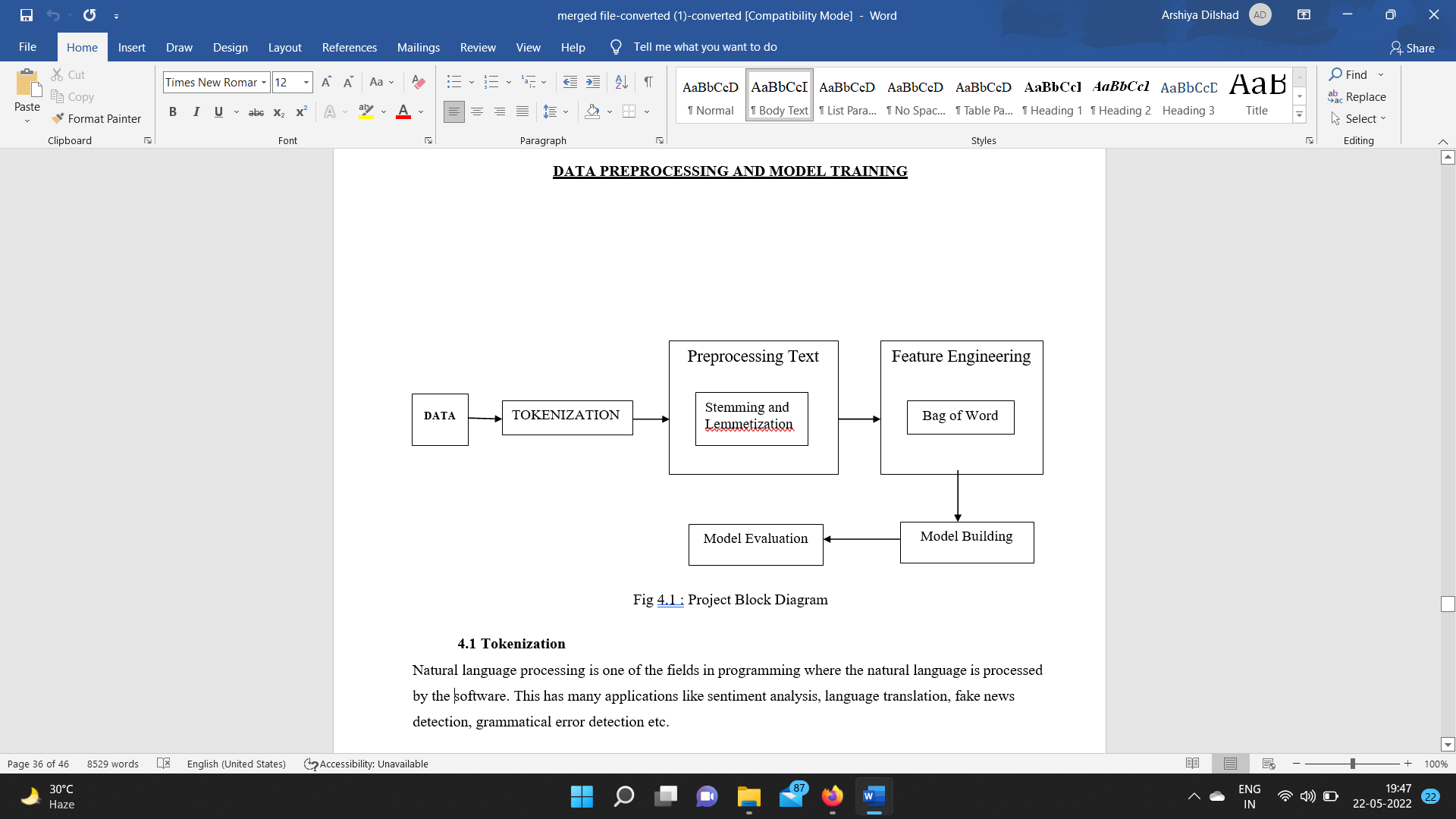
****

Figure 5 – Data flow diagram

**Chapter -2**

**FEASIBLITY STUDY**

This study helps us to analyze whether our project is feasible or not. Our Chatbot i.e iBot is a completely feasible project because it requirements are a list of easily available and daily used technologies. No extra cost is required to maintain or develop this Bot.

**2.1 GENERAL DESCRIPTION**

* **Purpose**

The purpose of our AI Chatbot is to answer the query of the people regrading particular thing. It is a form of artificial intelligence (AI) used in messaging apps.

* **Use**

Chatbots allow businesses to connect with customers in a personal way without the expense of human representatives. This tool helps add convenience for customers they are automated programs that interact with customers like a human would and cost little to nothing to engage with.

* **Objective**

Chatbots are frequently used to improve the IT service management experience, which delves towards self-service and automating processes offered to internal staff.

* **Overview**

With an intelligent chatbot, common tasks such as password updates, system status, outage alerts, and knowledge management can be readily automated and made available 24/7.

**Chapter -3**

**REQUIREMENTS ANALYSIS**

This section of the report brings us to the part of understanding the requirement analysis phase of the project. It is the stage where Software Requirement Specification (SRS) document is created which includes the details of all the functional and non-functional requirements.

**3.1 Software requirements: -**

1. Anaconda PowerShell Prompt
2. Anaconda Navigator
3. Spyder

**3.2 Frontend Languages: -**

1. HTML
2. JavaScript
3. CSS
4. Bootstrap

**3.3 Backend Languages: -** Python

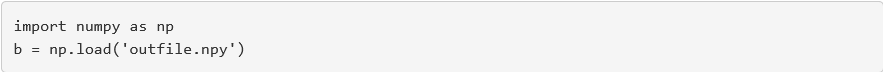
**3.4 Required modules**

1. Following modules needs to be installed for it to work properly:
2. Numpy
3. Tensorflow
4. Regular Expression
5. Time

**Numpy:**

It is a Python library that provides a multidimensional array object, various derived objects. NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

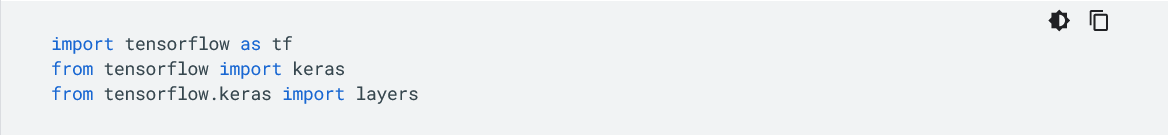
It can be installed using “pip install numpy”



**Tensorflow:**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of Tenserflow. We have used keras model for sequential model.

A Sequential model is appropriate for **a plain stack of layers** where each layer has **exactly one input tensor and one output tensor**.



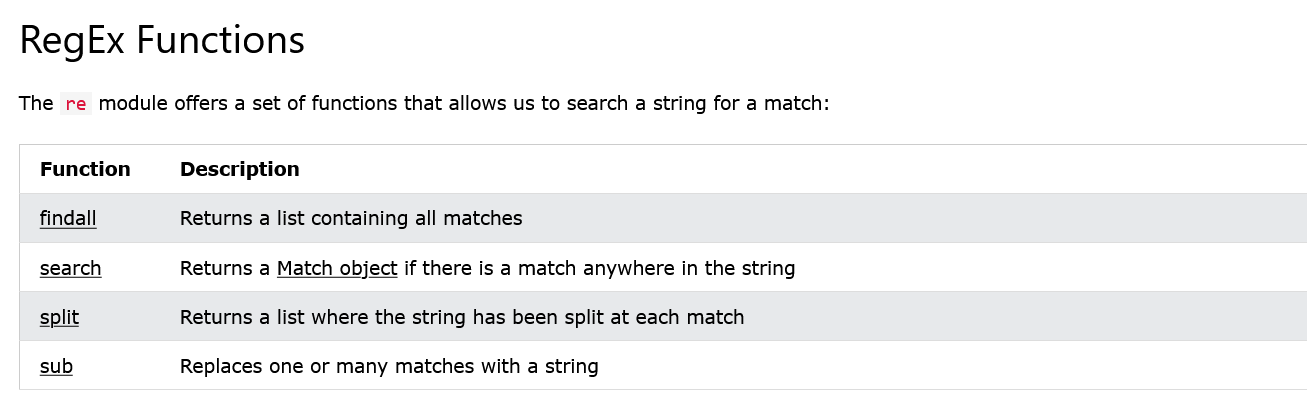
**Regular Expression (RE):**

Regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

Regular expressions are a powerful language for matching text patterns. This page gives a basic introduction to regular expressions themselves sufficient for our Python exercises and shows how regular expressions work in Python. The Python "re" module provides regular expression support

Regular expressions can be concatenated to form new regular expressions; if *A* and *B* are both regular expressions, then *AB* is also a regular expression. In general, if a string *p* matches *A* and another string *q* matches *B*, the string *pq* will match AB. This holds unless *A* or *B* contain low precedence operations; boundary conditions between *A* and *B*; or have numbered group references. Thus, complex expressions can easily be constructed from simpler primitive expressions like the ones described here.





**Time:**

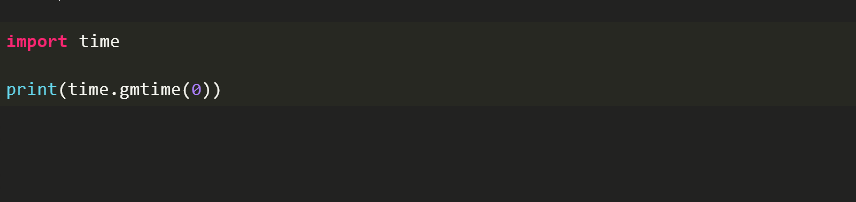
As the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module we need to import it.

The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.



**Epoch**

The epoch is the point where the time starts and is platform-dependent. On Windows and most Unix systems, the epoch is January 1, 1970, 00:00:00 (UTC), and leap seconds are not counted towards the time in seconds since the epoch. To check what the epoch is on a given platform we can use [time.gmtime(0)](https://www.geeksforgeeks.org/python-time-gmtime-method/).



**3.5 TECHNOLOGY USED**

Technologies used in chatbot are:-

1. Artificial Intelligence (AI)
2. Natural Language Processing (NLP)
3. Deep Learning (DL)
4. Bag of words (BOW)
5. Seq2Seq
6. Tokenization
7. Lemmatization

**Artificial Intelligence**

Chatbots is also called chatterbots. It is a form of artificial intelligence (AI) used in messaging apps. This tool helps add convenience for customers they are automated programs that interact with customers like a human would and cost little to nothing to engage with.

Artificial intelligence chatbots are text- or voice-based interfaces that provide support and connect human users with the services or information they need by simulating a traditional person-to-person conversation.

Text-based chatbots are often deployed online on websites and social media platforms to provide customer support and outreach. Voice-based chatbots, on the other hand, are most typically used for call deflection and sorting or over-the-phone customer service.

Most smartphones come equipped with a built-in chatbot, and smart speakers with chatbot functionality have been trendy gift-giving items for several years.

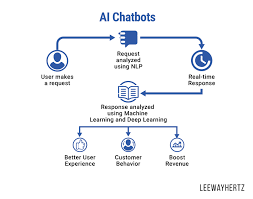


Figure 6 – AI chatbots

**Natural Language processing**

Natural language processing strives to build machines that understand and respond to text or voice data and respond with text or speech of their own in much the same way humans do.

Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There’s a good chance you’ve interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

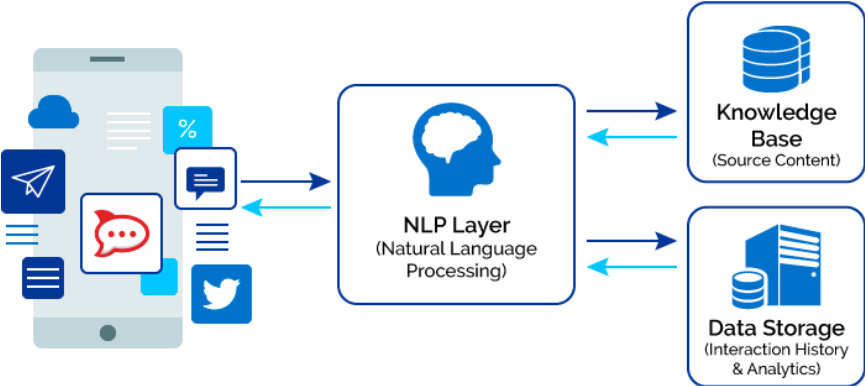


Figure 7 – NLP layer

**Deep Learning**

**Deep Learning** is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

**Deep learning** also known as **deep structured learning** is part of a broader family of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) methods based on [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) with [representation learning](https://en.wikipedia.org/wiki/Representation_learning). Learning can be [supervised](https://en.wikipedia.org/wiki/Supervised_learning), [semi-supervised](https://en.wikipedia.org/wiki/Semi-supervised_learning) or [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning).

Deep-learning architectures such as [deep neural networks](https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks), [deep belief networks](https://en.wikipedia.org/wiki/Deep_belief_network), [deep reinforcement learning](https://en.wikipedia.org/wiki/Deep_reinforcement_learning), [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks) and [convolutional neural networks](https://en.wikipedia.org/wiki/Convolutional_neural_networks) have been applied to fields including [computer vision](https://en.wikipedia.org/wiki/Computer_vision), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), [machine translation](https://en.wikipedia.org/wiki/Machine_translation), [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics), [drug design](https://en.wikipedia.org/wiki/Drug_design), [medical image analysis](https://en.wikipedia.org/wiki/Medical_image_analysis), [climate science](https://en.wikipedia.org/wiki/Climatology), material inspection and [board game](https://en.wikipedia.org/wiki/Board_game) programs, where they have produced results comparable to and in some cases surpassing human expert performance.

**Bag of words**

The **bag-of-words** (BOW) model is a representation that turns arbitrary text into **fixed-length vectors** by counting how many times each word appears. This process is often referred to as **vectorization.**

Let’s understand this with an example. Suppose we wanted to vectorize the following:

* the cat sat
* the cat sat in the hat
* the cat with the hat

# Step 1: Determine the Vocabulary

We first define our **vocabulary**, which is the set of all words found in our document set. The only words that are found in the 3 documents above are: the, cat, sat, in, the, hat, and with.

# Step 2: Count

To vectorize our documents, all we have to do is **count how many times each word appears**:



Figure 8 – BOW model

Now we have length-6 vectors for each document!

* the cat sat*:* [1, 1, 1, 0, 0, 0]
* the cat sat in the hat*:* [2, 1, 1, 1, 1, 0]
* the cat with the hat*:* [2, 1, 0, 0, 1, 1]

Notice that we lose contextual information, e.g. where in the document the word appeared, when we use BOW. It’s like a literal **bag**-of-words: it only tells you what words occur in the document, not where they occurred.

**Seq2Seq**

Sequence to Sequence (often abbreviated to seq2seq) models is a special class of Recurrent Neural Network architectures that we typically use (but not restricted) to solve complex Language problems like Machine Translation, Question Answering, creating Chatbots, Text Summarization, etc.

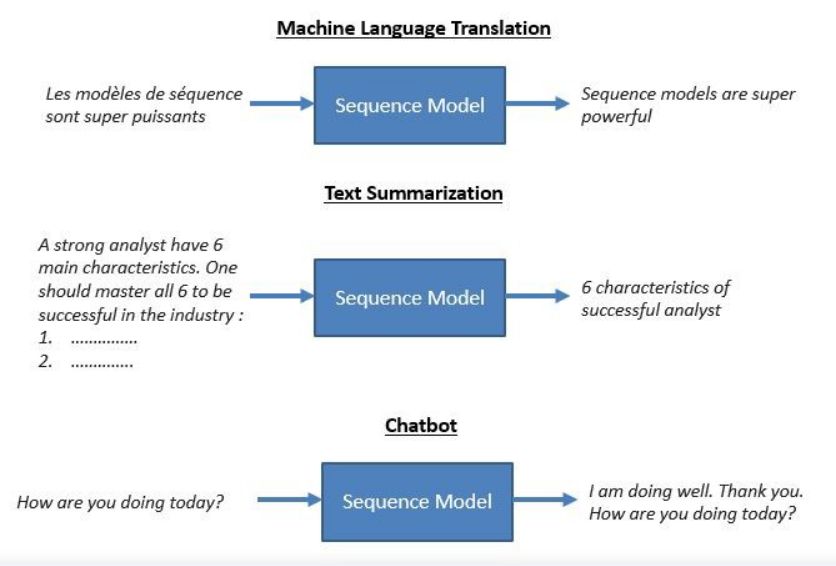


Figure 9 – ML Translation

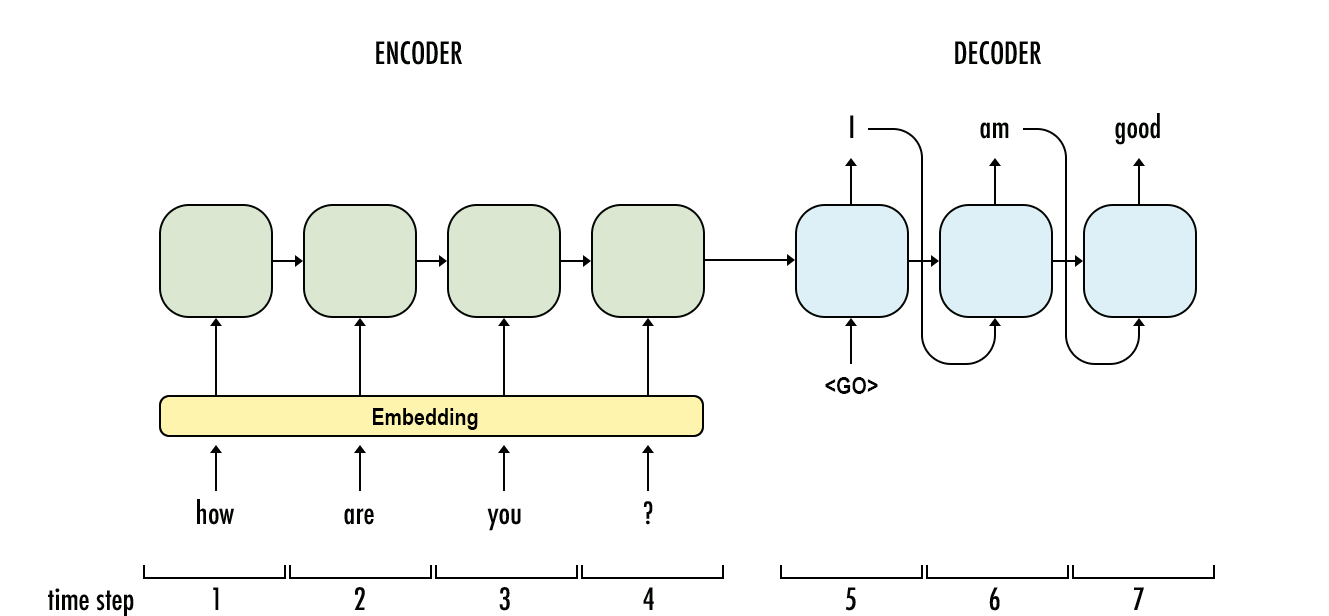


Figure 10 – Encoder-decoder

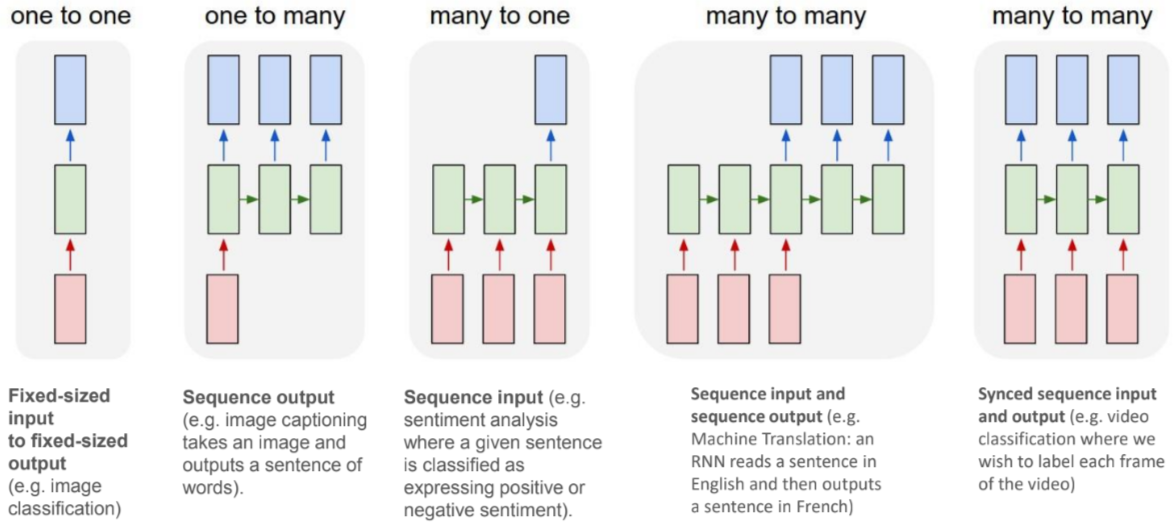


Figure 11 – RNN relationship

**Tokenization**

Tokenization is the task of chopping text up into pieces, called tokens, and at the same time throwing away certain characters, such as punctuation. These tokens are linguistically representative of the text. Tokenization is a common task of Natural Language Processing.



Figure 12 - Tokenization

**Lemmatization**

Lemmatization is another form of stemming. In lemmatization, different words with variation are considered as one word so that the program will consider them as once. The major difference between stemming and lemmatization is that stemming uses words that are not actual words but lemmatization uses actual language words.

**CHAPTER -4**

**PROJECT DEVELOPMENT**

**4.1 BACKEND CODE OF THE CHAT BOT**

1. **FILE NAME: -nltk\_utlis.py**

import numpy as np

import nltk

nltk.download('punkt')

from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()

def tokenize(sentence):

"""

split sentence into array of words/tokens

a token can be a word or punctuation character, or number

"""

return nltk.word\_tokenize(sentence)

def stem(word):

"""

stemming = find the root form of the word

examples:

words = ["organize", "organizes", "organizing"]

words = [stem(w) for w in words]

-> ["organ", "organ", "organ"]

"""

return stemmer.stem(word.lower())

def bag\_of\_words(tokenized\_sentence, words):

"""

return bag of words array:

1 for each known word that exists in the sentence, 0 otherwise

example:

sentence = ["hello", "how", "are", "you"]

words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]

bog = [ 0 , 1 , 0 , 1 , 0 , 0 , 0]

"""

# stem each word

sentence\_words = [stem(word) for word in tokenized\_sentence]

# initialize bag with 0 for each word

bag = np.zeros(len(words), dtype=np.float32)

for idx, w in enumerate(words):

if w in sentence\_words:

bag[idx] = 1

return bag

**2- FILE NAME: -train.py**

import numpy as np

import random

import json

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader

from nltk\_utils import bag\_of\_words, tokenize, stem

from model import NeuralNet

with open('intents.json', 'r') as f:

intents = json.load(f)

all\_words = []

tags = []

xy = []

# loop through each sentence in our intents patterns

for intent in intents['intents']:

tag = intent['tag']

# add to tag list

tags.append(tag)

for pattern in intent['patterns']:

# tokenize each word in the sentence

w = tokenize(pattern)

# add to our words list

all\_words.extend(w)

# add to xy pair

xy.append((w, tag))

# stem and lower each word

ignore\_words = ['?', '.', '!']

all\_words = [stem(w) for w in all\_words if w not in ignore\_words]

# remove duplicates and sort

all\_words = sorted(set(all\_words))

tags = sorted(set(tags))

print(len(xy), "patterns")

print(len(tags), "tags:", tags)

print(len(all\_words), "unique stemmed words:", all\_words)

# create training data

X\_train = []

y\_train = []

for (pattern\_sentence, tag) in xy:

# X: bag of words for each pattern\_sentence

bag = bag\_of\_words(pattern\_sentence, all\_words)

X\_train.append(bag)

# y: PyTorch CrossEntropyLoss needs only class labels, not one-hot

label = tags.index(tag)

y\_train.append(label)

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

# Hyper-parameters

num\_epochs = 1000

batch\_size = 8

learning\_rate = 0.001

input\_size = len(X\_train[0])

hidden\_size = 8

output\_size = len(tags)

print(input\_size, output\_size)

class ChatDataset(Dataset):

def \_\_init\_\_(self):

self.n\_samples = len(X\_train)

self.x\_data = X\_train

self.y\_data = y\_train

# support indexing such that dataset[i] can be used to get i-th sample

def \_\_getitem\_\_(self, index):

return self.x\_data[index], self.y\_data[index]

# we can call len(dataset) to return the size

def \_\_len\_\_(self):

return self.n\_samples

dataset = ChatDataset()

train\_loader = DataLoader(dataset=dataset,

batch\_size=batch\_size,

shuffle=True,

num\_workers=0)

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)

# Loss and optimizer

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)

# Train the model

for epoch in range(num\_epochs):

for (words, labels) in train\_loader:

words = words.to(device)

labels = labels.to(dtype=torch.long).to(device)

# Forward pass

outputs = model(words)

# if y would be one-hot, we must apply

# labels = torch.max(labels, 1)[1]

loss = criterion(outputs, labels)

# Backward and optimize

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if (epoch+1) % 100 == 0:

print (f'Epoch [{epoch+1}/{num\_epochs}], Loss: {loss.item():.4f}')

print(f'final loss: {loss.item():.4f}')

data = {

"model\_state": model.state\_dict(),

"input\_size": input\_size,

"hidden\_size": hidden\_size,

"output\_size": output\_size,

"all\_words": all\_words,

"tags": tags

}

FILE = "data.pth"

torch.save(data, FILE)

print(f'training complete. file saved to {FILE}')

**3-FILE NAME: - model.py**

import torch

import torch.nn as nn

class NeuralNet(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size, num\_classes):

super(NeuralNet, self).\_\_init\_\_()

self.l1 = nn.Linear(input\_size, hidden\_size)

self.l2 = nn.Linear(hidden\_size, hidden\_size)

self.l3 = nn.Linear(hidden\_size, num\_classes)

self.relu = nn.ReLU()

def forward(self, x):

out = self.l1(x)

out = self.relu(out)

out = self.l2(out)

out = self.relu(out)

out = self.l3(out)

# no activation and no softmax at the end

return out

**4- FILE NAME:-chat.py**

import random

import json

import torch

from model import NeuralNet

from nltk\_utils import bag\_of\_words, tokenize

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

with open('intents.json', 'r') as json\_data:

intents = json.load(json\_data)

FILE = "data.pth"

data = torch.load(FILE)

input\_size = data["input\_size"]

hidden\_size = data["hidden\_size"]

output\_size = data["output\_size"]

all\_words = data['all\_words']

tags = data['tags']

model\_state = data["model\_state"]

model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)

model.load\_state\_dict(model\_state)

model.eval()

bot\_name = "iBot"

print("Let's chat! (type 'quit' to exit)")

while True:

# sentence = "do you use credit cards?"

sentence = input("You: ")

if sentence == "quit":

break

sentence = tokenize(sentence)

X = bag\_of\_words(sentence, all\_words)

X = X.reshape(1, X.shape[0])

X = torch.from\_numpy(X).to(device)

output = model(X)

\_, predicted = torch.max(output, dim=1)

tag = tags[predicted.item()]

probs = torch.softmax(output, dim=1)

prob = probs[0][predicted.item()]

if prob.item() > 0.75:

for intent in intents['intents']:

if tag == intent["tag"]:

print(f"{bot\_name}: {random.choice(intent['responses'])}")

else:

print(f"{bot\_name}: I do not understand...")

**FRONTEND CODE**

1. **File Name: index.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">

<script src="https://kit.fontawesome.com/a658a7b479.js" crossorigin="anonymous"></script>

<link rel="stylesheet" href="https://unpkg.com/aos@next/dist/aos.css" />

<link rel="icon" href="./static/img/icon.png">

<link rel="stylesheet" href="./static/demo\_bot/chatbot-ui.css">

<link rel="stylesheet" href="./static/prism.css" />

<link rel="stylesheet" href="./static/styles.css">

<title>I-Bot</title>

</head>

<body>

<div class="hero" id="hero">

<nav class="navbar navbar-expand-lg navbar-light ">

<div class="container">

<a class="navbar-brand" href="#hero"><i class="fas fa-atom"></i> iBot <span></span> </a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation"><span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav ml-auto">

<li class="nav-item">

<a class="nav-link text-white hover" href="#features">Features</a>

</li>

<li class="nav-item active">

<a class="nav-link text-white font-weight-bold hover" href="#get-started">Get Started</a>

</li>

<li class="nav-item">

<a class="nav-link text-white hover" href="#themes-and-upgrades">Themes And Upgrades</a>

</li>

</ul>

</div>

</div>

</nav>

<br>

<div class="hero-content">

<div class="hero-bot mobile">

<img src="./static/img/homepage-bot.png " alt="Bot Image ">

</div>

<div data-aos="fade-down-right" data-aos-duration="1000" class="hero-header">

<h1>Get your hands on our<br><span class="primary-color"> Ready-to-use integrated Chatbot</span><br> for your esteemed organisation!</h1>

<br />

<p>

Looking for an AI to handle your customer queries and handle multiple requests simultaneuosly with responsive features? Look no further! Explore our ready to use iBot and ensure smooth communication across all queries. He's smart, he's savvy, he's bold, he's a bit like us! Don't

you worry! Customize him according to your liking very quickly and easily.

</p>

<br>

<h3><span class="primary-color cachy-line">Embed. Customize. Repeat.</span></h3>

</div>

<div data-aos="fade-up-left" data-aos-duration="1000" class="hero-bot desktop ">

<img src="./static/img/homepage-bot.png " alt="Bot Image ">

</div>

</div>

<br>

</div>

<div id="features">

<h1 class="section-title"><span class="primary-color ">Features</span></h1><br>

<div class="container">

<div class="card-outer-container">

<div data-aos="fade-right" data-aos-duration="800" class="card feature-card" style="width: 18rem;background-color: rgb(48,90,117);color:white ">

<div class="card-body">

<i class="fa-3x fa fa-desktop mb-3" aria-hidden="true "></i>

<br>

<h5 class="card-title">EASY TO USE</h5>

<p class="card-text">Chat with him like you would with anyone!Throw away your queries! It's as simple as that.</p>

</div>

</div>

<div data-aos="zoom-in" data-aos-duration="800" class="card feature-card" style="width: 18rem;background-color: rgb(14,55,74);color:white ">

<div class="card-body">

<i class="fa-3x fa fa-trophy mb-3" aria-hidden="true"> </i>

<br>

<h5 class="card-title">INTELLIGENT</h5>

<p class="card-text">Keep talking and he'll keep learning. Trained on multiple datasets to efficiently answer all your questions </p>

</div>

</div>

<div data-aos="fade-left" data-aos-duration="800" class="card feature-card" style="width: 18rem;background-color: rgb(48,90,117);color:white ">

<div class="card-body">

<i class="fa-3x fa fa-magic mb-3" aria-hidden="true"></i>

<br>

<h5 class="card-title">EASY TO CUSTOMIZE</h5>

<p class="card-text">Explore different packages and designs available for you. You can also customize it according to your taste!</p>

</div>

</div>

</div>

</div>

</div>

<br><br>

<br>

<br><br>

</div>

</div>

<!-- Themes and Upgrade Section Completed -->

<br><br>

<div class="p-2 bg-warning font-weight-bold text-dark text-center bottom-0">Developed by - Athar Zahid Usmani | Arshiya Dilshad | Aamna Akhtar <a href="mailto:abc@gmail.com " class="text-danger "></a></div>

<!-- Chat Bot -->

<div id="chat-container"></div>

<script src="./static/demo\_bot/chatbot-ui.js"></script>

<script>

createChatBot(host = 'http://localhost:5000/chat',

botLogo = "./static/img/bot-logo.png",

title = "I BOT", welcomeMessage = "Hey, i am UI Bot, I am here to help you.",

inactiveMsg = "This is inactive message, show casing that the server is down"

)

</script>

<!-- <div class="aos-init" data-aos="zoom-in-up" data-aos-duration="3000"></div> -->

<script src="./static/prism.js "></script>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js " crossorigin="anonymous "></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js " integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s " crossorigin="anonymous "></script>

<script src="https://unpkg.com/aos@next/dist/aos.js "></script>

<script>

AOS.init();

</script>

<script>

$('[data-aos]').parent().addClass('hideOverflowOnMobile');

</script>

</body>

</html>

1. **File Name: -Chatbot-ul.js**

/\*

Makes backend API call to rasa chatbot and display output to chatbot frontend

\*/

function init() {

//---------------------------- Including Jquery ------------------------------

var script = document.createElement('script');

script.src = 'https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js';

script.type = 'text/javascript';

document.getElementsByTagName('head')[0].appendChild(script);

//--------------------------- Important Variables----------------------------

botLogoPath = "./imgs/bot-logo.png"

//--------------------------- Chatbot Frontend -------------------------------

const chatContainer = document.getElementById("chat-container");

template = ` <button class='chat-btn'><img src = "./icons/comment.png" class = "icon" ></button>

<div class='chat-popup'>

<div class='chat-header'>

<div class='chatbot-img'>

<img src='${botLogoPath}' alt='Chat Bot image' class='bot-img'>

</div>

<h3 class='bot-title'>Covid Bot</h3>

<button class = "expand-chat-window" ><img src="./icons/open\_fullscreen.png" class="icon" ></button>

</div>

<div class='chat-area'>

<div class='bot-msg'>

<img class='bot-img' src ='${botLogoPath}' />

<span class='msg'>Hi, How can i help you?</span>

</div>

<!-- <div class='bot-msg'>

<img class='bot-img' src ='${botLogoPath}' />

<div class='response-btns'>

<button class='btn-primary' onclick= 'userResponseBtn(this)' value='/sign\_in'>sample btn</button>

</div>

</div> -->

<!-- <div class='bot-msg'>

<img class='msg-image' src = "https://i.imgur.com/nGF1K8f.jpg" />

</div> -->

<!-- <div class='user-msg'>

<span class='msg'>Hi, How can i help you?</span>

</div> -->

</div>

<div class='chat-input-area'>

<input type='text' autofocus class='chat-input' onkeypress='return givenUserInput(event)' placeholder='Type a message ...' autocomplete='off'>

<button class='chat-submit'><i class='material-icons'>send</i></button>

</div>

</div>`

chatContainer.innerHTML = template;

//--------------------------- Important Variables----------------------------

var inactiveMessage = "Server is down, Please contact the developer to activate it"

chatPopup = document.querySelector(".chat-popup")

chatBtn = document.querySelector(".chat-btn")

chatSubmit = document.querySelector(".chat-submit")

chatHeader = document.querySelector(".chat-header")

chatArea = document.querySelector(".chat-area")

chatInput = document.querySelector(".chat-input")

expandWindow = document.querySelector(".expand-chat-window")

root = document.documentElement;

chatPopup.style.display = "none"

var host = ""

//------------------------ ChatBot Toggler -------------------------

chatBtn.addEventListener("click", () => {

mobileDevice = !detectMob()

if (chatPopup.style.display == "none" && mobileDevice) {

chatPopup.style.display = "flex"

chatInput.focus();

chatBtn.innerHTML = `<img src = "./icons/close.png" class = "icon" >`

} else if (mobileDevice) {

chatPopup.style.display = "none"

chatBtn.innerHTML = `<img src = "./icons/comment.png" class = "icon" >`

} else {

mobileView()

}

})

chatSubmit.addEventListener("click", () => {

let userResponse = chatInput.value.trim();

if (userResponse !== "") {

setUserResponse();

send(userResponse)

}

})

expandWindow.addEventListener("click", (e) => {

// console.log(expandWindow.innerHTML)

if (expandWindow.innerHTML == '<img src="./icons/open\_fullscreen.png" class="icon">') {

expandWindow.innerHTML = `<img src = "./icons/close\_fullscreen.png" class = 'icon'>`

root.style.setProperty('--chat-window-height', 80 + "%");

root.style.setProperty('--chat-window-total-width', 85 + "%");

} else if (expandWindow.innerHTML == '<img src="./icons/close.png" class="icon">') {

chatPopup.style.display = "none"

chatBtn.style.display = "block"

} else {

expandWindow.innerHTML = `<img src = "./icons/open\_fullscreen.png" class = "icon" >`

root.style.setProperty('--chat-window-height', 500 + "px");

root.style.setProperty('--chat-window-total-width', 380 + "px");

}

})

}

// end of init function

var passwordInput = false;

function userResponseBtn(e) {

send(e.value);

}

// to submit user input when he presses enter

function givenUserInput(e) {

if (e.keyCode == 13) {

let userResponse = chatInput.value.trim();

if (userResponse !== "") {

setUserResponse()

send(userResponse)

}

}

}

// to display user message on UI

function setUserResponse() {

let userInput = chatInput.value;

if (passwordInput) {

userInput = "\*\*\*\*\*\*"

}

if (userInput) {

let temp = `<div class="user-msg"><span class = "msg">${userInput}</span></div>`

chatArea.innerHTML += temp;

chatInput.value = ""

} else {

chatInput.disabled = false;

}

scrollToBottomOfResults();

}

function scrollToBottomOfResults() {

chatArea.scrollTop = chatArea.scrollHeight;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Frontend Part Completed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// host = 'http://localhost:5005/webhooks/rest/webhook'

function send(message) {

chatInput.type = "text"

passwordInput = false;

chatInput.focus();

console.log("User Message:", message)

$.ajax({

url: host,

type: 'POST',

contentType: 'application/json',

data: JSON.stringify({

"message": message,

"sender": "User"

}),

success: function(data, textStatus) {

if (data != null) {

setBotResponse(data);

}

console.log("Rasa Response: ", data, "\n Status:", textStatus)

},

error: function(errorMessage) {

setBotResponse("");

console.log('Error' + errorMessage);

}

});

chatInput.focus();

}

//------------------------------------ Set bot response -------------------------------------

function setBotResponse(val) {

setTimeout(function() {

if (val.length < 1) {

//if there is no response from Rasa

// msg = 'I couldn\'t get that. Let\' try something else!';

msg = inactiveMessage;

var BotResponse = `<div class='bot-msg'><img class='bot-img' src ='${botLogoPath}' /><span class='msg'> ${msg} </span></div>`;

$(BotResponse).appendTo('.chat-area').hide().fadeIn(1000);

scrollToBottomOfResults();

chatInput.focus();

} else {

//if we get response from Rasa

for (i = 0; i < val.length; i++) {

//check if there is text message

if (val[i].hasOwnProperty("text")) {

const botMsg = val[i].text;

if (botMsg.includes("password")) {

chatInput.type = "password";

passwordInput = true;

}

var BotResponse = `<div class='bot-msg'><img class='bot-img' src ='${botLogoPath}' /><span class='msg'>${val[i].text}</span></div>`;

$(BotResponse).appendTo('.chat-area').hide().fadeIn(1000);

}

//check if there is image

if (val[i].hasOwnProperty("image")) {

var BotResponse = "<div class='bot-msg'>" + "<img class='bot-img' src ='${botLogoPath}' />"

'<img class="msg-image" src="' + val[i].image + '">' +

'</div>'

$(BotResponse).appendTo('.chat-area').hide().fadeIn(1000);

}

//check if there are buttons

if (val[i].hasOwnProperty("buttons")) {

var BotResponse = `<div class='bot-msg'><img class='bot-img' src ='${botLogoPath}' /><div class='response-btns'>`

buttonsArray = val[i].buttons;

buttonsArray.forEach(btn => {

BotResponse += `<button class='btn-primary' onclick= 'userResponseBtn(this)' value='${btn.payload}'>${btn.title}</button>`

})

BotResponse += "</div></div>"

$(BotResponse).appendTo('.chat-area').hide().fadeIn(1000);

chatInput.disabled = true;

}

}

scrollToBottomOfResults();

chatInput.disabled = false;

chatInput.focus();

}

}, 500);

}

function mobileView() {

$('.chat-popup').width($(window).width());

if (chatPopup.style.display == "none") {

chatPopup.style.display = "flex"

// chatInput.focus();

chatBtn.style.display = "none"

chatPopup.style.bottom = "0"

chatPopup.style.right = "0"

// chatPopup.style.transition = "none"

expandWindow.innerHTML = `<img src = "./icons/close.png" class = "icon" >`

}

}

function detectMob() {

return ((window.innerHeight <= 800) && (window.innerWidth <= 600));

}

function chatbotTheme(theme) {

const gradientHeader = document.querySelector(".chat-header");

const orange = {

color: "#FBAB7E",

background: "linear-gradient(19deg, #FBAB7E 0%, #F7CE68 100%)"

}

const purple = {

color: "#B721FF",

background: "linear-gradient(19deg, #21D4FD 0%, #B721FF 100%)"

}

if (theme === "orange") {

root.style.setProperty('--chat-window-color-theme', orange.color);

gradientHeader.style.backgroundImage = orange.background;

chatSubmit.style.backgroundColor = orange.color;

} else if (theme === "purple") {

root.style.setProperty('--chat-window-color-theme', purple.color);

gradientHeader.style.backgroundImage = purple.background;

chatSubmit.style.backgroundColor = purple.color;

}

}

function createChatBot(hostURL, botLogo, title, welcomeMessage, inactiveMsg, theme = "blue") {

host = hostURL;

botLogoPath = botLogo;

inactiveMessage = inactiveMsg;

init()

const msg = document.querySelector(".msg");

msg.innerText = welcomeMessage;

const botTitle = document.querySelector(".bot-title");

botTitle.innerText = title;

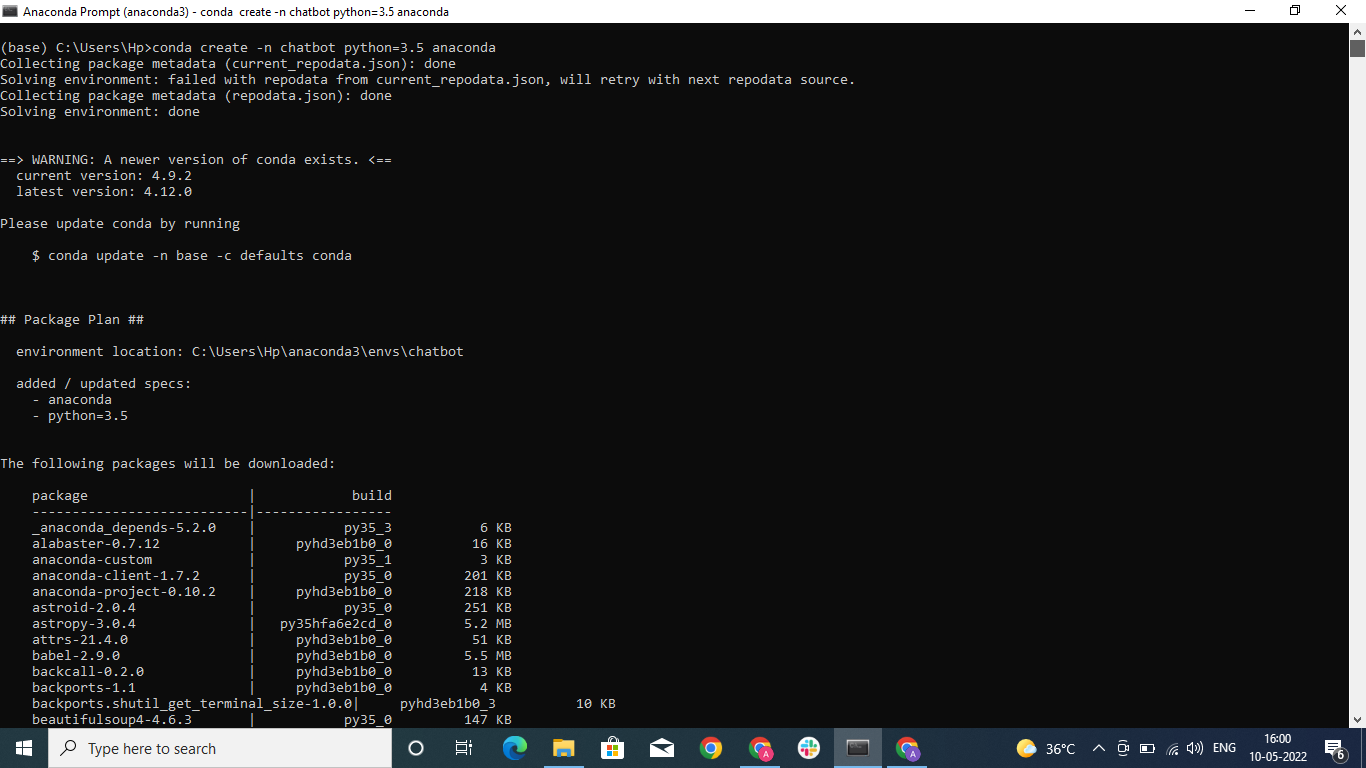
chatbotTheme(theme)

}

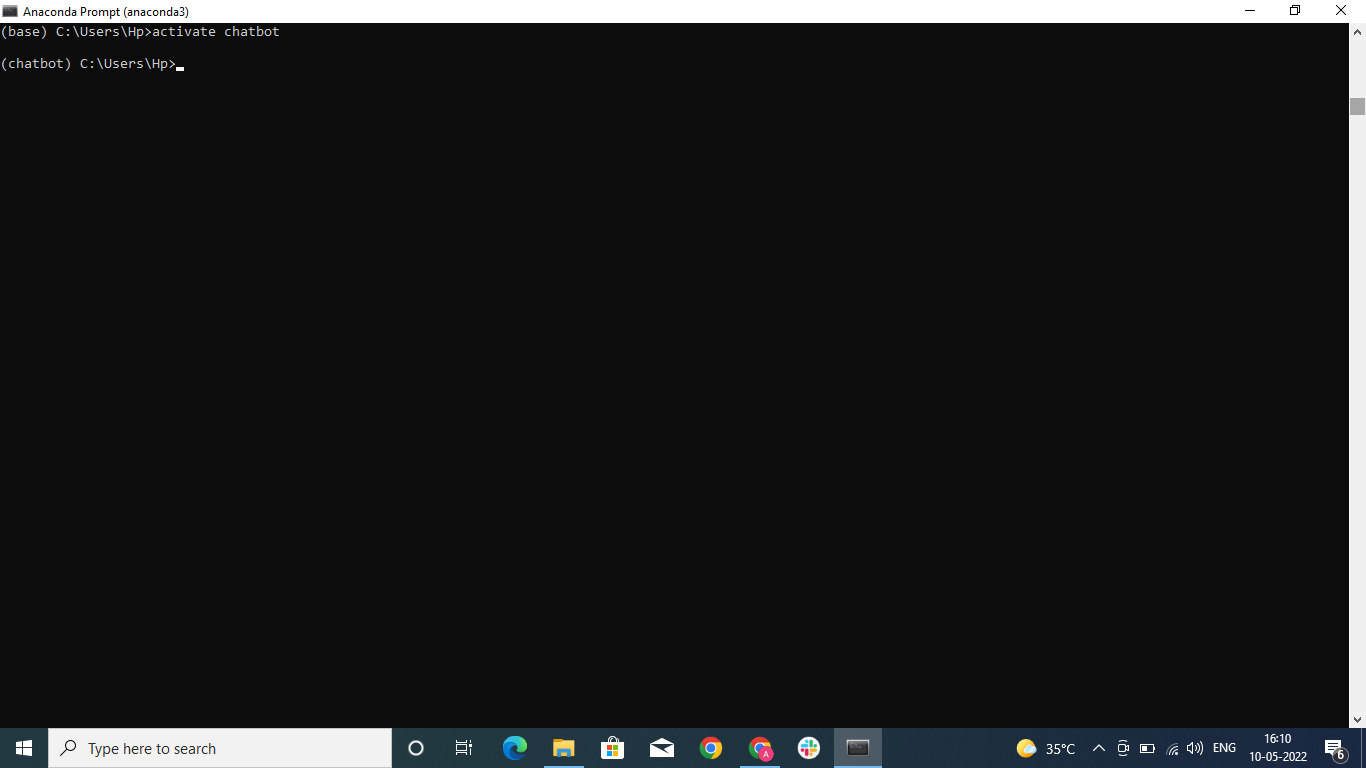
**IMPLEMENTATION**

**Building A chatbot using Deep NLP**

**Step 1:** Creating environment named chatbot, for our iBot

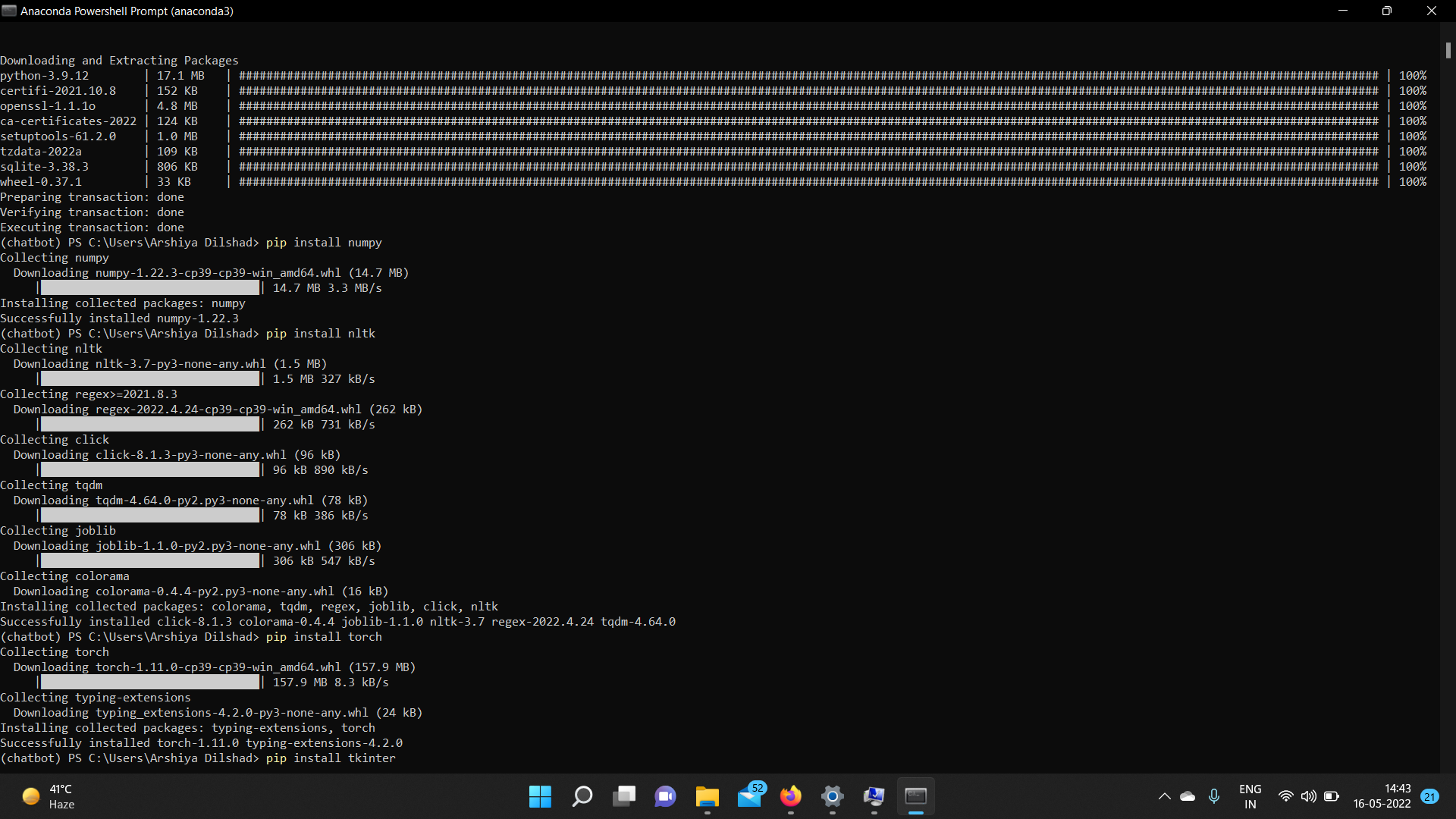


Activating the chatbot environment.



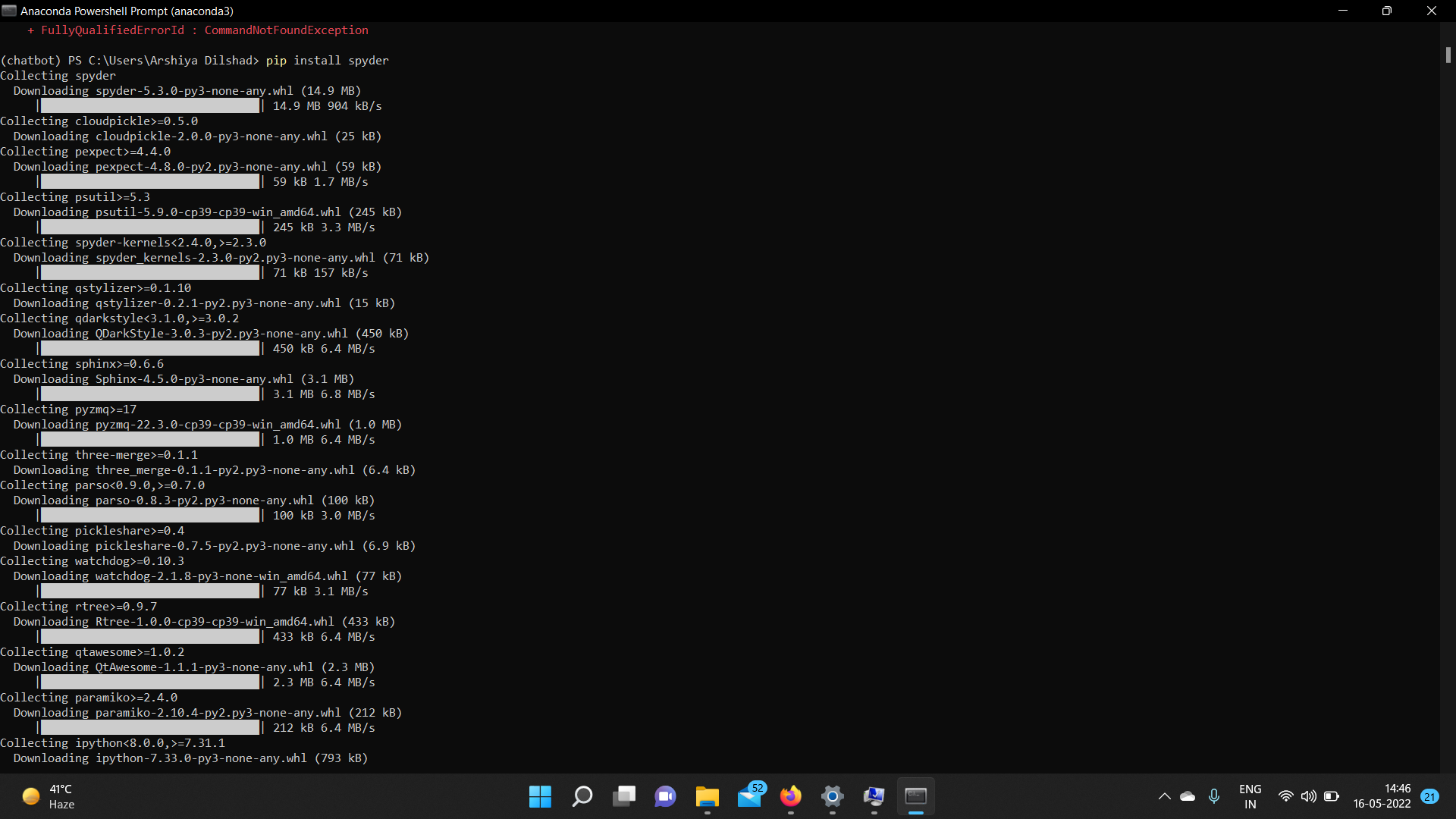
Installing the required libraries-

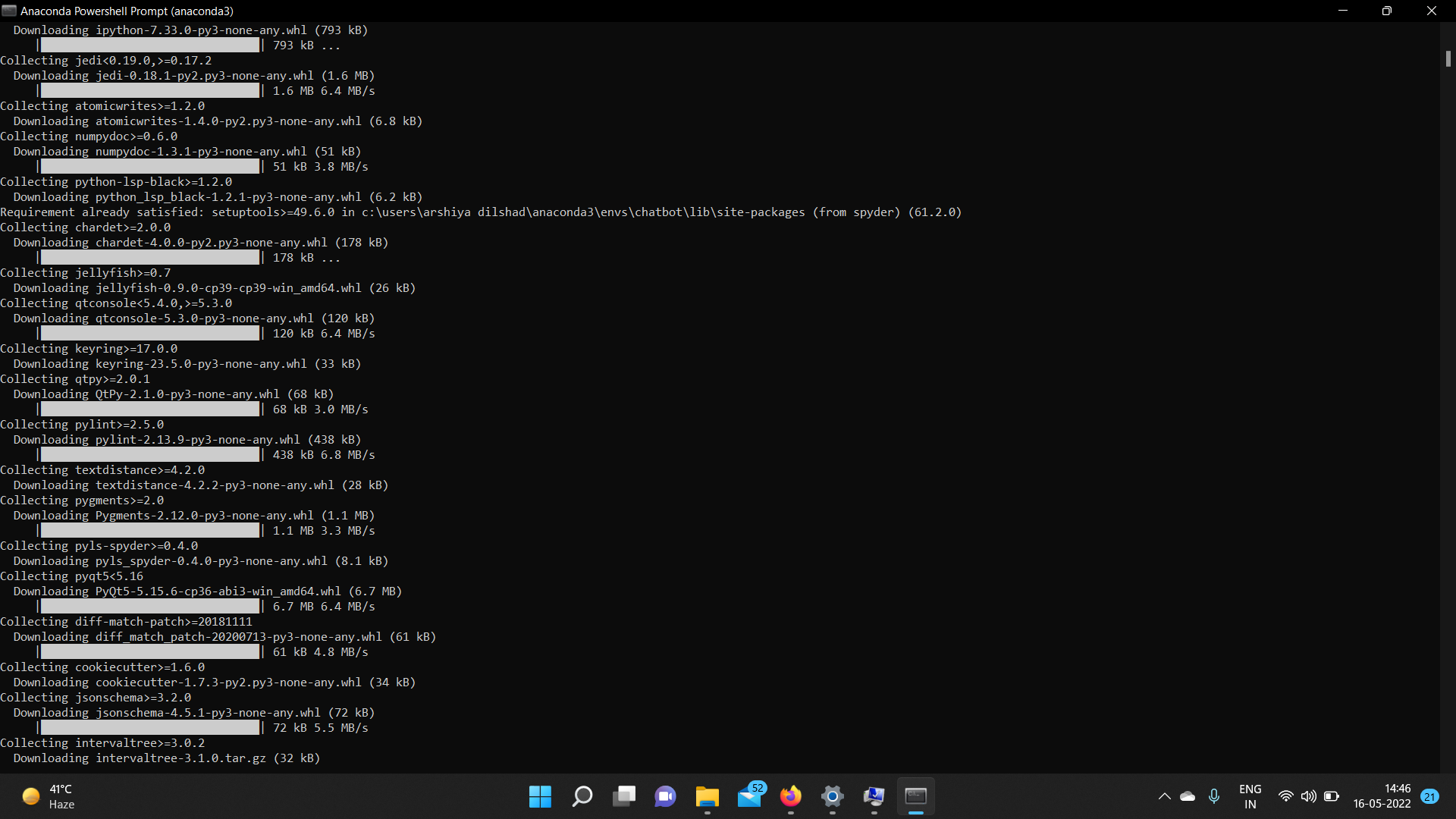
1. Numpy

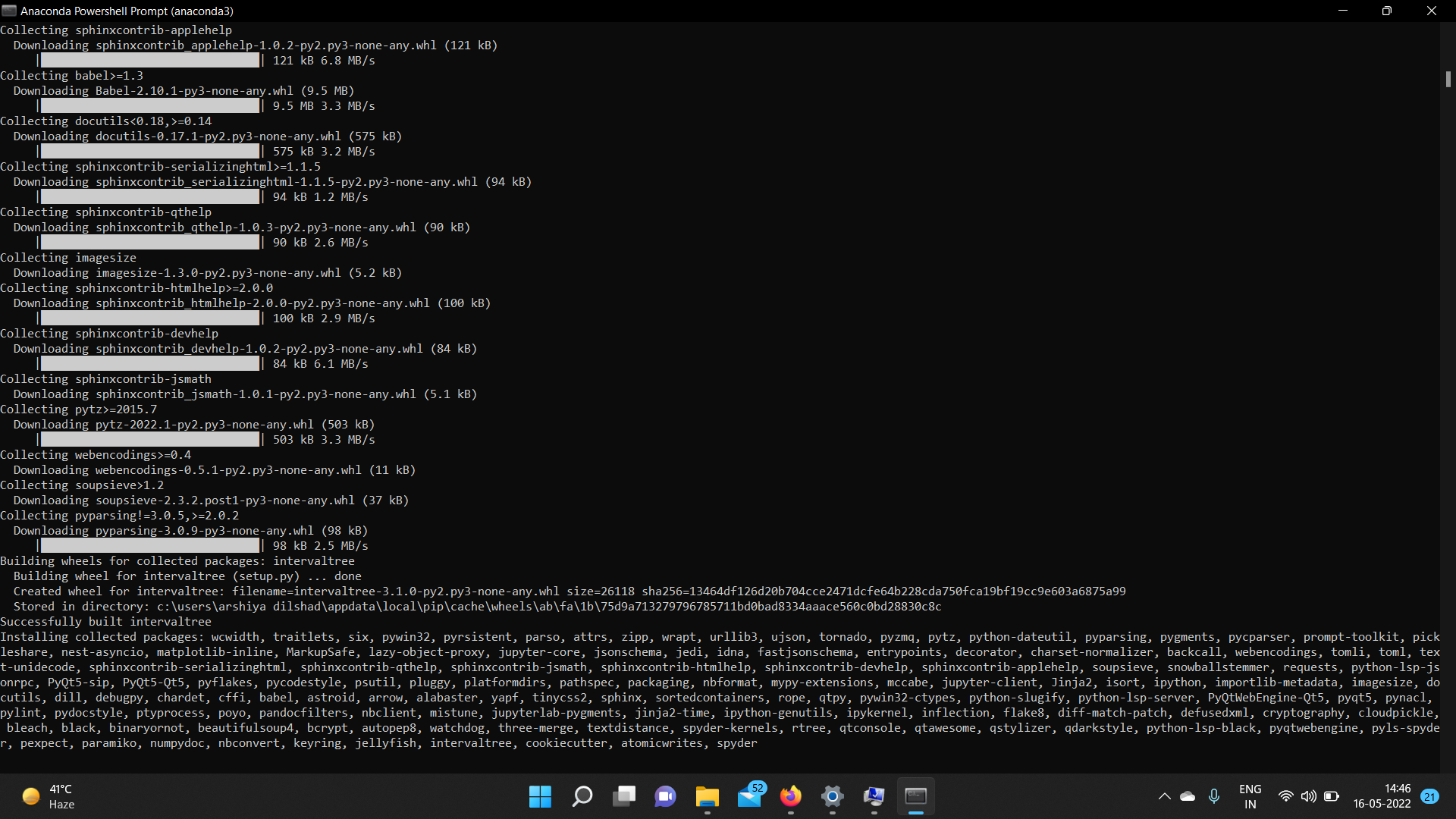


And like wise, installed nltk and torch.

Installing spyder IDE

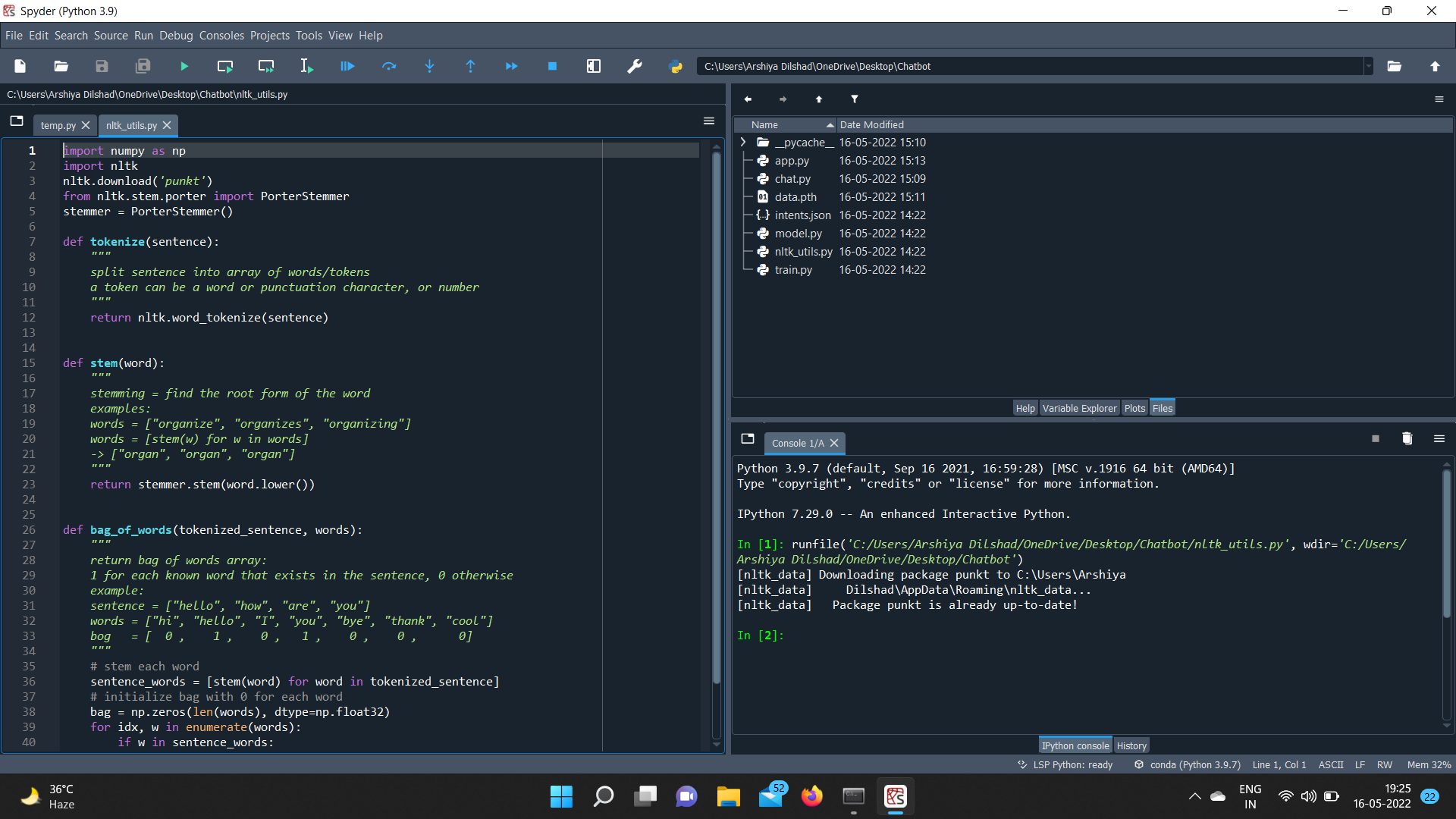


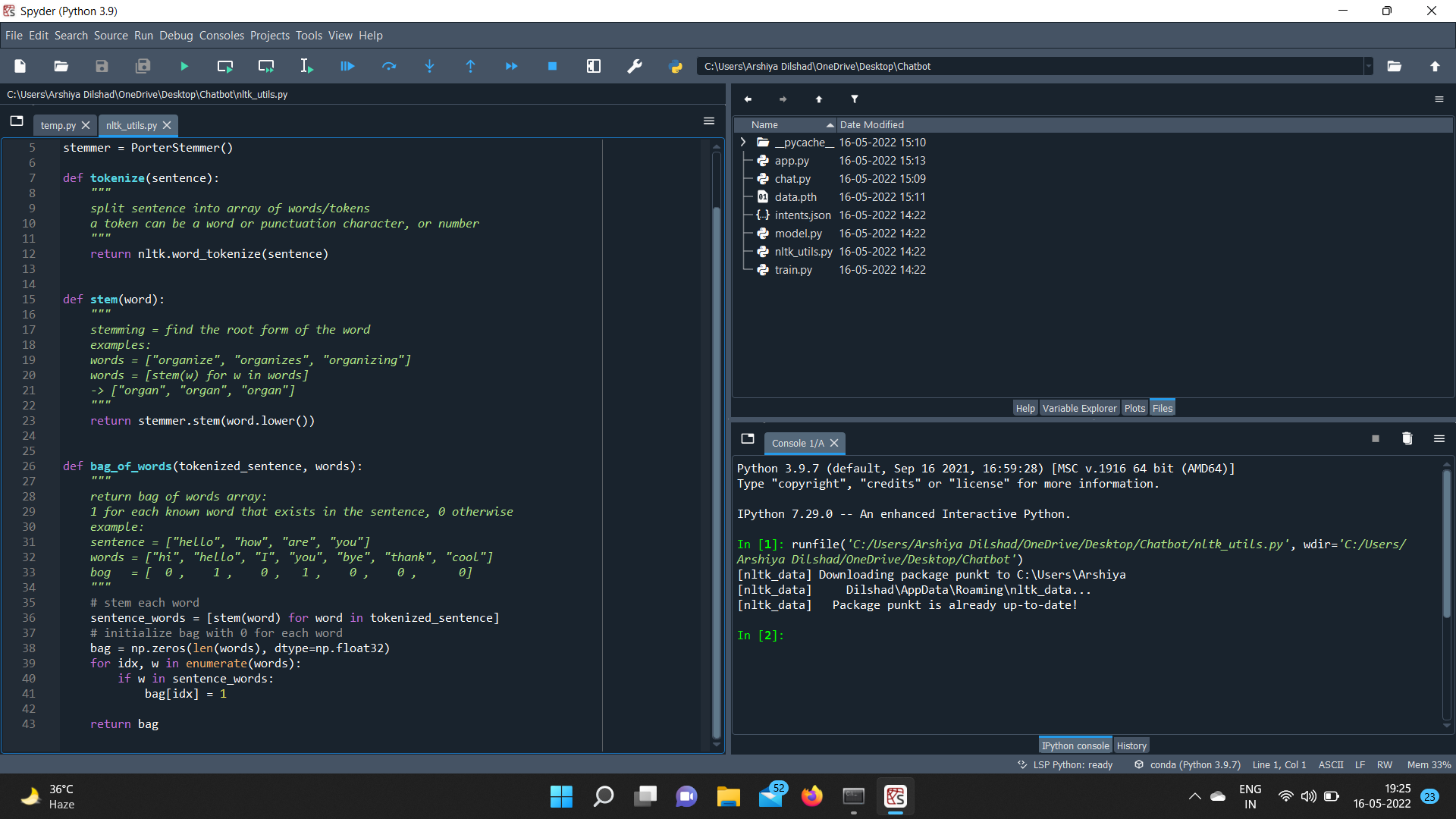




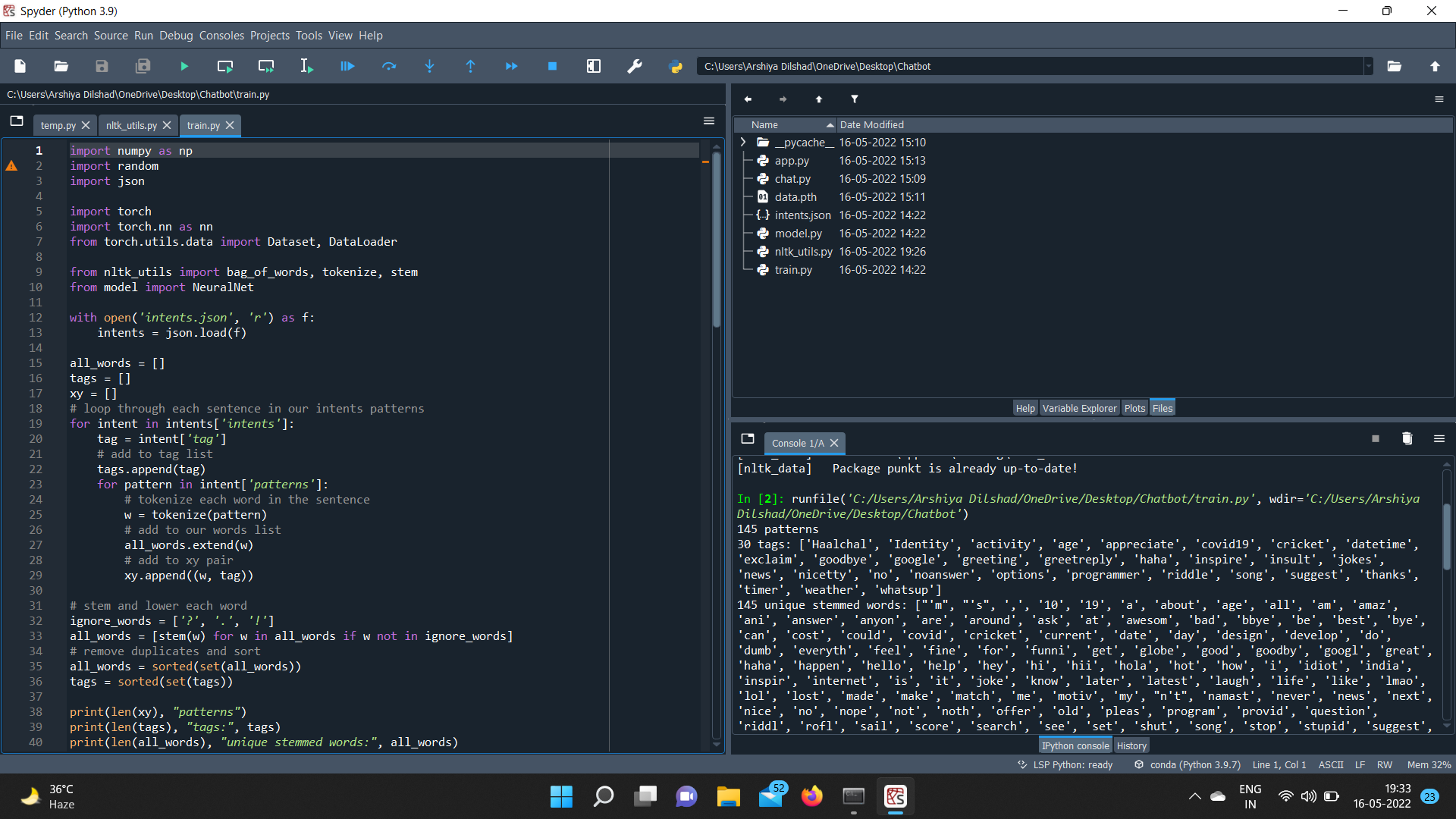


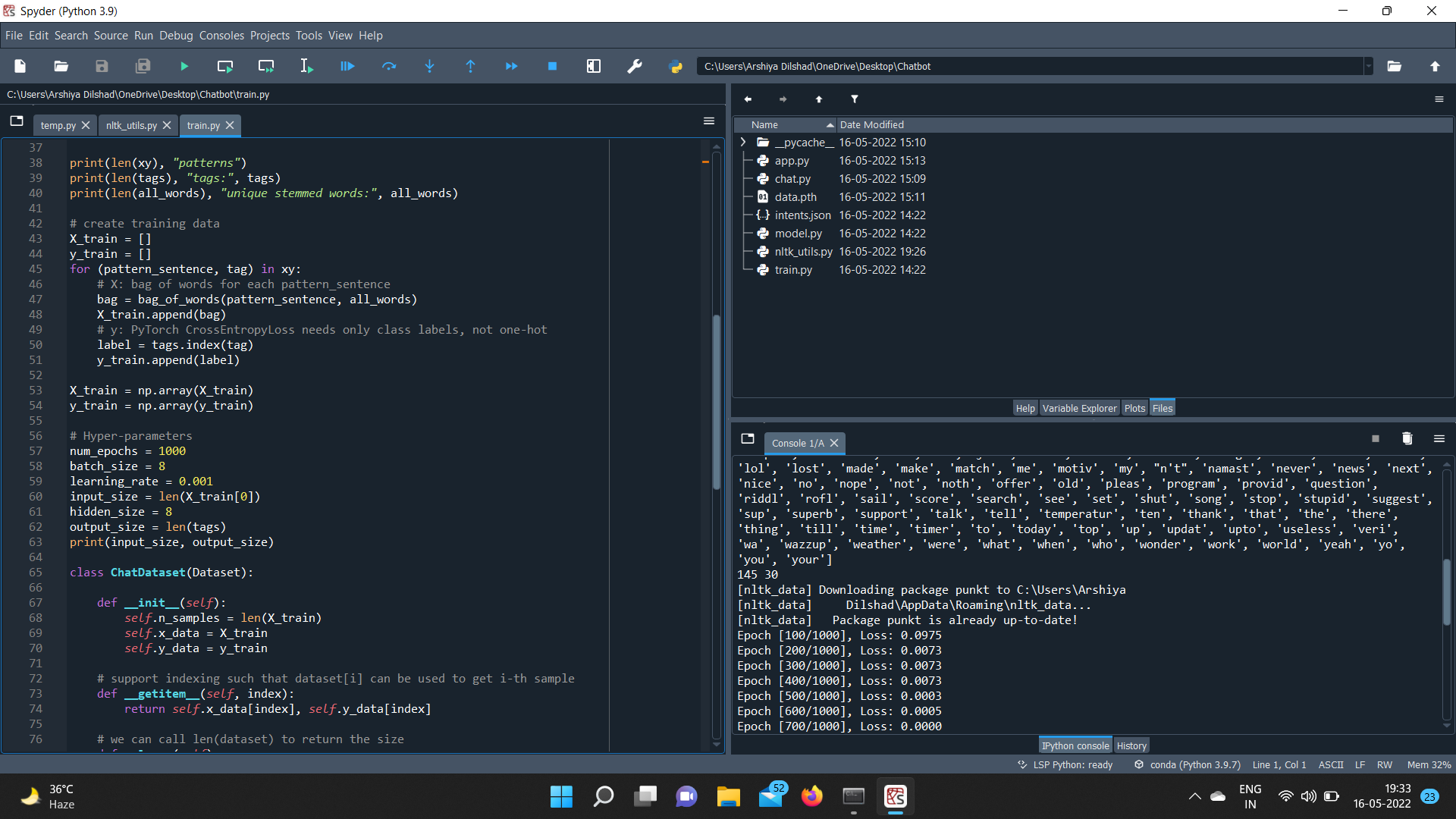
**Step 2:** First we created an nltk\_utils.py file and ran it

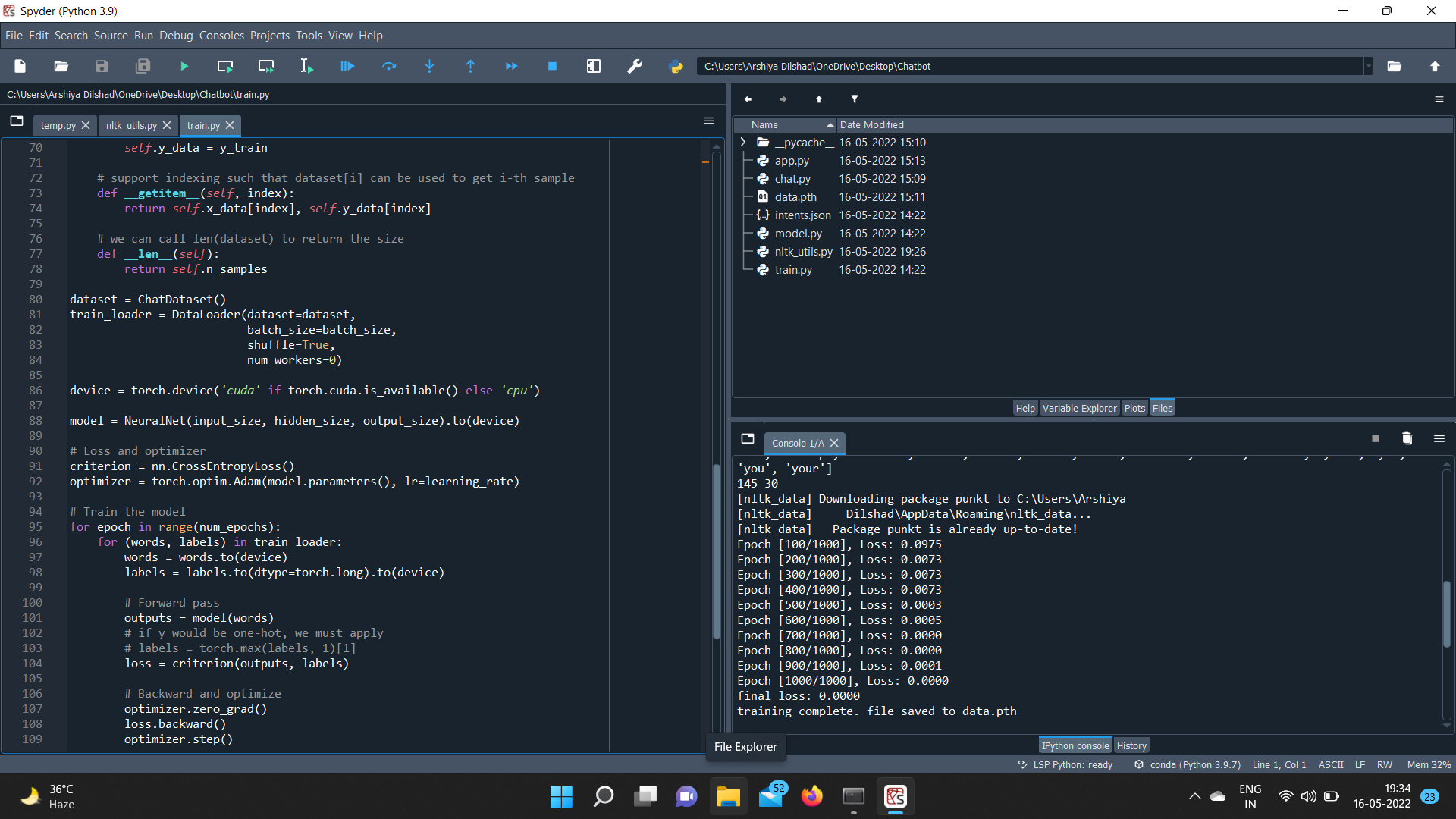


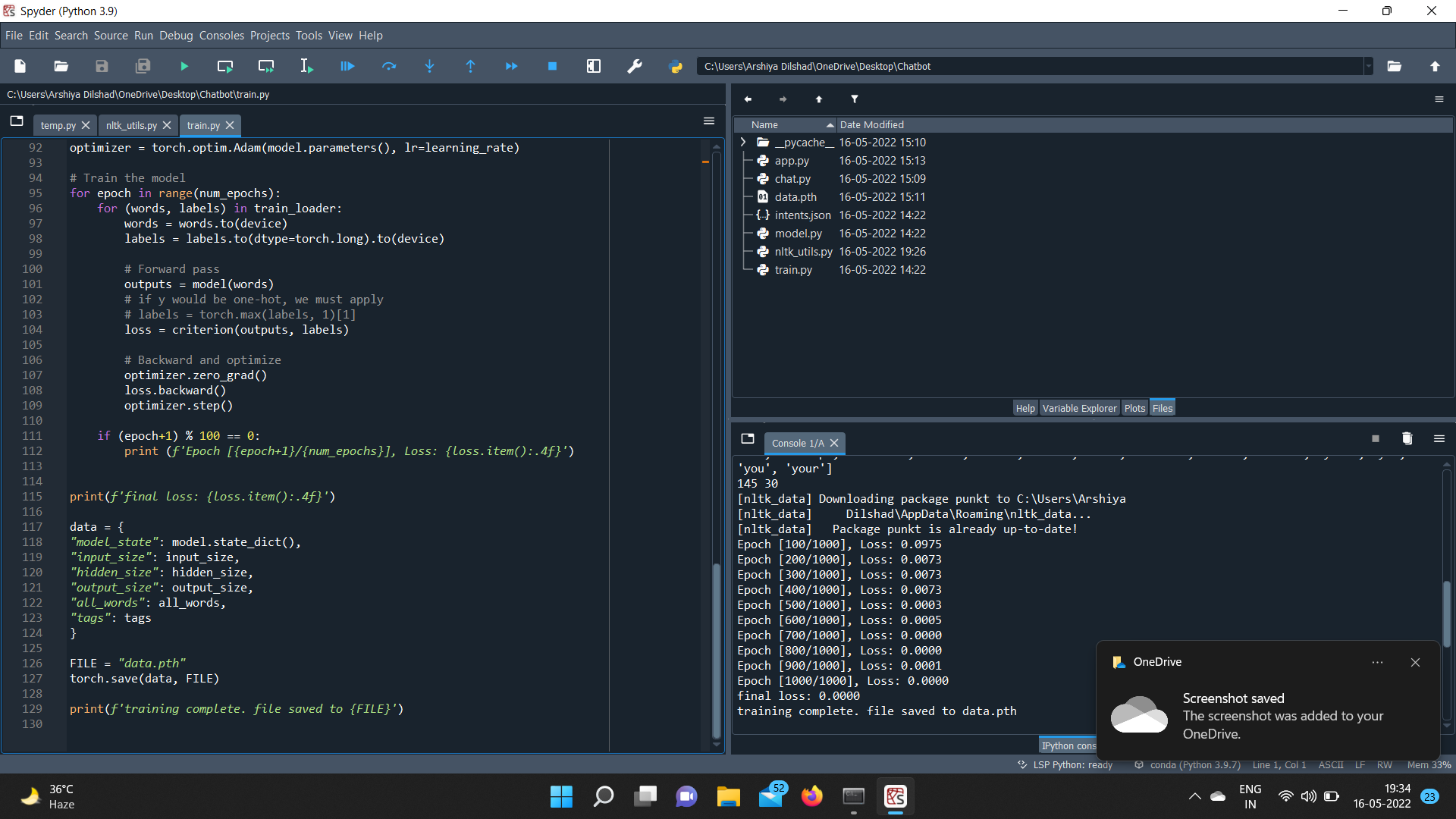


**Step 3:** Then, train.py file was created and ran

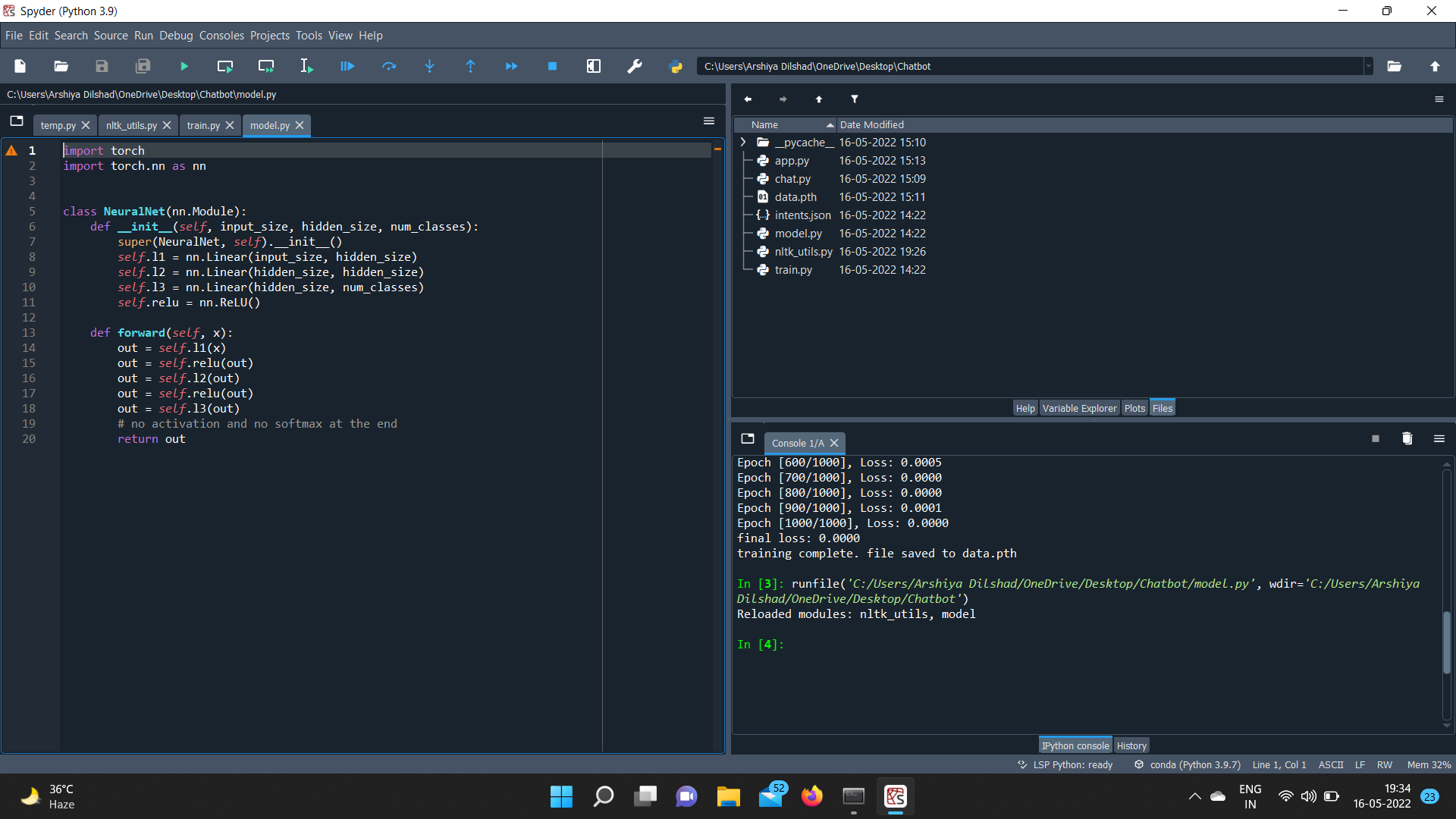




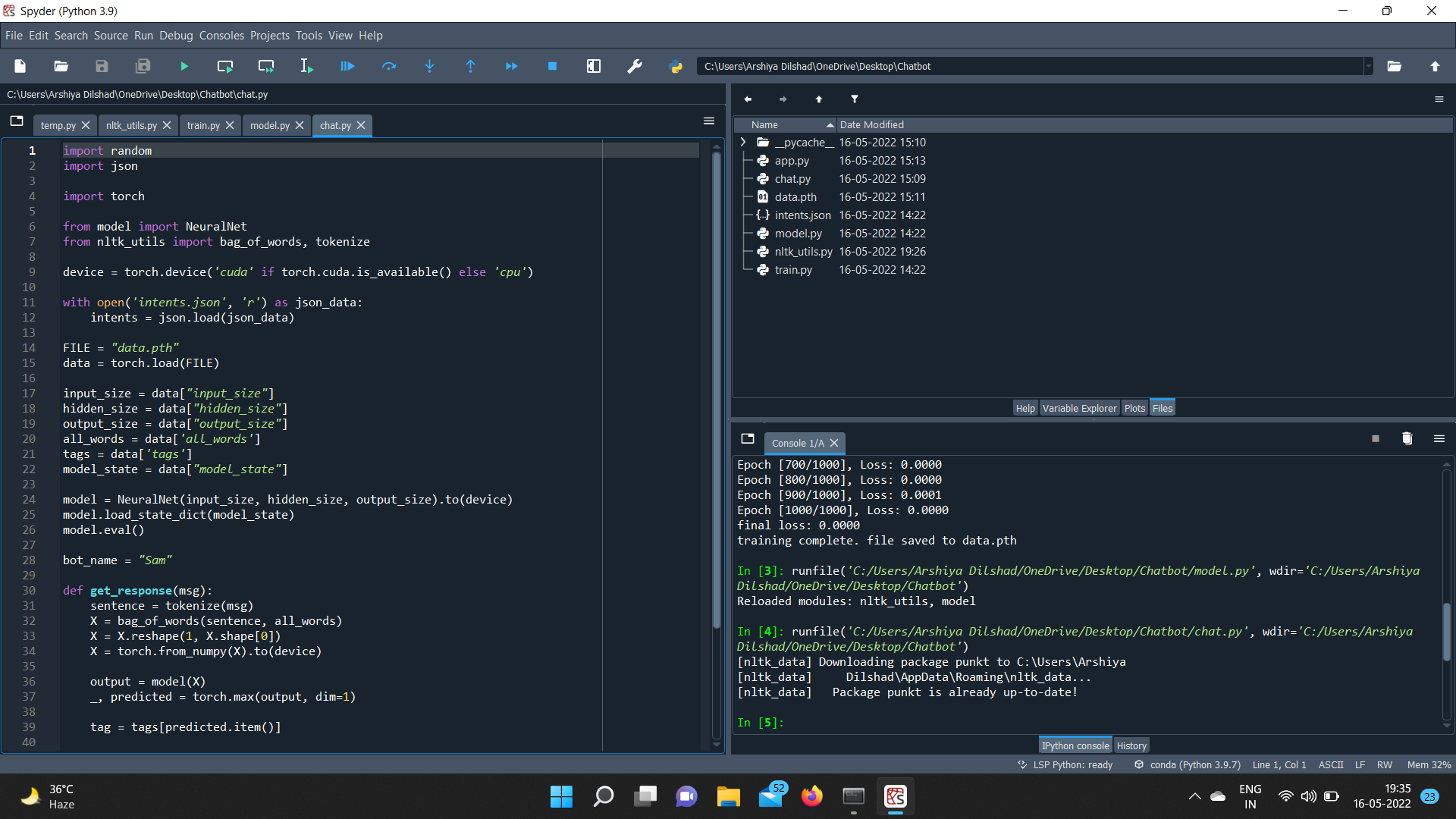


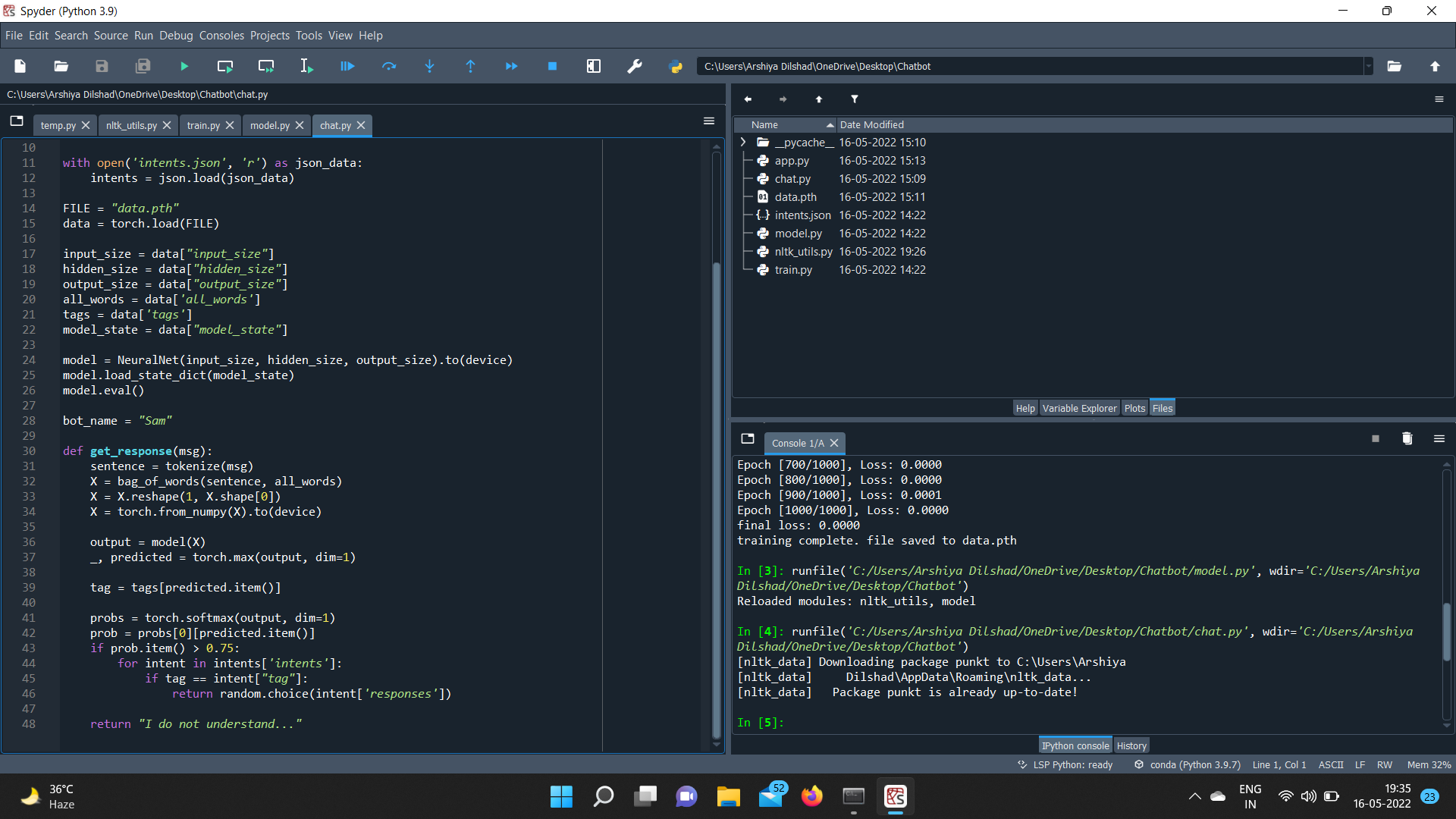


**Step 4:** Then, model.py file was created and ran



**Step 5:** Then, chat.py file was created and run

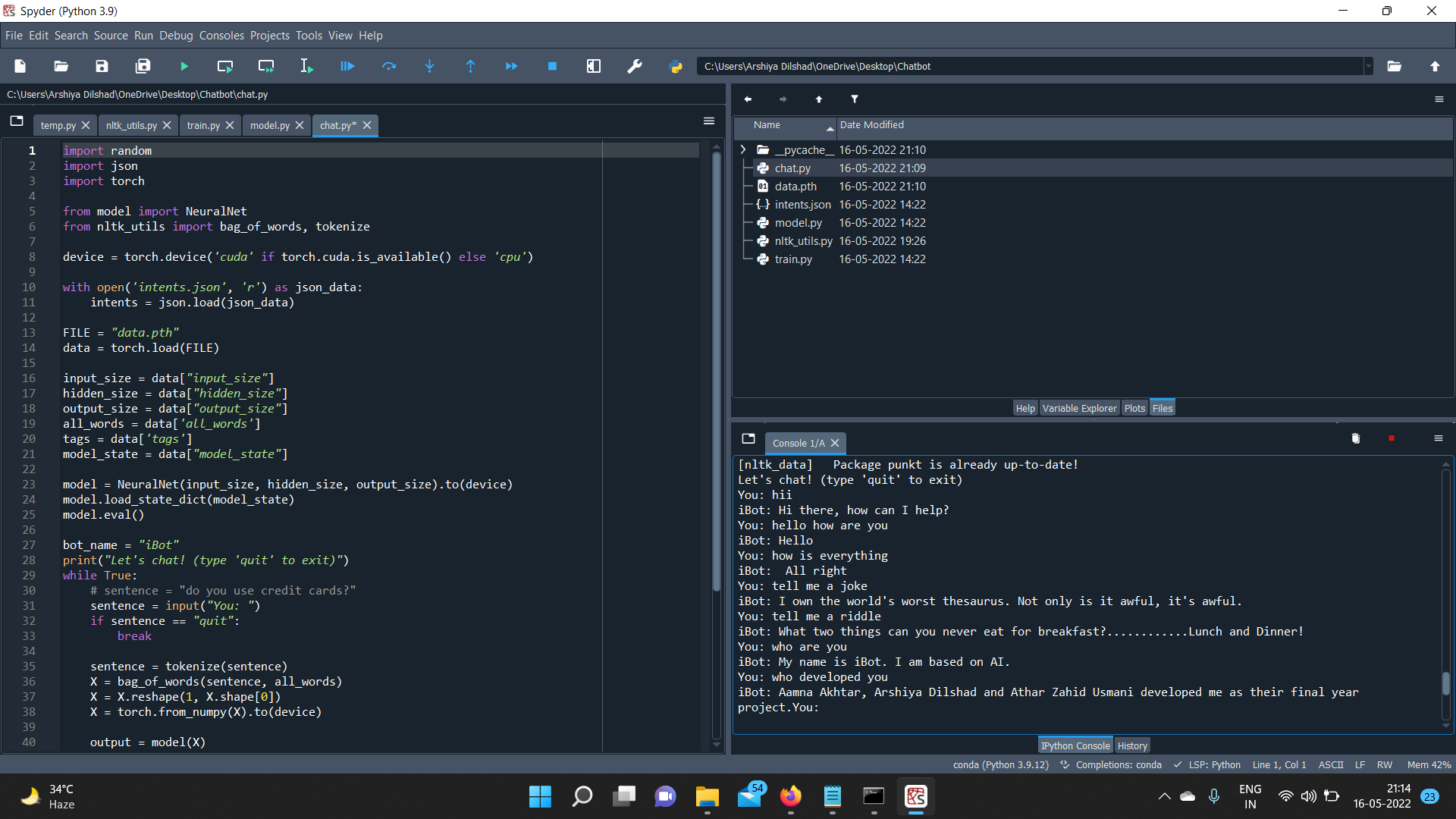


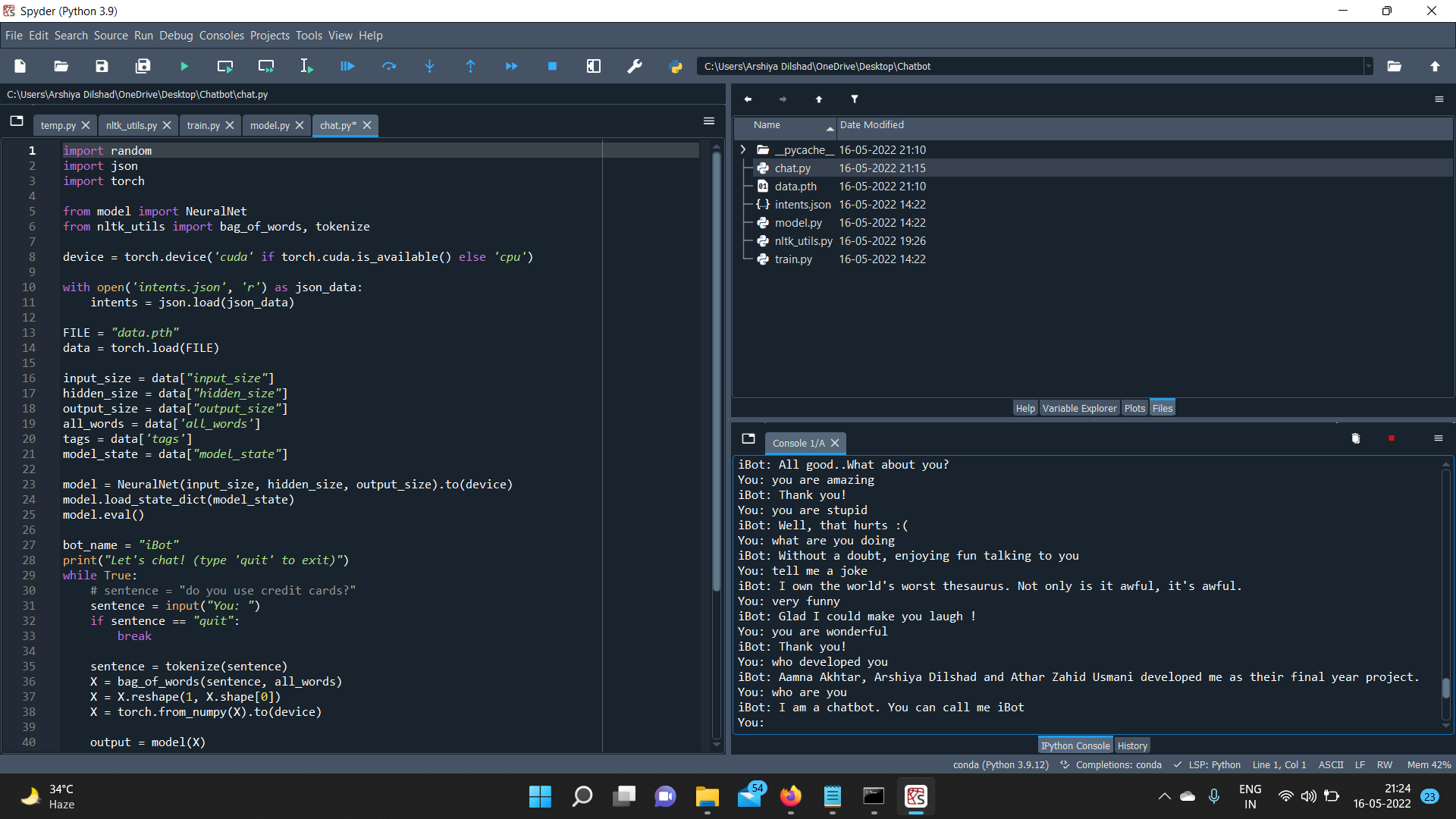


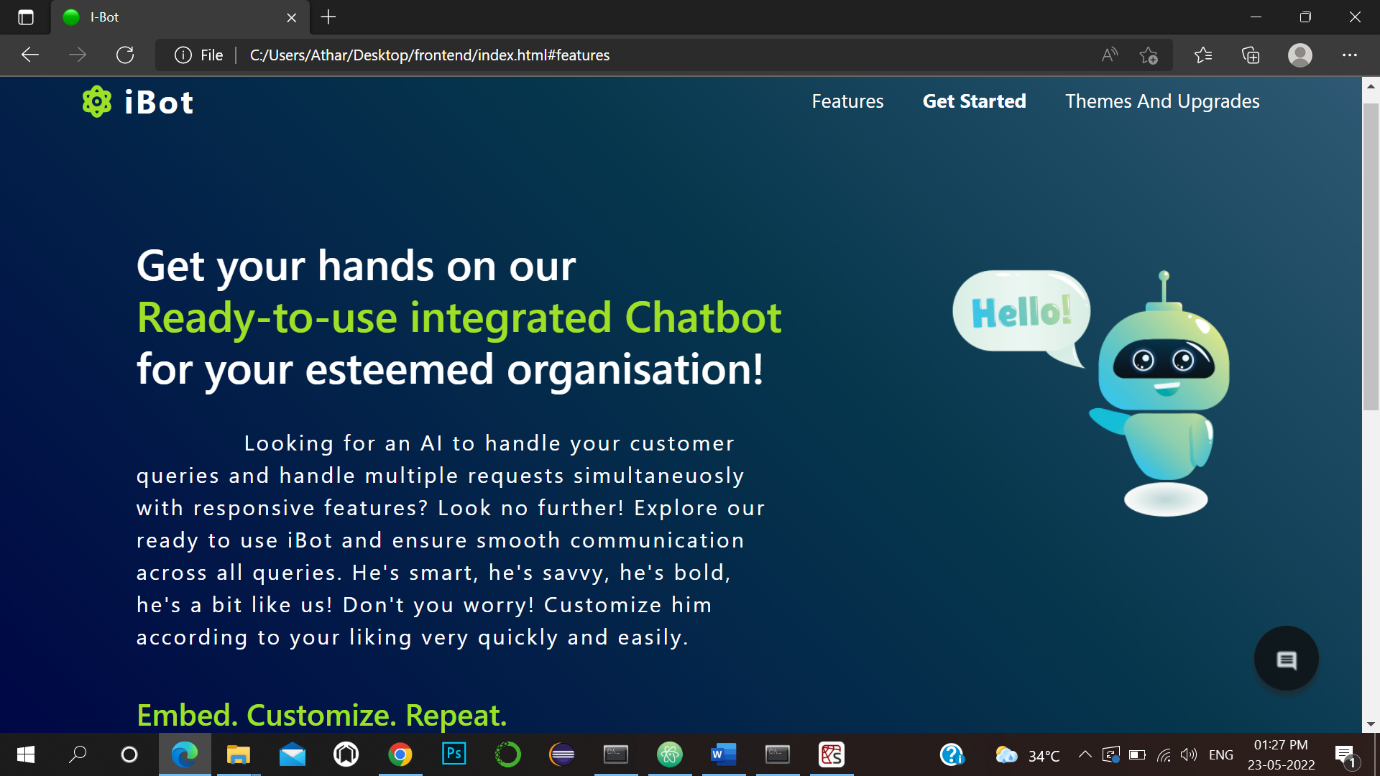
**CHAPTER-5**

**RESULT ANALYSIS AND FUTURE WORK**

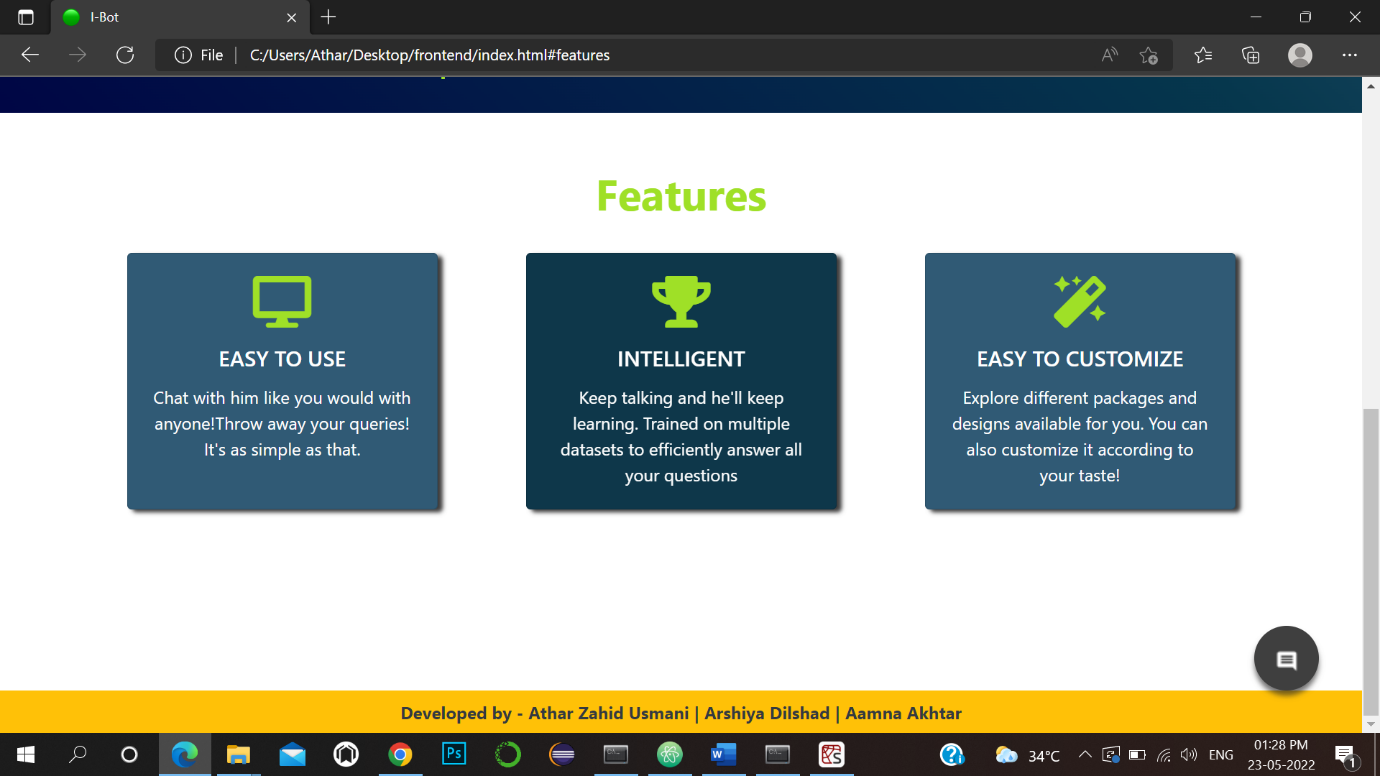
**5.1 Final, snapshots of some of the conversations with iBot**.

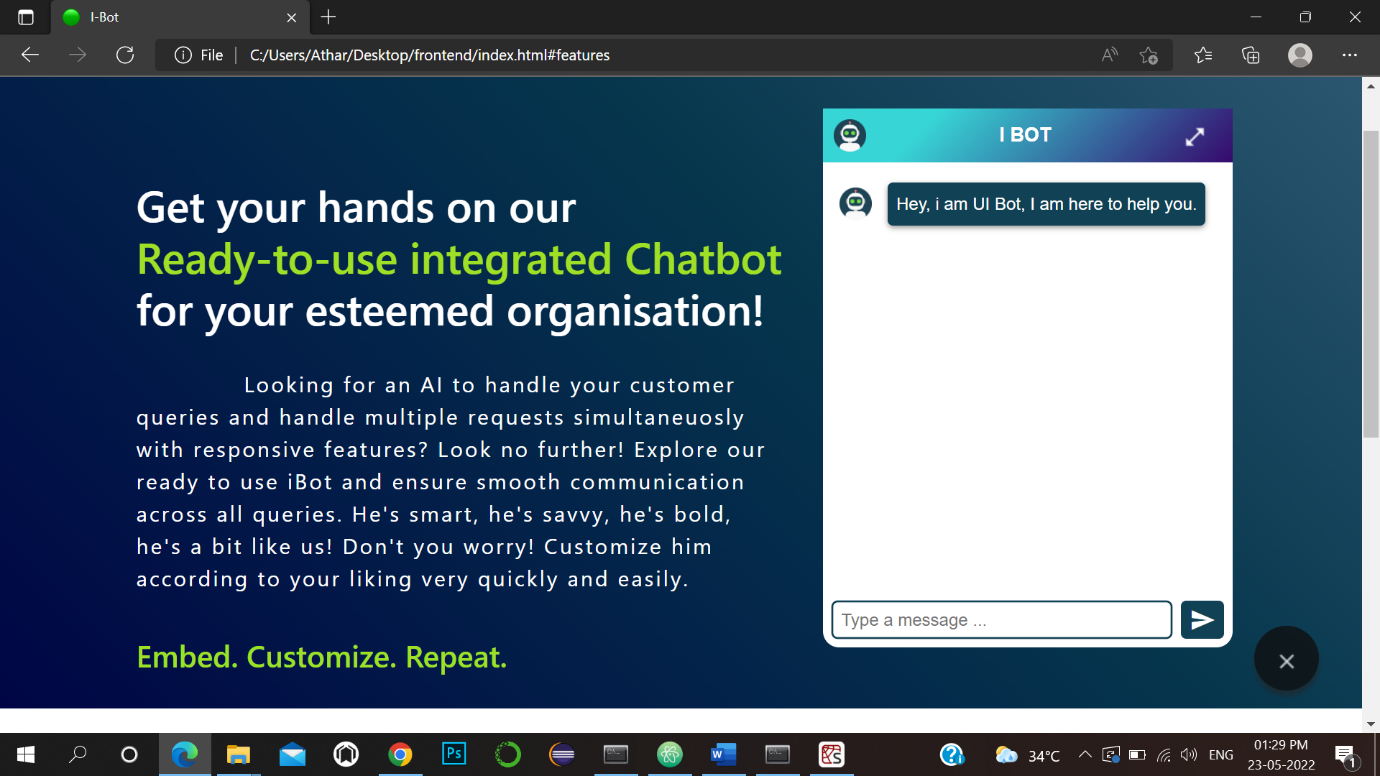


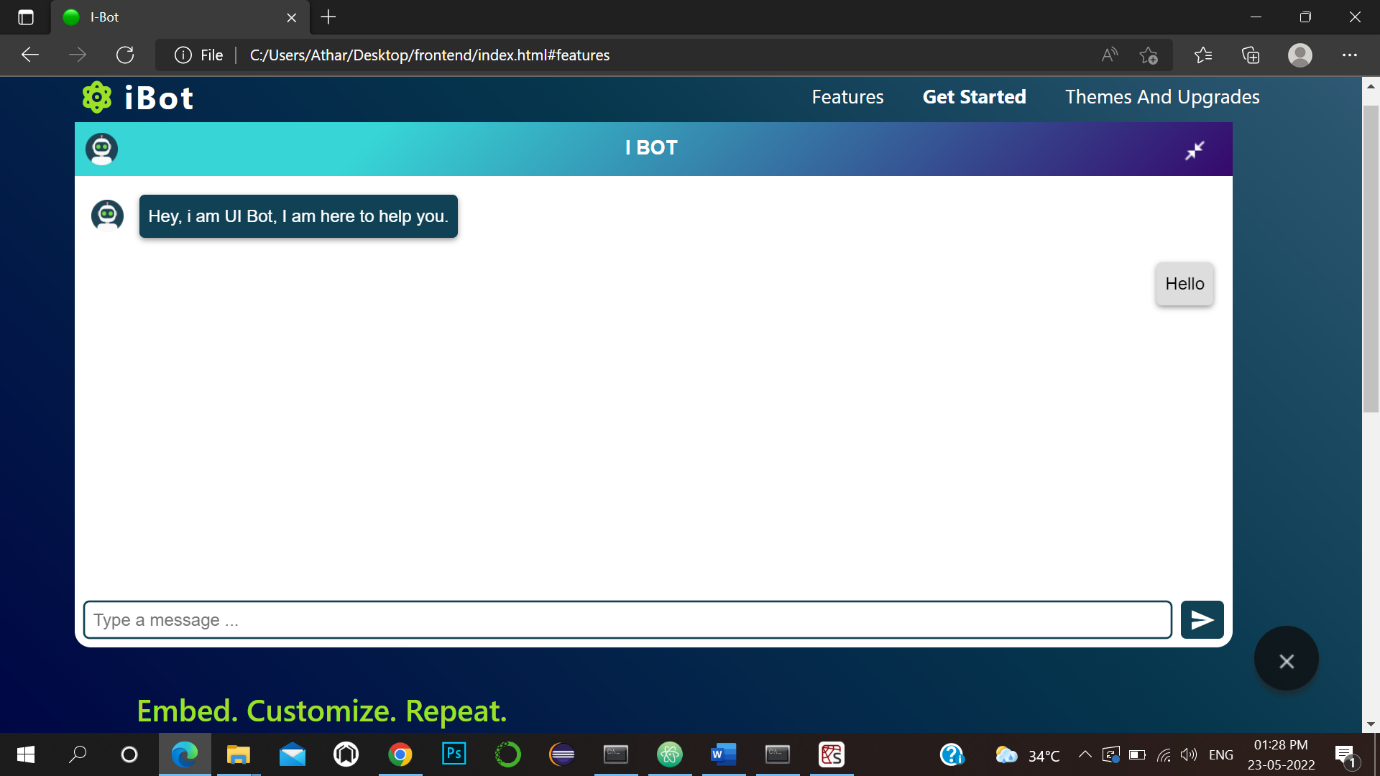


\

\







**5.2 Future Work**

The study of machine learning will continue to increase in the near future and there is great scope of machine learning in the present and future.

Machine learning is in a trend and will continue to be in trend according to experts and data scientists due to continuous increase in the production of tremendous amount of data and this data will continue to increase as day by day we are becoming more and more dependent on internet.

The field of Machine learning is experiencing exponential growth today, especially in the subject of computer vision. Today, the error rate in humans is only 3% in computer vision. This means computers are already better at recognizing and analyzing images than humans. What an amazing feat! Decades ago, computers were hunks of machinery the size of a room; today, they can perceive the world around us in ways that we never thought possible.

We have been able to reach our goal to a great extent. We have made this project not only from python but there is a need for us learn web development also and through this project we realized making a project in a team is the best way to gain knowledge and do task easily and efficiently and projects are the best way to enhance skills.

Undertaking a project can be challenging and exciting. It is challenging because a tremendous amount of self-discipline, time and effort needs to be put into it. It is exciting because a successful project rewards with great satisfaction and experiential learning.

The project requires the amalgamation of different kinds of skills: problem solving, studying and communication, both written and spoken. It stretches our ability to limits you never thought possible.

**Deep customer insights to drive chatbot behaviour:**

Chatbots are becoming more conversational to communicate effectively and the next step is to improve user experience. Sentiment analysis is important to train chatbots with more human alike capabilities.

It is not just about providing an effective response, but also need to create a delightful customer experience. With the help of sentiment analysis, bots could understand whether the conversation was going well and respond to customer emotions accordingly. So, our next step would be to add emotions and emojis to our iBot.

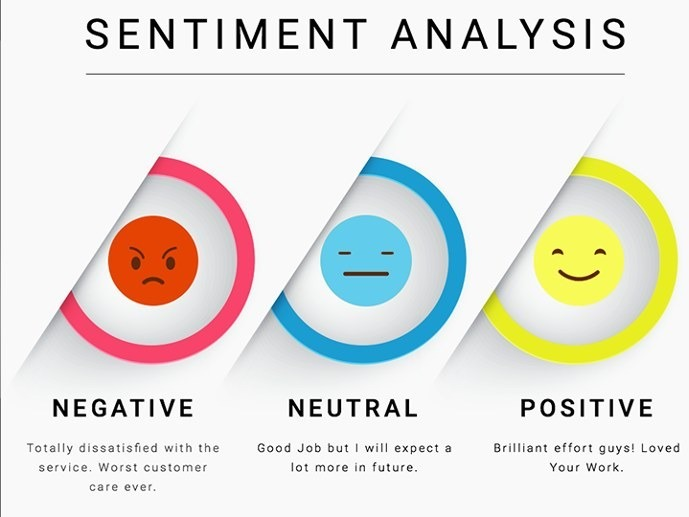


Figure 13 – Sentiment analysis

**Voice bots are becoming mainstream:**

Voice is the next big thing!

Users are already used to starting their days with “Ok Google, what’s in my calendar today?”. According to Forbes, “More than 50% of all searches will be voice-driven”. It is a greatly emerging conversational banking trend.

It’s all about facilitating the user with a seamless experience with your business, and can achieve that by voice-driven chatbots.

Why are voice bots gaining popularity?

Text can be monotonous at times while voice bots engage customers with automated, intelligence-based communication.

Voice-enabled bot, we can provide reliable data insights to our customers. It also helps in giving correct real-time information. We would also like to add voice-enabled feature in our bot, in near future.

**CHAPTER 6**

**CONCLUSION**

The rise and popularity of chatbots can clearly understood by the fact that almost every webpage has an integrated bot. With this in mind the data gathered from testing the chatbot justifies the recent growth and demand for companies wanting to integrate a chatbot. It was determined that chatbots perform at a very high standard and provide reliable and rapid responses to users compared to that of traditional

methods. The average time spent interacting with the chatbot is very low as it provides an

efficient way for users to manage their banking. The low interaction time reflects the high

understanding and speech recognition rates, offered through the adoption of conversational

user interfaces thus allowing users to freely interact with the chatbot to meet the demands of

modern life. The chatbot has proven to fulfil the demand of users wanting instant access and

availability information and services

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE.NO.** | **FIGURE NAME** | **PAGE NO.** |
| 1. | Chatbot | 1 |
| 2. | Illustration | 1 |
| 3. | How chatbot works | 2 |
| 4. | Chatbot workflow | 3 |
| 5. | Data flow diagram | 7 |
| 6. | AI chatbot | 13 |
| 7. | NLP layer | 14 |
| 8. | BOW Model | 16 |
| 9. | ML translation | 17 |
| 10. | Encoder-decoder | 17 |
| 11. | RNN relationship | 18 |
| 12. | Tokenization | 18 |
| 13. | Sentiment Analysis | 55 |

**REFERENCES AND BIBLIOGRAPHY**

**Books**:

1. The complete reference ‘Python’ - **Martin C. Brown**.

2. Machine learning using python - **Maharanjan Pradhan.**

3. Machine learning A-Z: Hands on Python in Data science

**Websites:**

1. https://towardsdatascience.com/

2. https://www.udemy.com/

3. https://colab.research.google.com/

4. https://www.data-flair.training.com

5. https://www.kaggle.com

**Papers/ Journals:**

1. JW Shavlik, T Dietterich, TG Dietterich - 1990

2. ZC Lipton, J Steinhardt - Queue, 2019 - dl.acm.org

3. P Langley - 1987 – Spring

**GitHub Profile :-** <https://github.com/aamnaakhtar/chatbot>

