Institute of Engineering and Rural Technology, Allahabad

Project name: Admission Chatbot

Project team:Shweta
Riya Patel
Anushka Verma

Supervisor:Mr. Rohit Gupta

Abstract

As a result of the rapid technological development and the development of the chatbot concept and the time and effort it can save. Many specialised frameworks have emerged to undertake chatbot creation and development. By relying on artificial intelligence, the chatbot has integrated machine learning within it, and it has become more comprehensive and wider for various technological fields. Therefore, we will create a chatbot for the University's Admission and Registration, the project aims to build a chatbot to facilitate the process of accessing information related to students' inquiries towards admissions, Registration and the university itself. As a conclusion, it lies in answering frequent and common questions by people and providing the answer to these questions at any time the person wants.

Content list

Introduction

Chatbot is a computer program that humans will interact with in natural spoken language and including artificial intelligence techniques such as NLP (Natural language processing) that makes the chatbot more interactive and more reliable. Based on the recent epidemiological situation, the increasing demand and reliance on electronic education has become very difficult to access to the university due to the curfew imposed, and this has led to limited access to information for academics at the university. This project aims to build a chatbot for Admission and Registration to any person who asks about the university, colleges, majors and admission policy.

1.2 Problem Statement

At the start of each academic semester,registration opensforthosewishing to join the university in various disciplines, and telephone calls for admission and registration abound.This leads to an increase in the loads and work for the employees of the Deanship of Admission and Registration as a result of the constant pressure of those wishing to register and their families by flocking to the Deanship, so the employees are not able to answer the phone calls and social media. This often leads to many students who wish to register to be ignored.

1.3 Objectives

- Save effort and time for both the Admission and registration staff and students who wish to enrol.
- Provide detailed information about colleges and majors.
- Easy access to information.

1.4 Scope

- People who wish to enrol I.E.R.T. College.
- Admission and registration staff

Background and Alternatives

2.1 Chatbot :

A chatbot is a computer program that can simulate a conversation or chat with a user in natural language through messaging applications, website, or mobile applications and interact with users according to their input and should be available 24/7. Chatbots are developed and became so popular due to the increased use of smart devices and IoT technology.

2.2 Types of chatbots

a. Base-line chatbot:

It is a chatbot that is based on a database and uses if / then logic to create a conversation flow and that takes a lot of time to ensure the understanding of the question and the answer needed.

b. AI chatbot:

This type of chatbot is more complex than base-line but it is more interactive and personalised and needs big data training to be impressive if the problem is matched to their capabilities.

c. Hybrid Model:

A hybrid approach mixes the Base-line & AI chatbot to make it smart and his behaviour more expected by depending on the database and Ai algorithm to work together.

2.3 How do chatbots work?

Briefly and as mentioned in the definition, humans interact with chatbots. There are two ways to interact with a chatbot:

a. Text

chatbot analyses the inputted text and matches the text with predefined data called intents which are categorised to manage the conversation. The user utterance is tagged with one of these intents, even if what the user says stretches over two or more intents. Most chatbots will take the intent with the highest score and take the conversation down that avenue.

b. Voice

Some chatbots can interact and understand the voice of the user using a set of application programming interfaces (api's) that convert the recorded voice to the language and then convert the voice to words of that language and then deal with the transformed text as mentioned above.

2.4 Options to build a chatbot:

a. From Scratch

At first we have to identify the opportunities for our chatbot and decide its field and scope to achieve efficiency and accuracy. and a precise understanding of the customer needs is required to solve the operational challenges. Then the design of the bot comes to be a significant stage to decide the user engagement with your app or website. and we can categorise chatbot interactions as structured and unstructured interactions.

- ❏ Structured interaction. You already know about this kind of interaction. You know what your customers will ask and can design it easily — it's just like an FAQ section of your app or website . This information will link to your contact information, services, products, etc.
- ❏ Unstructured interaction. The unstructured conversation flow includes freestyle plain text. It's hard to predict what queries will emerge and it looks like an extempore speech competition for your chatbot. The role of AI comes to light here, it decodes the context of the text based on NLP analysis. while the same NLP will provide a voice to the chatbot.

The later choice will need specialised chatbot developers with an understanding of programming languages, machine learning, and AI. We can use some of the code-based frameworks to build and handle the chatbot like wit.ai and api.ai.

b.Using platforms

It is similar to scratch chatbots but the only difference is that you do not have to hire a specialised developer and use the chatbot builder platforms like Chatfuel, Botsify and Rasa, it's not hard or impossible to achieve it. but it's not possible to create a NLP-enabled chatbot that can deal with unstructured data.

System Requirements

3.1. Requirements:

● **Functional Requirements**:

1. The system must provide clear information about Admission policy.
2. The system must provide clear and fully detailed information about university colleges.
3. The system must provide clear and fully detailed information about colleges' programs.
4. The system must provide clear and fully detailed information about colleges' majors.
5. The system should clarify information about the permitted secondary school branch for each major.
6. The system should clarify the minimum GPA in high school for each major.
7. The system should clarify the duration of study for each major.
8. The system should clarify the parallel study policy for each major.
9. The system should provide the graduation plans for each major.
10. The system should provide information about placement tests.
11. The system should provide information about first instalment costs for each major and the credit hour price.

● **Non-Function**

1. The system shall handle multiple users inputs, If two or more students are chatting with the bot, none of the students has to wait too long to be answered by the system.
2. The bot should have a delay in response, to let the student feel like he/she is talking to a human instead of a bot. A little late response from the chatbot makes the student feel as though he is talking to a human.
3. The system should have the appropriate data set. The correct data set is the basis for the chatbot, when the data set is correct and tuned, the chatbot will be trained on it to give the best possible result.
4. The system should have Data Training. Data training mainly depends on the content targeted at Admission and Registration deanship.
5. The system should prevent abusive language.
6. The system should be used on http://reg.ppu.edu/ website.
7. Ability to extend the project to include all colleges.

3.2 Use case diagram

In this part will explain the use case diagram that is used to describe the system classes and relation between them.

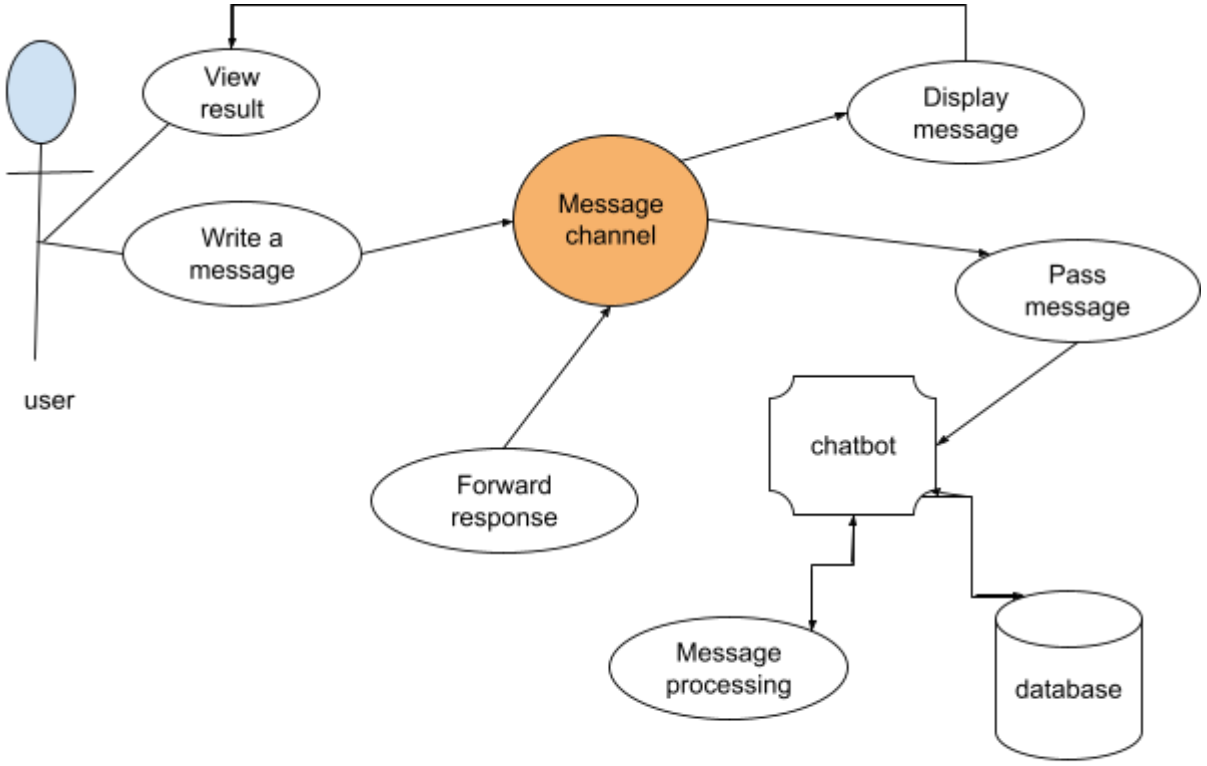
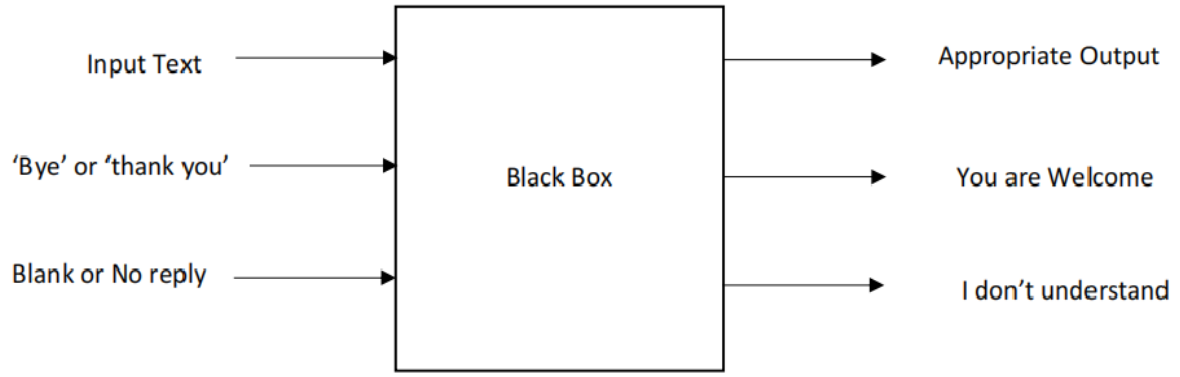
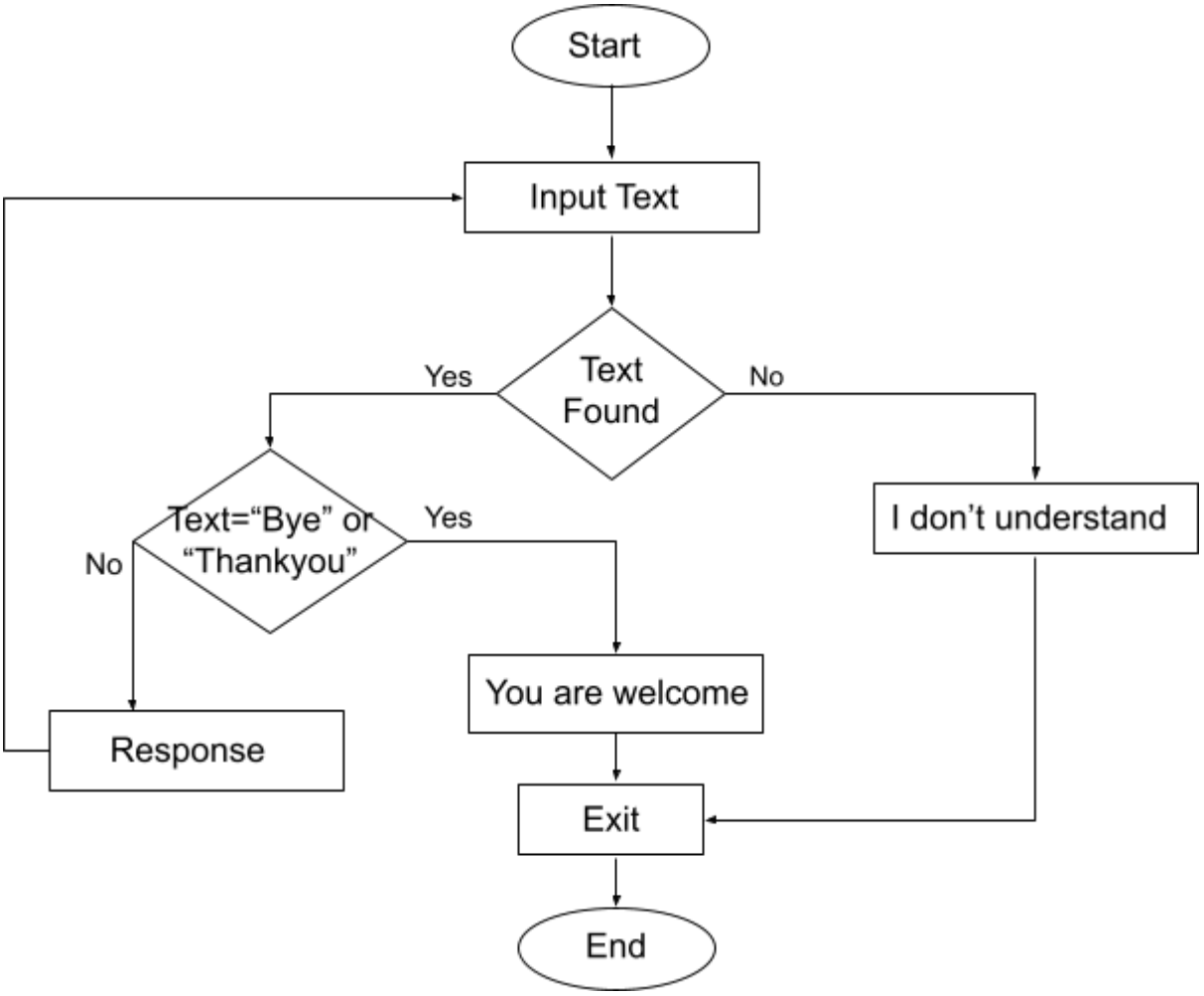


Fig 3.1: Use case diagram



3.2.Black Box Diagram



3.3 Flow Chart Diagram

Models and interfaces

Basics Of nltk

Natural language processing (NLP) is a field that focuses on making natural human language usable by computer programs. NLTK, or Natural Language Toolkit, is a Python package that you can use for NLP. You can use this library for text

classification, sentiment analysis, tokenization, stemming, tagging, parsing, semantic reasoning and many more tasks of Natural Language Processing.

Installation

Run the Python interpreter and type the commands:

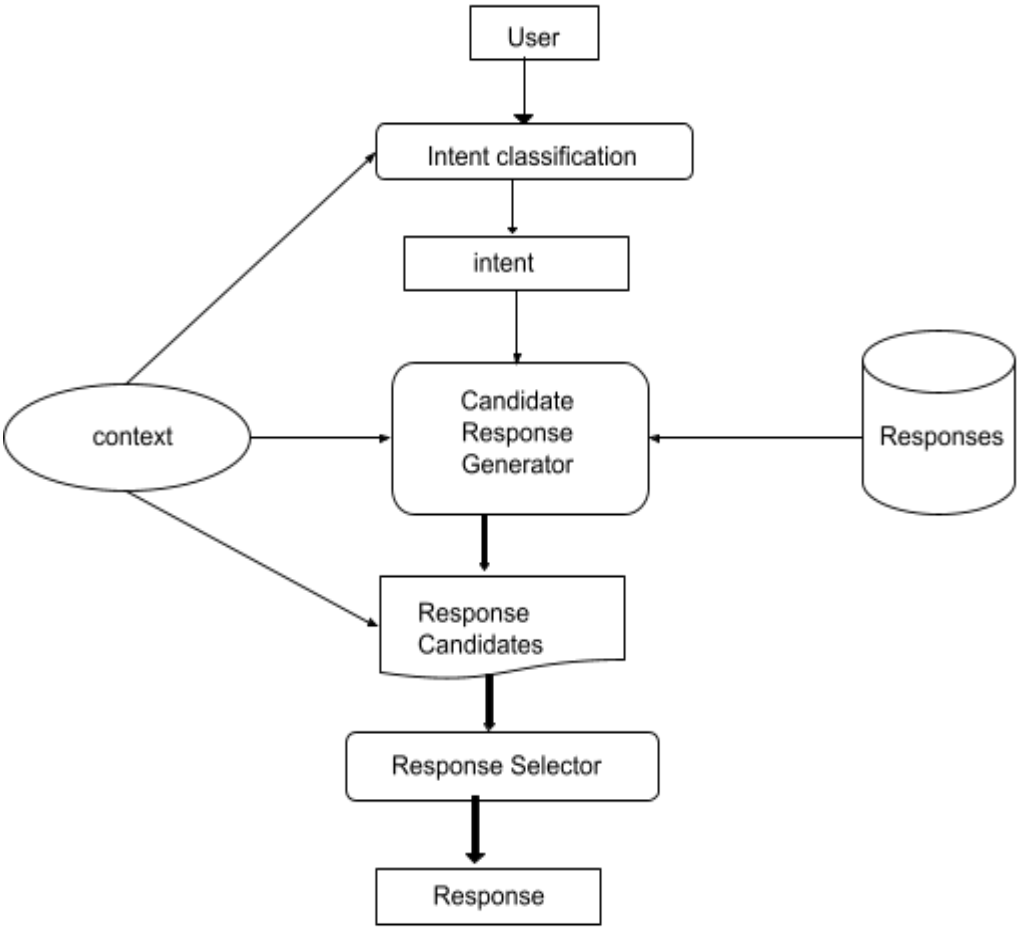
```
>>> import nltk
>>> nltk.download()
```

Test that the data has been installed as follows. (This assumes you downloaded the Brown Corpus):

```
>>> from nltk.corpus import brown
>>> brown.words()
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

4.2 Architectural model

Thediagrambelowprovidesanoverviewof theRasaOpenSourcearchitecture.Thetwopri marycomponents are Natural Language Understanding(NLU) and dialogue management.NLU is the part that handles intent classification,entity extraction,and response retrieval.It's Shown below astheNLU Pipeline because it processes user utterances using NLU model thatisgeneratedbythetrained pipeline.The dialogue management component decides then extraction in a conversation based on the context. This is displayed as the Dialogue Policies in the diagram.



4.3 NLTK library files:

Intents.py: The File contains the Natural language understanding (NLU).This includes intents,which are user goals,and example utterances that represent those intents.TheNLU training data also labels the entities,important keywords, the assistant should extract from the example utterance.

Model.py: A model is a class that represents a table or collection in our DB, and where every attribute of the class is a field of the table or collection. Models are defined in the models.py

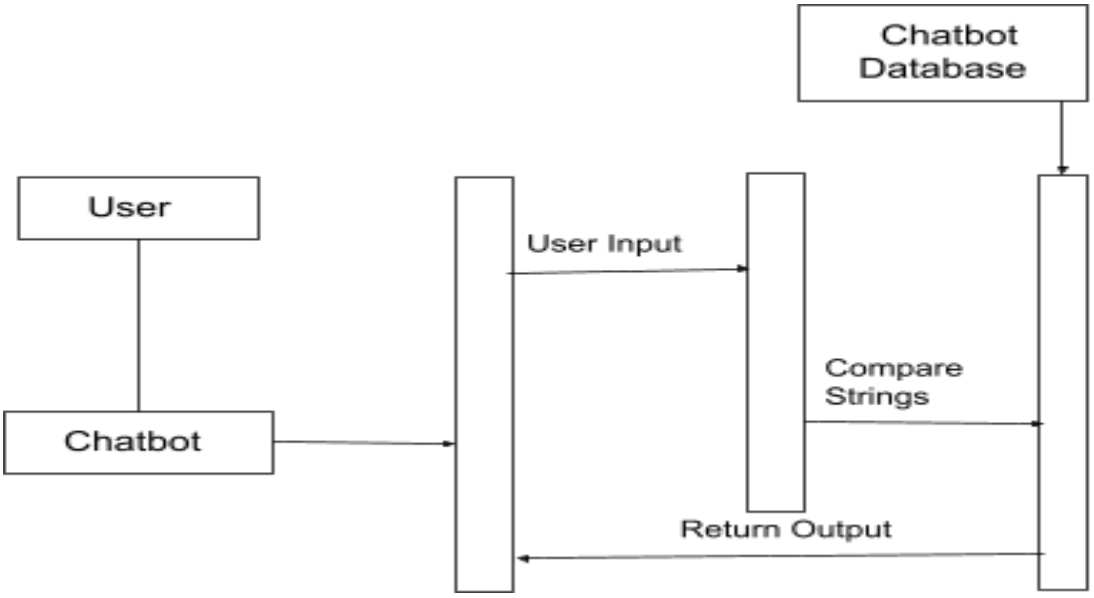
Chat.py:

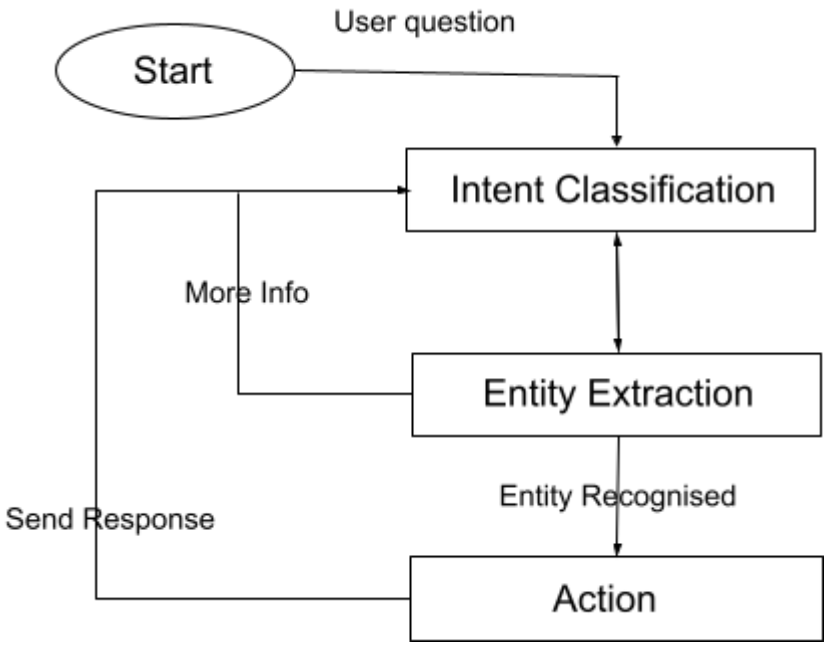
App.py:

Nltk_util.py:

Train.py:

4.4 State and Sequence diagrams: In the following figure, we will show the state diagram and what is the state of the data entry process by the user and where it passes. The bot sends a message to the user asking for more information. When the bot gets enough information to reveal intents and entities, it sends the available data to a special section called Actions. In the Actions section, the bot specifies the appropriate response to be sent to the user after performing matching operations for extracted entities based on the highest corresponding percentage of these entities. After that, the bot sends the resulting information and shows it as a message, we'll show the sequence diagram. In this diagram, we will carry out a complete tracking process for all stages of the system, how it handles and processes user input.. For example, if we ask what's the weather like tomorrow? When sending this question, the framework identifies the intentions through some of the key words in the question, and here the key word is weather. Then they are recognized and likewise the entities are extracted which are tomorrow in this case.





Coding

base.html

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="style.css">

<head>
    <meta charset="UTF-8">
    <title>Chatbot</title>
```

```
</head>
<body>
<div class="container">
    <div class="chatbox">
        <div class="chatbox__support">
            <div class="chatbox__header">
                <div class="chatbox__image--header">
                    <img
src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5
--v1.png" alt="image">
                </div>
                <div class="chatbox__content--header">
                    <h4 class="chatbox__heading--header">Chat support</h4>
                    <p class="chatbox__description--header">Hi! How can I
help you?</p>
                </div>
            </div>
            <div class="chatbox__messages">
                <div></div>
            </div>
            <div class="chatbox__footer">
                <input type="text" placeholder="Write a message...">
                <button class="chatbox__send--footer
send__button">Send</button>
            </div>
        </div>
        <div class="chatbox__button">
            <button><img src="./images/chatbox-icon.svg" /></button>
        </div>
    </div>
</div>


    <script src="./app.js"></script>

</body>
</html>
```

Style.css

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: 'Nunito', sans-serif;
    font-weight: 400;
    font-size: 100%;
    background: #F1F1F1;
}

*, html {
    --primaryGradient: linear-gradient(93.12deg, #581B98 0.52%, #9C1DE7
100%);
    --secondaryGradient: linear-gradient(268.91deg, #581B98 -2.14%, #9C1DE7
99.69%);
    --primaryBoxShadow: 0px 10px 15px rgba(0, 0, 0, 0.1);
    --secondaryBoxShadow: 0px -10px 15px rgba(0, 0, 0, 0.1);
    --primary: #581B98;
}

/* CHATBOX
=============== */
.chatbox {
    position: absolute;
    bottom: 30px;
    right: 30px;
}

/* CONTENT IS CLOSE */
.chatbox__support {
    display: flex;
    flex-direction: column;
    background: #eee;
```

```
    width: 300px;
    height: 350px;
    z-index: -123456;
    opacity: 0;
    transition: all .5s ease-in-out;
}

/* CONTENT ISOPEN */
.chatbox--active {
    transform: translateY(-40px);
    z-index: 123456;
    opacity: 1;

}

/* BUTTON */
.chatbox__button {
    text-align: right;
}

.send__button {
    padding: 6px;
    background: transparent;
    border: none;
    outline: none;
    cursor: pointer;
}


/* HEADER */
.chatbox__header {
    position: sticky;
    top: 0;
    background: orange;
}

/* MESSAGES */
.chatbox__messages {
    margin-top: auto;
```

```
    display: flex;
    overflow-y: scroll;
    flex-direction: column-reverse;
}

.messages__item {
    background: orange;
    max-width: 60.6%;
    width: fit-content;
}

.messages__item--operator {
    margin-left: auto;
}

.messages__item--visitor {
    margin-right: auto;
}

/* FOOTER */
.chatbox__footer {
    position: sticky;
    bottom: 0;
}

.chatbox__support {
    background: #f9f9f9;
    height: 450px;
    width: 350px;
    box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
}

/* HEADER */
.chatbox__header {
    background: var(--primaryGradient);
    display: flex;
    flex-direction: row;
```

```
    align-items: center;
    justify-content: center;
    padding: 15px 20px;
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    box-shadow: var(--primaryBoxShadow);
}

.chatbox__image--header {
    margin-right: 10px;
}

.chatbox__heading--header {
    font-size: 1.2rem;
    color: white;
}

.chatbox__description--header {
    font-size: .9rem;
    color: white;
}

/* Messages */
.chatbox__messages {
    padding: 0 20px;
}

.messages__item {
    margin-top: 10px;
    background: #E0E0E0;
    padding: 8px 12px;
    max-width: 70%;
}

.messages__item--visitor,
.messages__item--typing {
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    border-bottom-right-radius: 20px;
```

```
}

.messages__item--operator {
    border-top-left-radius: 20px;
    border-top-right-radius: 20px;
    border-bottom-left-radius: 20px;
    background: var(--primary);
    color: white;
}

/* FOOTER */
.chatbox__footer {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: space-between;
    padding: 20px 20px;
    background: var(--secondaryGradient);
    box-shadow: var(--secondaryBoxShadow);
    border-bottom-right-radius: 10px;
    border-bottom-left-radius: 10px;
    margin-top: 20px;
}

.chatbox__footer input {
    width: 80%;
    border: none;
    padding: 10px 10px;
    border-radius: 30px;
    text-align: left;
}

.chatbox__send--footer {
    color: white;
}

.chatbox__button button,
.chatbox__button button:focus,
.chatbox__button button:visited {
```

```
    padding: 10px;
    background: white;
    border: none;
    outline: none;
    border-top-left-radius: 50px;
    border-top-right-radius: 50px;
    border-bottom-left-radius: 50px;
    box-shadow: 0px 10px 15px rgba(0, 0, 0, 0.1);
    cursor: pointer;
}
```

App.js

```
class Chatbox {
    constructor() {
        this.args = {
            openButton: document.querySelector('.chatbox__button'),
            chatBox: document.querySelector('.chatbox__support'),
            sendButton: document.querySelector('.send__button')
        }

        this.state = false;
        this.messages = [];
    }

    display() {
        const {openButton, chatBox, sendButton} = this.args;

        openButton.addEventListener('click', () =>
this.toggleState(chatBox))

        sendButton.addEventListener('click', () =>
this.onSendButton(chatBox))

        const node = chatBox.querySelector('input');
        node.addEventListener("keyup", ({key}) => {
            if (key === "Enter") {
                this.onSendButton(chatBox)
```

```
        }
    })
}

toggleState(chatbox) {
    this.state = !this.state;

    // show or hides the box
    if(this.state) {
        chatbox.classList.add('chatbox--active')
    } else {
        chatbox.classList.remove('chatbox--active')
    }
}

onSendButton(chatbox) {
    var textField = chatbox.querySelector('input');
    let text1 = textField.value
    if (text1 === "") {
        return;
    }

    let msg1 = { name: "User", message: text1 }
    this.messages.push(msg1);

    fetch('http://127.0.0.1:5000/predict', {
        method: 'POST',
        body: JSON.stringify({ message: text1 }),
        mode: 'cors',
        headers: {
          'Content-Type': 'application/json'
        },
      })
      .then(r => r.json())
      .then(r => {
        let msg2 = { name: "Sam", message: r.answer };
        this.messages.push(msg2);
        this.updateChatText(chatbox)
        textField.value = ''
```

```
        }).catch((error) => {
            console.error('Error:', error);
            this.updateChatText(chatbox)
            textField.value = ''
          });
    }

    updateChatText(chatbox) {
        var html = '';
        this.messages.slice().reverse().forEach(function(item, index) {
            if (item.name === "Sam")
            {
                html += '<div class="messages__item
messages__item--visitor">' + item.message + '</div>'
            }
            else
            {
                html += '<div class="messages__item
messages__item--operator">' + item.message + '</div>'
            }
          });

        const chatmessage = chatbox.querySelector('.chatbox__messages');
        chatmessage.innerHTML = html;
    }
}


const chatbox = new Chatbox();
chatbox.display();
```

App.py

```
from email import message
import json
from urllib import response
from flask import Flask, render_template, request, jsonify
```

```
from flask_cors import CORS

from chat import get_response

app = Flask(__name__)
CORS(app)


@app.post("/predict")
def predict():
    text = request.get_json().get("message")
    # todo: check if text is valid
    response = get_response(text)
    message = {"answer": response}
    return jsonify(message)

if __name__ == "__main__":
    app.run(debug = True)
```

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 = "Sam"

def get_response(msg):
    sentence = tokenize(msg)
    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"]:
                return random.choice(intent['responses'])

    return "I do not understand..."


if __name__ == "__main__":
    print("Let's chat! (type 'quit' to exit)")
    while True:
        # sentence = "do you use credit cards?"
        sentence = input("You: ")
        if sentence == "quit":
```

```
            break

        resp = get_response(sentence)
        print(resp)
```

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
```

Nltk_util.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
```

Conclusion

This bot was built to respond to the inquiries of the Tawjihi students regarding each of the university's faculties and their specialisations, with extracted information for each specialisation, familiarising students with the level exams that students submit about their enrollment in the university, introducing the educational qualification diploma program and the mechanism for joining it. Giving students notes on the electronic enrollment application package, the locations of approved banks, and how to fill out the application. Introduce Bagrut students to the conditions and notes that must be taken into account in the event of joining Palestine Polytechnic University and the mechanism for calculating grades. Introduce students to the procedures followed to reserve a seat and what documents are required after the student is accepted. Introducing students to the system of transferring to Palestine Polytechnic University from another university on the undergraduate system. Informing students of the university's teaching system and language. Introducing students to the student exchange system with other universities. Introducing students to the system of grants, exemptions, and financial aid provided to students. Informing students of cases in which the student loses his university seat. Introducing students to the instalment refund system for new students and its conditions.

Challenges:

We faced many challenges, and the biggest challenge during the work was the arrest of my colleague Ali Jboor, a month before the deadline for the project. Dealing with the framework is one of the biggest challenges, as it is new and not widespread. And also the challenge of learning the necessary Python language to complete the construction of this bot. We can say that the current health situation contributed to creating communication problems between team members and problems related to slow internet and power outages during work. We also mentioned the challenge we faced, so we divided the work into two parts, part for the admin panel and a part to follow up on the work on the bot. Unfortunately, Ali was arrested and I had to do the project alone in his absence, especially after we agreed on a day to share the results of the work, but that day Ali was arrested.

Interesting decisions:

1- The idea of the project itself was very interesting. We considered this project and set it as a challenge to our abilities and ourselves. To be based on learning to use and implement programs using a non-renowned framework for a project that is the most important in a university student's career.

2- The idea of changing the operating system used to run the bot was one of the most crucial decisions in the workflow of the project. Windows was the best and most worthy on paper, according to the sources. But there was one problem that we

had encountered for such a long time that made an operating system change necessary.

Recommendations:

It is possible to modify and increase the efficiency of the bot to the fullest extent if the time factor and the human factor are available. Unfortunately, we were not able to deliver the bot to the maximum extent that we drew and expected due to the circumstances that befell us. Additional matters necessary for students related to registration, student status, and forms to official documents can be added. Future improvements: The project scope may be expanded to include all corners of the university, including faculties and deanships of registration and follow-up of all matters that the student is interested in during their academic life. The ability to communicate using voice messages.

References

1. "Chatbots - Artificial Solutions." Chatbots |

Artificial Solutions: Conversational AI Platform for Enterprise - Teneo

2. "How Do Bots and Chatbots Work?"

https://www.cxtoday.com/contact-centre/how-do-bots-and-chatbots-work/

3.” NLTK “

https://www.nltk.org/

4. “Python “

https://www.python.org/

5. “torchvision “

https://pytorch.org/vision/stable/index.html