VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : B.Tech(CSE- AIR)

Course Title : Fundamentals of Artificial Intelligence

Course Code : CSE2039

Slot : F1

Title: FFCS Chatbot

Team Members:

Abraham Kuriakose- 20BRS1119

Lavish Jain R- 20BRS1120

Sai Teja Bandaru- 20BRS1129

Rohan Jacob John- 20BRS1159

Faculty: Dr. R. Jothi

Sign:

Date: 29/4/22

FFCS CHATBOT

by

ABRAHAM KURIAKOSE- 20BRS1119
LAVISH JAIN- 20BRS1120
SAI TEJA BANDARU- 20BRS1129
ROHAN JACOB JOHN- 20BRS1159

A project report submitted to

Dr. JOTHI R

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

in partial fulfilment of the requirements for the course of

CSE2039 - FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE

in

B.TECH COMPUTER SCIENCE ARTIFICIAL INTELLIGENCE AND ROBOTICS

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

APRIL 2022

BONAFIDE CERTIFICATE

Certified that this project report entitled "**VIT FFCS Chatbot"** is a bonafide work of **Abraham Kuriakose – 20BRS1119, Lavish Jain - 20BRS1120 , Sai Teja Bandaru -20BRS1129 and Rohan Jacob John - 20BRS1159** who carried out the Project work under my supervision and guidance for **CSE2039- Fundamentals Of Artificial Intelligence.**

Dr. JOTHI R

Assistant Professor

School of Computer Science and Engineering (SCOPE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

A chatbot, is a program that can act and have a conversation like a normal human being. The chatbot discussed in this report mainly intends to help students of Vellore Institute of Technology during their Fully Flexible Credit System course registration. The chatbot is designed to assist the student throughout the whole process of the course registration. The main objective of the chatbot we have designed is to assist the students in making the right decisions in the most important part of their career. The chatbot will provide various insights into what each term in FFCS means as well as how to plan and choose the right slots in making the most suitable timetable,

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Jothi R,** Associate Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Ganesan R , Professor and Dean, VIT Chennai**, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department **Dr. Rekha D** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Abraham **Lavish**

NAME WITH SIGNATURE **NAME WITH SIGNATURE**

Rohan **Sai Teja**

NAME WITH SIGNATURE **NAME WITH SIGNATURE**

TABLE OF CONTENTS

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

- Design a fully functioning chatbot for FFCS course registration.
- Provide insight on the procedure for course registration.
- Being able to answer the most frequently asked questions.
- Providing tips to students for the course registration.
- To ease the pressure on the students.
- To make the student's life easier.
- Providing relevant information to the questions asked.

1.2 APPLICATIONS

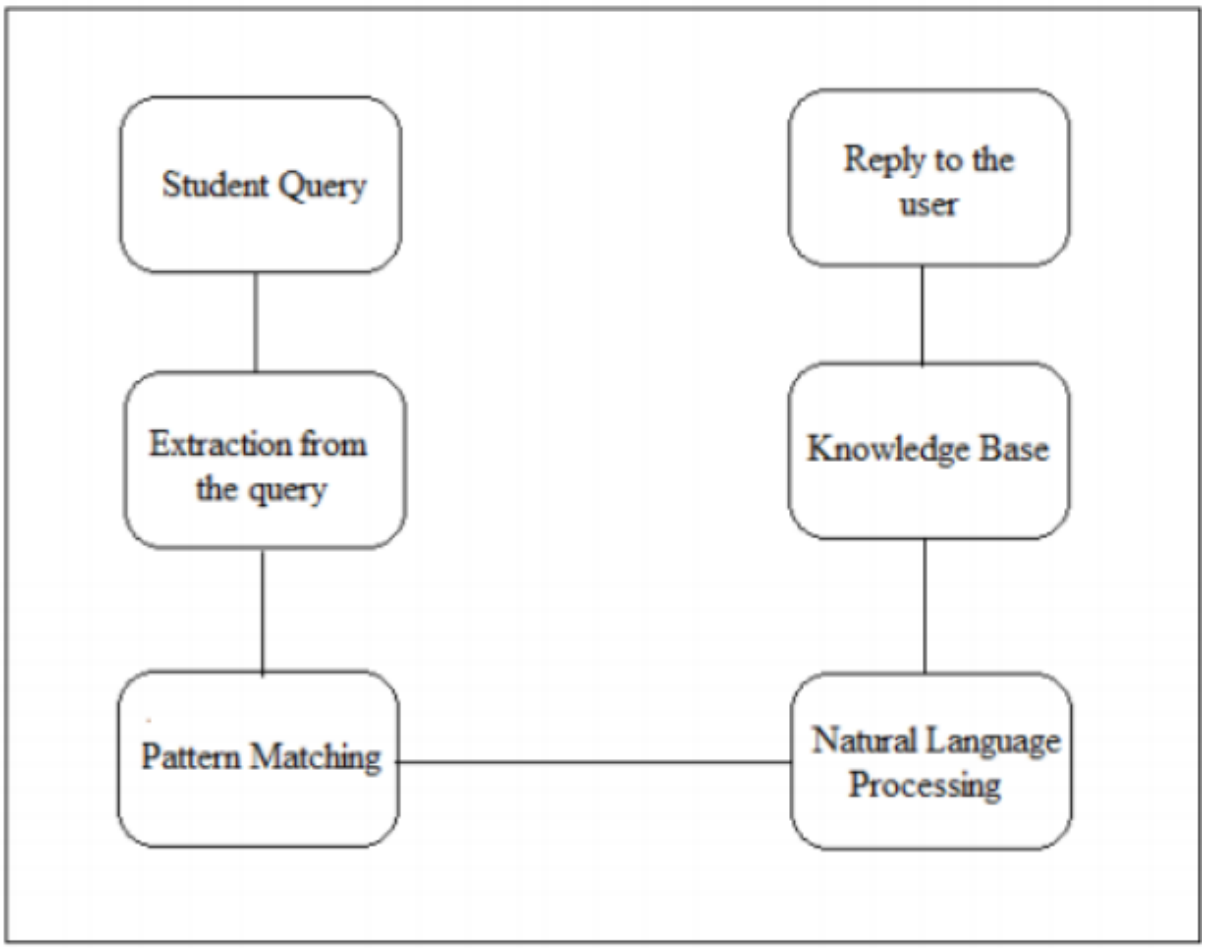
- Creating TimeTables
- Solving FFCS doubts

1.3 FEATURES

- Customized customer interactions
- Trainable intelligence to reduce resolution time
- Extensible integrations with 3rd-party applications
- Easy human-takeover for live support

2. METHODOLOGY

The system is first trained using a pre-defined dataset (“Intents”). The dataset includes basic greetings and all the queries that the user might possibly have for FFCS registration. Then, when a user asks a query, the query is processed and it finds the most matched sequence in the given dataset. This is done by lemmatizing and keyword extraction which is then processed using pattern matching and Natural Language Processing(NLP). The system then chooses the best possible reply and outputs it to the user.



3. SOFTWARE CODING AND ANALYSIS

- train chat_bot

```
import nltk
from nltk.stem import WordNetLemmatizer


lemmatizer = WordNetLemmatizer()
import json
import pickle


import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD
import random


words = []
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)


for intent in intents['intents']:
    for pattern in intent['patterns']:


        # tokenize each word
```

```
            w = nltk.word_tokenize(pattern)
            words.extend(w)
            # add documents in the corpus
            documents.append((w, intent['tag']))

            # add to our classes list
            if intent['tag'] not in classes:
                classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w
not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print(len(documents), "documents")
# classes = intents
print(len(classes), "classes", classes)
# words = all words, vocabulary
print(len(words), "unique lemmatized words", words)

pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))

# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
```

```
    # initialize our bag of words

    bag = []

    # list of tokenized words for the pattern

    pattern_words = doc[0]

    # lemmatize each word - create base word, in attempt to
represent related words

    pattern_words = [lemmatizer.lemmatize(word.lower()) for word
in pattern_words]

    # create our bag of words array with 1, if word match found
in current pattern

    for w in words:

        bag.append(1) if w in pattern_words else bag.append(0)


    # output is a '0' for each tag and '1' for current tag (for
each pattern)

    output_row = list(output_empty)

    output_row[classes.index(doc[1])] = 1


    training.append([bag, output_row])

# shuffle our features and turn into np.array

random.shuffle(training)

training = np.array(training)

# create train and test lists. X - patterns, Y - intents

train_x = list(training[:, 0])

train_y = list(training[:, 1])

print("Training data created")


# Create model - 3 layers. First layer 128 neurons, second
layer 64 neurons and 3rd output layer contains number of
neurons

# equal to number of intents to predict output intent with
softmax

model = Sequential()
```

```
model.add(Dense(128, input_shape=(len(train_x[0]),),
activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(len(train_y[0]), activation='softmax'))


# Compile model. Stochastic gradient descent with Nesterov
accelerated gradient gives good results for this model

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

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


# fitting and saving the model

hist = model.fit(np.array(train_x), np.array(train_y),
epochs=200, batch_size=5, verbose=1)

model.save('chatbot_model.h5', hist)


print("model created")
```

- chatgui.py

```
import nltk

from nltk.stem import WordNetLemmatizer


lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np


from keras.models import load_model


model = load_model('chatbot_model.h5')

import json
```

```
import random


intents = json.loads(open('intents.json').read())

words = pickle.load(open('words.pkl', 'rb'))

classes = pickle.load(open('classes.pkl', 'rb'))



def clean_up_sentence(sentence):
   # tokenize the pattern - split words into array
   sentence_words = nltk.word_tokenize(sentence)
   # stem each word - create short form for word
   sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
   return sentence_words


# return bag of words array: 0 or 1 for each word in the bag that
exists in the sentence


def bow(sentence, words, show_details=True):
   # tokenize the pattern
   sentence_words = clean_up_sentence(sentence)
   # bag of words - matrix of N words, vocabulary matrix
   bag = [0] * len(words)
   for s in sentence_words:
       for i, w in enumerate(words):
           if w == s:
               # assign 1 if current word is in the vocabulary
position
               bag[i] = 1
               if show_details:
                   print("found in bag: %s" % w)
```

```
    return (np.array(bag))



def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words, show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r >
ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability":
str(r[1])})
    return return_list



def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if (i['tag'] == tag):
            result = random.choice(i['responses'])
            break
    return result



def chatbot_response(msg):
    ints = predict_class(msg, model)
```

```
    res = getResponse(ints, intents)
    return res



# Creating GUI with tkinter
import tkinter
from tkinter import *



def send():
    msg = EntryBox.get("1.0", 'end-1c').strip()
    EntryBox.delete("0.0", END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)



base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)
```

```
# Create Chat window

ChatLog = Text(base, bd=0, bg="white", height="8", width="50",
font="Arial", )


ChatLog.config(state=DISABLED)


# Bind scrollbar to Chat window

scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")

ChatLog['yscrollcommand'] = scrollbar.set


# Create Button to send message

SendButton = Button(base, font=("Verdana", 12, 'bold'), text="Send",
width="12", height=5,

                    bd=0, bg="#32de97", activebackground="#3c9d9b",
fg='#ffffff',

                    command=send)


# Create the box to enter message

EntryBox = Text(base, bd=0, bg="white", width="29", height="5",
font="Arial")

# EntryBox.bind("<Return>", send)



# Place all components on the screen

scrollbar.place(x=376, y=6, height=386)

ChatLog.place(x=6, y=6, height=386, width=370)

EntryBox.place(x=128, y=401, height=90, width=265)

SendButton.place(x=6, y=401, height=90)


base.mainloop()
```

- intents.json

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi there", "How are you", "Is anyone
there?","Hey","Hola", "Hello", "Good day", "Wassup"],
         "responses": ["Hi there, how can I help?"],
         "context": [""]
        },
        {"tag": "Me",
         "patterns": ["Who are you?", "who is it?", "what is your
name?", "what are you"],
         "responses": ["Hi there, I am LARS. Nice to meet You!"],
         "context": [""]
        },

        {"tag": "goodbye",
         "patterns": ["Bye", "See you later", "Goodbye", "Nice
chatting to you, bye", "Till next time", "Ok"],
         "responses": ["See you!", "Have a nice day", "Bye! Come back
again soon."],
         "context": [""]
        },
        {"tag": "thanks",
         "patterns": ["Thanks", "Thank you", "That's helpful",
"Awesome, thanks", "Thanks for helping me"],
         "responses": ["Happy to help!", "Any time!", "My pleasure"],
         "context": [""]
        },
        {"tag": "noanswer",
         "patterns": ["","  "],
         "responses": ["Sorry, can't understand you", "Please give me
more info", "Not sure I understand"],
         "context": [""]
        },
```

```
        {"tag": "options",

         "patterns": ["How could you help me?", "What you can do?",
"What help you provide?", "How you can be helpful?", "What support is
offered"],

         "responses": ["I will provide you with a safe way to go about
the FFCS in VIT"],

         "context": [""]

        },

        {"tag": "FFCS",

         "patterns": ["FFCS", "What is FFCS?"],

         "responses": ["FFCS is a Fully Flexible Credit System which
allows the student to choose the number of credits for a particular
semester, the tie slots for the subjects and the teachers "],

         "context": [""]

        },

        {"tag": "Courses",

         "patterns": ["Course wishlist", "wishlist"],

         "responses": ["Course wish list is where a student can select
its desired courses to pursue in the upcoming semester."],

         "context": [""]

        },

        {"tag": "Min and max",

         "patterns": ["Maximum Credits", "Minimum Credits"],

         "responses": ["The maximum number of credits a student can
choose for a particular semester 27 creds and the minimum is 19
creds"],

         "context": [""]

        },

        {"tag": "CGPA",

         "patterns": ["Is FFCS dependant on CGPA?", "CGPA dependent"],

         "responses": ["FFCS does not depend on the CGPA of a student.
It is mostly based on attendance. The first slot though will be given
to 9 pointers"],

         "context": [""]

        },
```

```
        {"tag": "slot time",

         "patterns": ["How many slots are there?","Are there slots for
ffcs?","How to register for ffcs?","How are slots allotted?"],

         "responses": ["There are 6 slots for ffcs registration. \n 1.
9-10 A.M - 9 Pointers. \n 2. 10-11 A.M - 100% Attendance. \n 3. 11-12
Noon - Day Scholars with 90% attendance and 8+ cgpa. \n 4. 12-1 P.M -
Rest day scholars and hostellers with 7+cgpa. \n 5. 1-2 P.M -
Hostellers with 6+ cgpa. \n 6. 2-3 P.M - Rest of them."],

         "context": [""]

        },


        {"tag": "Categories in ffcs",

         "patterns": ["What the categories for ffcs
registration?","What is university core?","What is program
core?","What is program elective?","What is university elective?"],

         "responses": ["There are 4 different categories. University
Core - University Core are courses which are common for all branches
and mostly will be done in first year and are mandatory for everyone.
\n Program Core - Programme Core are courses which are specific for
each branch and all are compulsory. \n University Elective -
University Electives are courses which you can choose from but are
not specific to branch and include management and humanities courses.
\n Program Elective - Programme Electives are courses which you can
choose from but are specific to each branch."],

         "context": [""]

        },


        {"tag": "Seats in each slot",

         "patterns": ["How many seats are available under one
teacher?","How many seats are allotted for each slot?","How many
students are there in one class?","How many students can register
under the same faculty?"],

         "responses": ["There are a total of 70 seats allotted for
each slot/teacher."],

         "context": [""]

        },


        {"tag": "Categories in ffcs",
```

```
        "patterns": ["What the categories for ffcs
registration?","What is university core?","What is program
core?","What is program elective?","What is university elective?"],

        "responses": ["There are 4 different categories. University
Core - University Core are courses which are common for all branches
and mostly will be done in first year and are mandatory for everyone.
\n Program Core - Programme Core are courses which are specific for
each branch and all are compulsory. \n University Elective -
University Electives are courses which you can choose from but are
not specific to branch and include management and humanities courses.
\n Program Elective - Programme Electives are courses which you can
choose from but are specific to each branch."],

        "context": [""]

       },


       {"tag": "LTPJC",

        "patterns": ["What is LTPJC?","What is the full form of
LTPJC?"],

        "responses": ["The LTPJC system is used for credit
distribution of courses in VIT University.\nL= Lectures given during
class by the professor.\nT= Tutorial, also class based with more
emphasis on problem solving\nP= Practical i.e. Lab Classes\nJ=
Project, introduced from 2015-16 academic year\nC is the total
credits of a course"],

        "context": [""]

       }


]

}
```


- words.pkl

]�(�'s��,��allotted��anyone��are�� available��awesome��be��bye��can��category��cgpa��chatting��class��c ore��could��course��credit��day�� dependant�� dependent��do��each��elective��faculty��ffcs��for��form��full��good�� goodbye��hello��help��helpful��helping��hey��hi��hola��how��in��is� ��it��later��ltpjc��many��maximum��me��minimum��name��next��nice ��of��offered��ok��on��one��program��provide��register��registration��

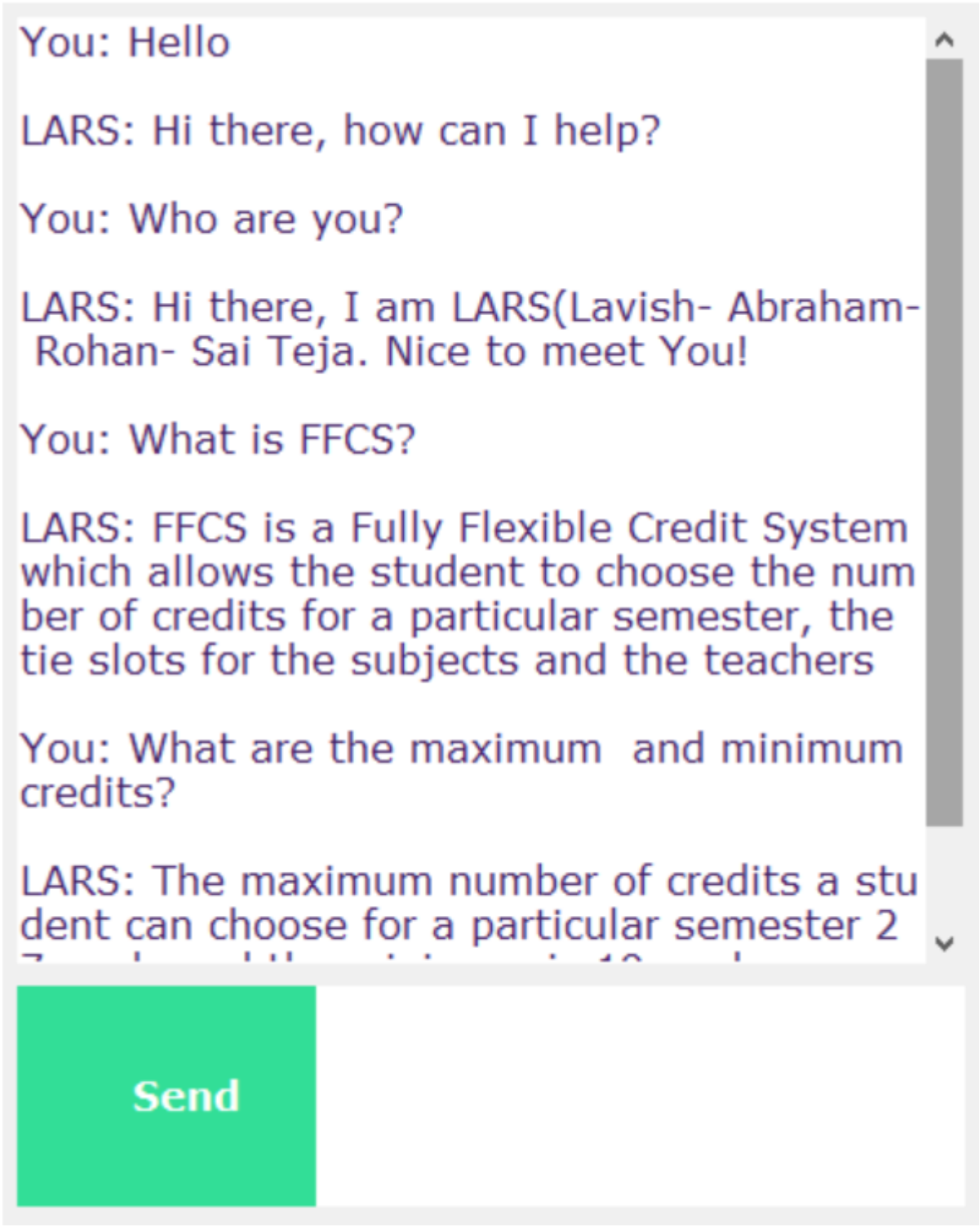
same��seat��see��slot��student��support��teacher��thank��thanks��that ��the��there��till��time��to��under��

university��wassup��what��who��wishlist��you��your�e.

- classes,pkl

  ���]�(�CGPA��Categories in ffcs��Courses��FFCS��LTPJC��Me�� Min and max��Seats in each slot��goodbye��greeting��noanswer��options�� slot time��thanks�e.

4. OUTPUT



RESULTS CONCLUSION AND INFERENCE

It can be concluded that the chatbot for FFCS registration could really help in making the registration much easier and hassle free for students. We believe it is a no brainer to not include this in all college course registrations. Chatbots have been an integral part of many applications and it ceases to amaze us how it hasn't been a part of course registrations.

Chatbots are able to provide high quality support to the students and get rid of any doubts in their mind.

Through the output results we have shown above it is evident that the chatbot is doing what is intended to do.

FUTURE WORK

With the rising popularity of chatbots it is very evident the scope for improvement in the future.

Some key improvements:

- Text-to-Speech and Speech-to-Text recognition.
- Additional Intents.
- Feedback Mechanism.
- Ability to handle questions that are unknown to the system.
- GUI Improvements.

REFERENCES

LIST OF PUBLICATIONS

- Winkler, R., & Söllner, M. (2018). Unleashing the potential of chatbots in education: A state-of-the-art analysis. In *Academy of Management Annual Meeting (AOM)*.

- Ali, S. M. (2019). Bots in libraries: they're coming for your jobs (or Is It?). Research Collection Library, 12(14), 1-21.

- Allison, D. (2012). Chatbots in the library: Is It Time? Library Hi Tech, 30(1), 95-107

- Liao, L., Zhou, Y., Ma, Y., Hong, R., & Tat-Seng, C. (2016). Knowledge-aware multimodal fashion chatbot. In 30th AAAI Conference on Artificial Intelligence, AAAI 2016, 3776-3783.

- Zumstein, D., & Hundertmark, S. (2017). CHATBOTS--AN INTERACTIVE TECHNOLOGY FOR PERSONALIZED COMMUNICATION, TRANSACTIONS AND SERVICES. *IADIS International Journal on WWW/Internet*, *15*(1).

- Shingte, Kshitija and Chaudhari, Anuja and Patil, Aditee and Chaudhari, Anushree and Desai, Sharmishta, Chatbot Development for Educational Institute (June 6, 2021).

BIODATA

Name : Abraham Kuriakose

Mobile Number : 7720976688

E-mail : abrahamkuriakosevit@gmail.com

Permanent Address: Pune, Maharashtra, India

Name : Lavish Jain

Mobile Number : 9025609996

E-mail : jiyalavish123@gmail.com

Permanent Address: Chennai, Tamil Nadu, India

Name :Sai Teja

Mobile Number : 7358455087

E-mail :

Permanent Address: Chennai, Tamil Nadu, India

Name : Rohan Jacob John

Mobile Number : 8089008430

E-mail : rohanjacobjohn@gmail.com

Permanent Address: Trivandrum, Kerala, India