TITLE OF PROJECT REPORT

**A major project report submitted for the partial fulfillment of the requirements for the Degree**

***Of***

**B.Tech**

By

**AMRUTESH MISHRA – 1601105006**
**PRATEEKSHYA PRIYADARSHINI – 1601105020**
**PRATIK PARIDA – 1601105021**
**PRIYANKA SINGH - 1601105025**

In

**Chatbot: Assisting tool for institutional queries**

Under the Guidance of

**Prof. SRINIVAS SETHI**
Dept of CSEA, IGIT Sarang



Department of Computer Science, Engineering & Applications

INDIRA GANDHI INSTITUTE OF TECHNOLOGY, SARANG
DHENKANAL, ODISHA-759146

# TABLE OF CONTENTS

# DECLARATION BY THE CANDIDATES

*We hereby declare that the synopsis entitled* **"Chatbot: An assisting tool for institution"** *under BPUT, ROURKELA, Odisha at* ***Indira Gandhi Institute Technology, Sarang, Dhenkanal, Odisha*** *is an authentic record of our work carried out under the supervision* ***Prof. Srinivas Sethi, Dept. of CSEA, IGIT, Sarang.*** *We have not submitted this major project report elsewhere for any other degree or diploma.*


AMRUTESH MISHRA

PRATEEKSHYA PRIYADARSHINI

PRATIK PARIDA

PRIYANKA SINGH

Department of CSEA,

IGIT,Sarang

**INDIRA GANDHI INSTITUTE OF TECHNOLOGY**
**SARANG**

# ***Certificate***

*This is to certify that this project entitled ""**Chatbot: An assisting tool for institution**" submitted by **Amrutesh Mishra (1601105006), Prateekshya Priyadarshini (1601105020), Pratik Parida (1601105021) and Priyanka Singh (1601105025)** of Computer Science Engineering & Applications Department, Indira Gandhi Institute of Technology, Sarang for the partial fulfillment of the requirements for the award of Bachelor of Technology (Computer Science & Engineering) Degree of BPUT, Odisha, is a record of students own study carried under our supervision & guidance.*

*This report has not been submitted to any other university or institution for the award of any degree.*

*Date:*

***Dr. Srinivas Sethi,***
***Associate Professor,***
***Dept. of CSEA, IGIT, Sarang***
***Dhenkanal, Odisha***

***(Dr. S. N. Mishra)***
***Professor & Head,***
***Dept. of CSEA, IGIT, Sarang***
***Dhenkanal, Odisha***

***Signature of***
***External Examiner***

# ACKNOWLEDGEMENTS

It is a great pleasure and privilege to express our profound sense of gratitude to prof. SRINIVAS SETHI, Dept. of CSEA, IGIT, Sarang, our guide for his suggestions, motivation and support during the major project preparation and keen personal interest throughout the progress of our course work.

We would like to express our sincere thanks to prof. (Dr.) S. N. Mishra, HOD, Dept. of CSEA, IGIT, Sarang, for his suggestion, motivation and support during the major project preparation.

We have been fortunate enough to have all support motivation from my parents.

We sincerely thank to the other faculties of the department of CSEA, IGIT, Sarang for their positive and constructive suggestion and kind cooperation during major project preparation and presentation.

.

**AMRUTESH MISHRA**
**PRATEEKSHYA PRIYADARSHINI**
**PRATIK PARIDA**
**PRIYANKA SINGH**

# Abstract

A chatbot is an artificial intelligence program that simulates interactive human conversation by using key pre-calculated user phrases and auditory or text-based signals. Chatbots, also known as conversational interfaces present a new way for individuals to interact with computer systems. A chatbot allows a user to simply ask questions and get a desirable answer, in the same manner that they would address a human. The technology used for their development is NLP (Natural Language Processing). Recent advances in ML (Machine Learning) have greatly improved the accuracy and effectiveness of chatbots, making them a viable option for many organizations. The 24x7 availability and immense knowledge held by the chatbots are the main reason for their bright future.

We are planning to create a user friendly chatbot to help people with any institution related queries. The System uses built in artificial intelligence to answer the queries put by the user. The user can query any college related activities through the system and need not personally go to the college for enquiry. Our system will also allow the users to add new answers and notify about the incorrect answers. The new information will be updated after being verified by the admin.

**Group Members:**

| | |
|---|---|
| Amrutesh Mishra | 1601105006 |
| Prateekshya Priyadarshini | 1601105020 |
| Pratik Parida | 1601105021 |
| Priyanka Singh | 1601105025 |

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **AIML** | Artificial Intelligence Markup Language |
| **HTML** | Hyper Text Markup Language |
| **DFD** | Data Flow Diagram |
| **UML** | Unified Modelling Language |
| **SQL** | Structured Query Language |

# 1 INTRODUCTION

## 1.1 CHATBOT AND MACHINE LEARNING

Machine learning chatbots works using artificial intelligence. User need not to be more specific while talking with a bot because it can understand the natural language, not only commands. This kind of bots get continuously better or smarter as it learns from past conversations it had with people. Here is a simple example which illustrate how they work. The following is a conversation between a human and a chatbot:

Human: “I need a flight from San Jose to New York.” Bot: “Sure! When would you like to travel?” Human: “From Dec 20, 2016 to Jan 28, 2017.” Bot: “Great! Looking for flights.”

In order to achieve the ultimate goal, we have taken an iterative approach and divided my work into four major deliverables. These deliverables not only helped us in understanding the code structure of our bot but also enhances the bot’s functionality. In the rest of the report, we will be discussing about the four deliverables. To understand more on chatbot service, we had implemented a Facebook Messenger Weather Bot in deliverable 1, which is discussed in next section. The purpose of deliverable 2 is to introduce chatbots to the college website. We have added Bot Configuration settings which is used to add bot users. In the next deliverable, we have added a functionality where the user will be able to call bots in a group thread. Activation of bots will happen by calling respective callback URL which is already configured that helps bots to have a conversation with users. More details on this is discussed in deliverable 3 section. We have added some sample chats with our bot in this document.

## 1.2 ARTIFICIAL INTELLIGENCE

AI was coined by John McCarthy, an American computer scientist, in 1956 at The Dartmouth Conference where the discipline was born. Today, it is an umbrella term that encompasses everything from robotic process automation to actual robotics. It has gained prominence recently due, in part, to big data, or the increase in speed, size and variety of data businesses are now collecting. AI can perform tasks such as identifying patterns in the data more efficiently than humans, enabling businesses to gain more insight out of their data.

### 1.2.1 Types of Artificial Intelligence

AI can be categorized in any number of ways, but here are two examples.

The first classifies AI systems as either **weak AI** or **strong AI**. **Weak AI**, also known as narrow AI, is an AI system that is designed and trained for a particular task. Virtual personal assistants, such as Apple's Siri, are a form of weak AI. **Strong AI**, also known as artificial general intelligence, is an AI system with generalized human cognitive abilities so that when presented with an unfamiliar task, it has enough intelligence to find a solution. The Turing Test, developed by mathematician Alan Turing in 1950, is a method used to determine if a computer can actually think like a human, although the method is controversial. The second example is from Arend Hintze, an assistant professor of integrative biology and computer science and engineering at Michigan State University. He categorizes AI into four types, from the kind of AI systems that exist today to sentient systems, which do not yet exist. His categories are as follows:

**Type 1: Reactive Machines**

An examples is Deep Blue, the IBM class program that beat Garry Kasparov in the 1990s. Deep Blue can identify pieces on the class board and make predictions, but it has no memory and cannot use past experiences to inform future ones. It analyzes possible moves – its own and its opponent – and chooses the most strategic move. Deep Blue and Google's AlphaGO were designed for narrow purposes and cannot easily be applied to another situation.

**Type 2: Limited Memory**

These AI systems can use past experiences to inform future decisions. Some of the decision-making functions in autonomous vehicles have been designed this way. Observations used to inform actions happening in the not-so-distant future, such as a car that has changed lanes. These observations are not stored permanently.

**Type 3: Theory of mind**

This is a psychology term. It refers to the understanding that others have their own beliefs, desires and intentions that impact the decisions they make. This kind of AI does not yet exist.

**Type 4: Self-awareness**

In this category, AI systems have a sense of self, have consciousness. Machines with self-awareness understand their current state and can use the information to infer what others are feeling. This type of AI does not yet exist.

### 1.2.2 Examples of AI technology

Automation is the process of making a system or process function automatically. Robotic process automation, for example, can be programmed to perform high-volume, repeatable performeance by humans. RPA is different from IT automation in that it can adapt to changin gcircumstances.Machine learning is the science of getting a computer to act without programming. Deep learning is a subset of machine learning that, in very simple terms, can be thought of as the automation of predictive analytics. There are three types of machine learning algorithms: supervised learning, in which data sets are labeled so that patterns can be detected and used to label new data sets; unsupervised learning, in which data sets aren't labeled and are sorted according to similarities or differences; and reinforcement learning, in which data sets aren't labeled but, after performing an action or several actions, the AI system is given feedback. Machine vision is the science of making computers see. Machine vision captures and analyzes visual information using a camera, analog-to-digital conversion and digital signal processing. It is often compared to human eyesight, but machine vision isn't bound by biology and can be programmed to see through walls, for example. It is used in a range of applications from signature identification to medical image analysis. Computer vision, which is focused on machine-based image processing, is often conflated with machine vision. Natural language processing (NLP) is the processing of human -- and not computer -- language by a computer program. One of the older and best known examples of NLP is spam detection, which looks at the subject line and the text of an email and decides if it's junk. Current approaches to NLP are based on machine learning. NLP tasks include text translation, sentiment analysis and speech recognition. Pattern recognition is a branch of machine learning that focuses on identifying patterns in data. The term, today, is dated. Robotics is a field of engineering focused on the design and manufacturing of robots. Robots are often used to perform tasks that are difficult for humans to perform or perform consistently. They are used in assembly lines for car production or by NASA to move large objects in space. More recently, researchers are using machine learning to build robots that can interact in social settings.

## 1.3 AI APPLICATIONS

### 1.3.1 AI in healthcare

The biggest bets are on improving patient outcomes and reducing costs. Companies are applying machine learning to make better and faster diagnoses than humans. One of the best known healthcare technologies is IBM Watson. It understands natural language and is capable of responding to questions asked of it. The system mines patient data and other available data sources to form a hypothesis, which it then presents with a confidence scoring schema. Other AI applications include chatbots, a computer program used online to answer questions and assist customers, to help schedule follow-up appointments or aiding patients through the billing process, and virtual health assistants that provide basic medical feedback

### 1.3.2 AI in business

Robotics process automation is being applied to highly repetitive tasks normally performed by humans. Machine learning algorithms are being integrated into analytics and CRM platforms to uncover information on how to better serve customers. Chatbots have been incorporated into websites to provide immediate service to customers. Automation of job positions has also become a talking point among academics and IT consultancies such as Gartner and Forrester.

### 1.3.3 AI in education

AI can automate grading, giving educators more time. AI can assess students and adapt to their needs, helping them work at their own pace. AI tutors can provide additional support to students, ensuring they stay on track. AI could change where and how students learn, perhaps even replacing some teachers.

### 1.3.4 AI in finance

AI applied to personal finance applications, such as Mint or Turbo Tax, is upending financial institutions. Applications such as these could collect personal data and provide financial advice. Other programs, IBM Watson being one, have been applie d to the process of buying a home. Today, software performs much of the trading on Wall Street.AI in law. The discovery process, sifting through of documents, in law is often overwhelming for humans. Automating this process is a better use of time and a more efficient process. Startups are also building question-and-answer computer assistants that can

sift programmed-to-answer questions by examining the taxonomy and ontology associated with a database.AI in manufacturing. This is an area that has been at the forefront of incorporating robots into the workflow. Industrial robots used to perform single tasks and were separated from human workers, but as the technology advanced that changed.

# 2 BACKGROUND AND LITERATURE REVIEW

## 2.1 NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is the study of letting computers understand human languages [3]. Without NLP, human language sentences are just a series of meaningless symbols to computers. Computers don't recognize the words and don't understand the grammars. NLP can be regard as a "translator", who will translate human languages to computer understandable information. Traditionally, users need to follow well-defined procedures accurately, in order to interact with computers. For example, in Linux systems, all commands must be precise. A single replace of one character or even a space can have significant difference. However, the emergence of NLP is changing the way of interacting. Apple Siri and Microsoft Cortana have made it possible to give command in everyday languages and is changing the way of interacting.

## 2.2 MACHINE LEARNING

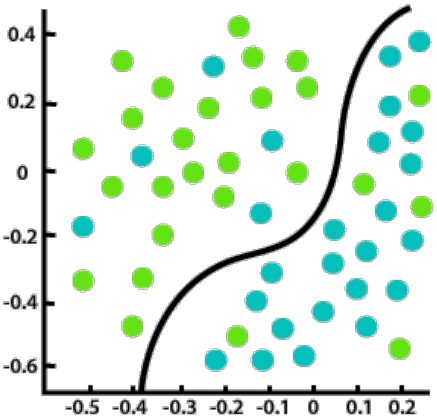
Machine Learning (ML) is an area of computer science that "gives computers the ability to learn without being explicitly programmed". The parameter of the formulas is calculated from the data, rather than defined by the programmer. Two most common usage of ML is Classification and Regression. As shown in figure1 [8], Classification means to categorize different types of data, while Regression means to find a way to describe the data. Basic ML program will have two stages, fitting and predicting. In the fitting stage, the program will be given a large set (at least thousands) of data. The program will try to adjust its parameter based on some statistical models, in order to make it "fit" the input data best. In the predicting stage, the program will give a prediction for a new input based on the parameters it just calculated out. For example, the famous Iris flower dataset [9] contains the measurement of several features of three different species of flowers, such as the length of

sepals and petals. A well-defined ML program can learn the pattern behind this feature and give prediction accordingly.

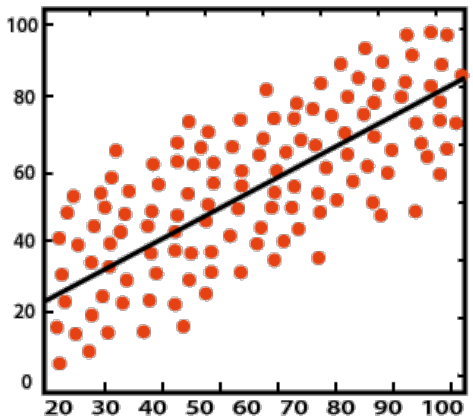
## 2.3 LITERATURE REVIEW

| Purpose | Papers | Year | Pros | Cons |
|---|---|---|---|---|
| Usefulness of chatbots | Chatbot for University related FAQs | 2017 | Significance of chatbot | Machine like response |
| Natural Language Processing | An overview of AI chatbots and example chatbot application | 2018 | Benefit and detailed overview of natural language processing in any AIML chatbot | Technical and obsolete problems in NLP mechanism |
| Real time solution for open learner model | Programming challenges of chatbot: current and future scenario | 2017 | Use of technology to enhance response | Limited information and problem in integration |
| Intelligent tutorial system | The role of chatbot in formal education | 2018 | Detailed description of python and AIML | Only for chatbot based AIML |
| AIML and brain loading | AI Chatbot With AIML submitted by Nano Dano | 2015 | Effective way of installation | Slow brain loading |

(TABLE2.1)




(Figure 2.1a: Classification)




(Figure 2.1b: Regression)

# 3 PROBLEM IDENTIFICATION

## 3.1 PROBLEM IDENTIFICATION

Changing of requirements: as an industrial project to build a product, we must follow the requirement from the user. However, because the project's goal is to be used by the users, but it is responsible by the technical team, the requirement changed a lot in the middle after a meeting with the business team. The users want a simple bot which can give basic information about IGIT.

### 3.1.1 Lack of training data

The quantity and quality of training data is critical to the performance to a machine learning model. However, because some confidential and privacy reasons, we couldn't get enough data for us and we had to make up data by our own. For the machine learning model, we generate some fake data based on our daily life experience, which is really biased, although with a good accuracy on the fake data.

### 3.1.2 Unstable API version

Because, API service we are using, are still under development and we cannot fix to a version of the API, the API may change over time. Moreover, there are inconsistencies between the APIs and their documents or sample codes.

### 3.1.3 Not familiar with the Python language and AIML

None of the four of us has previous knowledge with python. Programming in a new language in such a huge collection of scripting files is quite challenging for us at the beginning of the project. However, when we came to the later phase, we were more used to that.

# 4 REQUIREMENTS AND SPECIFICATION

## 4.1 HARDWARE REQUIREMENTS

Processor: Pentium IV (minimum)

Hard disk: 40 GB (minimum)

RAM: 256 MB (minimum)

## 4.2 SOFTWARE REQUIREMENTS

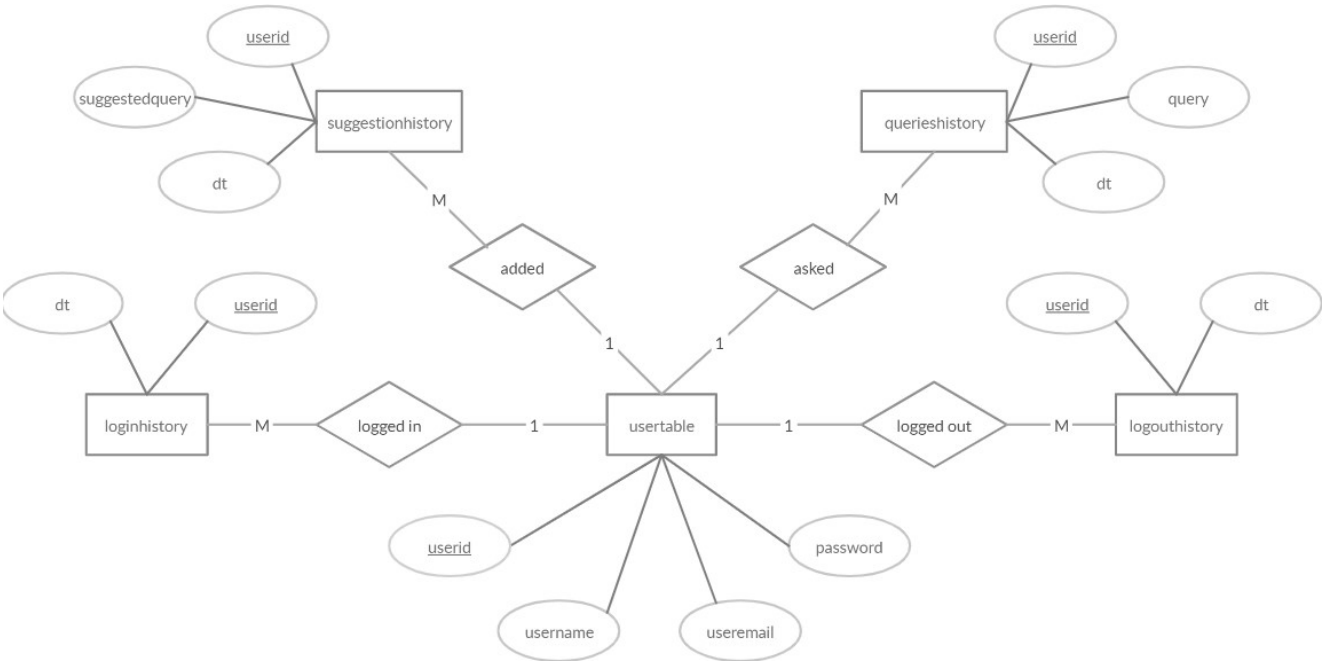
Operating system: Windows or Linux

Technology: Python, AIML

# 5 DATABASES AND UML DIAGRAM

## 5.1 ER DIAGRAM

The database is Oracle 10.2.0 and is stored in a local database. Every table within the database is represented using a model which allows logic and relationships between database objects to be defined and manipulated.




(Fig 5.1)

USERTABLE contains all the details of the users. Any person trying to use the bot has to register first. Then the user has to login. LOGINHISTORY and LOGOUTHISTORY are two tables which contain all the records of all users who are logging in and out with date and time. This is done for security purpose. The queries put by all the users are also tracked and stored in QUERIESHISTORY. When a user tries to add new questions to the bot, then those queries are stored in SUGGESTIONHISTORY table.

## 5.2 SCHEMA DIAGRAM

USERTABLE

| USERID | USERNAME | USEREMAIL | PASSWORD |
|---|---|---|---|

LOGINHISTORY

| USERID | DT |
|---|---|

LOGOUTHISTORY

| USERID | DT |
|---|---|

QUERIESHISTORY

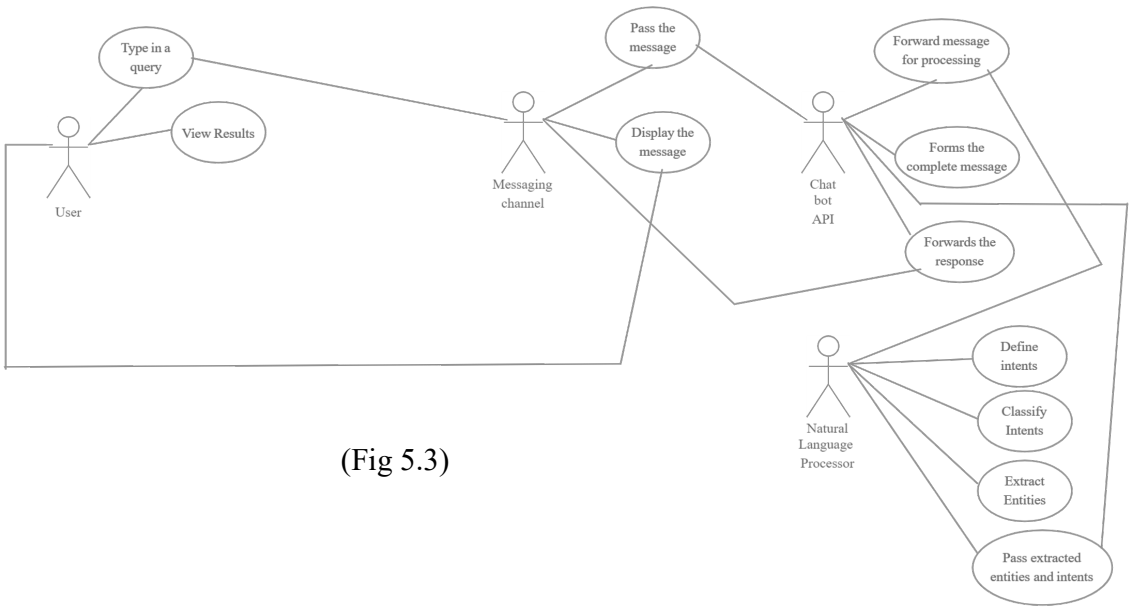
| USERID | QUERY | DT |
|---|---|---|

SUGGESTIONHISTORY

| USERID | SUGGESTEDQUERY | DT |
|---|---|---|

(FIG 5.2)

## 5.3 USE CASE DIAGRAM




(Fig 5.3)

## 5.4 TABLES

### 5.4.1 USERTABLE

| NAME | NULL? | TYPE |
|---|---|---|
| USERNAME | | VARCHAR2(50) |
| USERID | NOT NULL | VARCHAR2(25) |
| PASSWORD | | VARCHAR2(50) |
| USEREMAIL | | VARCHAR2(75) |

### 5.4.2 LOGINHISTORY

| NAME | NULL? | TYPE |
|---|---|---|
| USERID | | VARCHAR2(25) |
| DT | | DATE |

### 5.4.3 LOGOUTHISTORY

| NAME | NULL? | TYPE |
|---|---|---|
| USERID | | VARCHAR2(25) |
| DT | | DATE |

### 5.4.4 QUERIESHISTORY

| NAME | NULL? | TYPE |
|---|---|---|
| USERID | | VARCHAR2(25) |
| QUERY | | VARCHAR2(250) |
| DT | | DATE |

### 5.4.5 SUGGESTIONHISTORY

| NAME | NULL? | TYPE |
|---|---|---|
| USERID | | VARCHAR2(25) |
| SUGGESTEDQUERY | | VARCHAR2(250) |
| DT | | DATE |

## 5.5 PROCEDURES AND FUNCTIONS

### 5.5.1 PROCEDURES

Procedure to save login data:

```
create or replace procedure savelogindetails (uid in varchar2) as
      dt varchar2(25);
      begin
      select to_char(sysdate,'DD-MON-YYYY HH:MI:SS') into dt from dual;
      insert into loginhistory values (uid, TO_DATE(dt, 'DD-MON-YYYY
HH:MI:SS'));
      end;
/
```

Procedure to save logout data:

```
create or replace procedure savelogoutdetails (uid in varchar2) as
      dt varchar2(25);
      begin
      select to_char(sysdate,'DD-MON-YYYY HH:MI:SS') into dt from dual;
      insert into logouthistory values (uid, TO_DATE(dt, 'DD-MON-YYYY
HH:MI:SS'));
      end;
/
```

Procedure to save queries:

```
create or replace procedure savequerydetails (uid in varchar2, qry in
varchar2) as
      dt varchar2(25);
      begin
      select to_char(sysdate,'DD-MON-YYYY HH:MI:SS') into dt from dual;
      insert into querieshistory values (uid, qry, TO_DATE(dt, 'DD-MON-
YYYY HH:MI:SS'));
      end;
/
```

Procedures to save suggestions:

```
create or replace procedure savesuggestiondetails (uid in varchar2, qry in
varchar2) as
      dt varchar2(25);
      begin
      select to_char(sysdate,'DD-MON-YYYY HH:MI:SS') into dt from dual;
```

```
      insert into suggestionhistory values (uid, qry, TO_DATE(dt, 'DD-MON-
YYYY HH:MI:SS'));
      end;
/
```

### 5.5.2 FUNCTIONS

Function to match passwords:

```
create or replace function matchpassword (uid in varchar2, pwd in
varchar2) return varchar2 as result varchar2(25);
      pw usertable.password%type;
      begin
      result:='false';
      select password into pw from usertable where userid=uid;
      if pw=pwd then
            result:='true';
      end if;
      return result;
end;
/
```

Function to find the username given userid:

```
create or replace function findusername (uid in varchar2) return varchar2
as uname varchar2(50);
      begin
      select username into uname from usertable where userid=uid;
      return uname;
end;
/
```

# 6 PROPOSED SOLUTION

## 6.1 AIML SCRIPTING

We created the AIML file that only handles one pattern, load aiml bot. When we enter that command to the bot, it will try to load all the AIML files. It won't work unless we actually create it. Here is what you can put inside. We will match two basic patterns and respond. Sample AIML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0.1">

    <category>
       <pattern>HEY</pattern>
       <template>
          <srai>HELLO</srai>
       </template>
    </category>

    <category>
        <pattern>HELLO</pattern>
        <template>Hey! How may I help you?</template>
    </category>

</aiml>
```

(Fig 6.1)

## 6.2 CREATING A STARTUP FILE

It is standard to create a startup file called std-startup.xml as the main entry point for loading the AIML files. In this case we will create a basic file that matches one pattern and takes on action. We want to match the pattern **load aiml bot**, and it laods the aiml brain in response. We will create the aiml files in a minute.

```
<aiml version="1.0.1">
    <category>
        <pattern>LOAD AIML BOT</pattern>
        <template>
            <learn>affiliations.aiml</learn>
            <learn>academics.aiml</learn>
            <learn>studentslife.aiml</learn>
            <learn>funnyQuestions.aiml</learn>
            <learn>instituteProfile.aiml</learn>
            <learn>bogMembers.aiml</learn>
            <learn>administration.aiml</learn>
            <learn>faculties.aiml</learn>
            <learn>general.aiml</learn>
        </template>
    </category>
</aiml>
```

(Fig – 6.2a)

```
Enter your password - abcd
Loading startup.xml...done (1.77 seconds)
Loading affiliations.aiml...done (0.00 seconds)
Loading academics.aiml...done (0.01 seconds)
Loading studentslife.aiml...done (0.02 seconds)
Loading funnyQuestions.aiml...done (0.03 seconds)
Loading instituteProfile.aiml...done (0.01 seconds)
Loading bogMembers.aiml...done (0.01 seconds)
Loading administration.aiml...done (0.04 seconds)
Loading faculties.aiml...done (0.04 seconds)
Loading general.aiml...done (0.05 seconds)

Kernel bootstrap completed in 2.09 seconds
Saving brain to botBrain.brn...done (0.32 seconds)
prateekshya119 >>
```

(Fig – 6.2b)

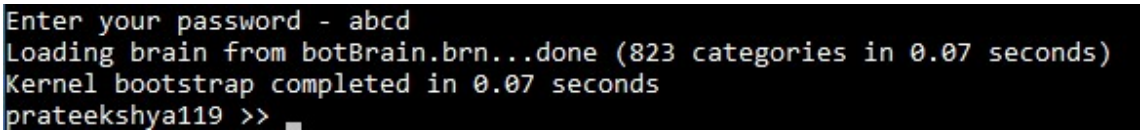
## 6.3 SPEEDING UP BRAIN LOAD

When you start to have a lot of AIML files, it can take a long time to learn. This is where

brain files come in. after the bot learns all the AIML files it can save its brain directly to a file which will drastically speed up load times on subsequent runs.

## 6.4 LOADING BRAIN

This is the simplest program we can start with. It creates the AIML object, learns the startup file, and then loads the rest of the AIML files. After that, it is ready to chat, and we enter an infinite loop that will continue to prompt the user for a message. You will need to enter a pattern that the bot recognizes. The patterns recognized depend on what AIML files you have loaded. We create the startup file as a separate entity so that we can add more AIML files to the bot later without having to modify any of the programs' source code. We can just add more and more files to learn in the .xml file.




(Fig – 6.3)

# 7 IMPLEMENTATION

## 7.1IMPLEMENTATION

This section covers the design and implementation of different modules of the bot, which contain the design of the PYTHON module, the Translator API and the AIML module.

### 7.1.1 Python code

```
import aiml
import os
import cx_oracle
import re
import random
try:
    dbtns=cx_oracle.makedsn('prateekshya','1521',service_name='xe')
    connection =
cx_oracle.connect(user='system',password='soni1999',dsn=dbtns)
    cur = connection.cursor()
except cx_oracle.databaseerror as e:
    print("error in database: ",e)
status = input("1- register\n2- login\nenter a number - ")
while (status == "1" or status == "2"):
    if status == "1":
        try:
            un = input("enter your name - ")
            umail = input ("enter you email id - ")
            upw = input("enter a password - ")
            i = len(un)
            flag = 0
            uid = un[0].lower()
            j = 1
```

```
            while (j != i):
                if un[j].isspace():
                    flag = flag + 1
                    uid = uid + un[j+1].lower()
                elif flag == 0:
                    uid = uid + un[j].lower()
                j = j + 1
            arr = random.sample(range(1,999),10)
            index = random.sample(range(0,9),1)[0]
            randomint = arr[index]
            uid = uid + str(randomint)
            querystring = "insert into usertable values
('"+un+"','"+uid+"','"+upw+"','"+umail+"')"
            cur.execute(querystring)
            connection.commit()
            print("registered successfully! your userid is- ",uid)
        except cx_oracle.databaseerror as e:
            print("error in database: ", e)
        status = "2"
    elif status == "2":
        uid = input("enter your userid - ")
        password = input("enter your password - ")
        try:
            querystring = "select password from usertable where userid =
('"+uid+"')"
            cur = connection.cursor();
            cur.execute(querystring)
            result = cur.var(cx_oracle.string)
            cur.callfunc('matchpassword',result,[uid,password])
            if result.getvalue()=="true":
                cur = connection.cursor()
                cur.callproc('savelogindetails',[uid])
                connection.commit()
                kernel = aiml.kernel()
                if os.path.isfile("botbrain.brn"):
                    kernel.bootstrap(brainfile = "botbrain.brn")
                else:
                    kernel.bootstrap(learnfiles = "startup.xml", commands
= "load aiml bot")
                    kernel.savebrain("botbrain.brn")
                while true:
                    message = input(uid+" >> ")
                    cur = connection.cursor()
                    cur.callproc('savequerydetails',[uid,message])
                    connection.commit()
                    if message == "save":
                        kernel.savebrain("botbrain.brn")
                    elif message == "learn":
                        query = input (uid+" >> ")
                        while (query != "end"):
                            try:
                                cur = connection.cursor()

cur.callproc('savesuggestiondetails',[uid,query])
                                connection.commit();
                            except cs_oracle.databaseerror as e:
                                print("error in database: ", e)
                            query = input (uid+" >> ")
                        print("bot >> thanks for teaching me! go ahead.");
                    else:
                        botresponse = kernel.respond(message)
                        if botresponse == "hey! how can i help you?" or
botresponse == "hope to see you again! have a good day! bye!" or
botresponse == "okay, have a happy life ahead!":
                            cur = connection.cursor()
                            name = cur.var(cx_oracle.string)
                            cur.callfunc('findusername',name,[uid])
```

```
                                printname = name.getvalue()[0]
                                i = len(name.getvalue())
                                flag = 0
                                j = 1
                                while (j != i):
                                    if name.getvalue()[j].isspace():
                                        flag = flag + 1
                                    elif flag == 0:
                                        printname = printname +
name.getvalue()[j]
                                    j = j + 1
                                if botresponse == "hey! how can i help you?":
                                    print ("bot >> hey, "+printname+"! how can
i help you?")
                                elif botresponse == "hope to see you again!
have a good day! bye!":
                                    print ("bot >> hope to see you again! have
a good day, "+printname+"! bye!")
                                    cur = connection.cursor()
                                    cur.callproc('savelogoutdetails',[uid])
                                    connection.commit()
                                    exit()
                                elif botresponse == "okay, have a happy life
ahead!":
                                    print ("bot >> okay, "+printname+"! have a
happy life ahead!")
                                    cur = connection.cursor()
                                    cur.callproc('savelogoutdetails',[uid])
                                    connection.commit()
                                    exit()
                            else:
                                print ("bot >> "+botresponse)
                    else:
                        print("wrong userid or password!")
                except cx_oracle.databaseerror as e:
                    print("error in database: ",e)
                status = "-1"
                cur.close()
                connection.close()
```

# 8 ADVANTAGES AND DISADVANTAGES

## 8.1 ADVANTAGES

### 8.1.1 Accessible anytime

We are sure most of you are always kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as emails and phone calls. Since chatbots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break. This improves your customer UX and helps you rank highly in your

sector. Another advantage of this instant response is that you can also skillfully craft your chatbot to maintain your image and brand.

### 8.1.2 Handling capacity

Unlike humans who can only communicate with one human at a time, chat bots can simultaneously have conversations with thousands of people. No matter what time of the day it is or how many people are contacting you, every single one of them will be answered immediately. Imagine you own a restaurant, and you have a good reputation for your food of which most of your revenues come from delivery. As the demand keeps rising, you will have more customers to take orders from but very few staff to attend them all. Having a chatbot would eliminate such problem and cater to each and every person and ensure that no order is missed. Companies like Taco Bell and Dominos are already using chatbots to arrange delivery of parcels.

### 8.1.3 Flexible attribute

Chatbots have the benefit that it can quite easily be used in any industry. Unlike other products where you have to do a lot of development and testing to change platforms, chatbots are relatively easy to switch. One has to just train the bot by giving the right conversation structure and flow to switch its current field or industry. Or if there is a lot of back and forth between two sections of the industry say customer support and sales, then you could have custom built presets which would already have the conversation flow and structure to carry out the interactions with the user.

### 8.1.4 Customer satisfaction

Humans are bound to change of emotions. Chatbots, on the other hand, are bound by some rules and obey them as long as they're programmed to. They will always treat a customer in the perfect way no matter how rough the person is or how foul language the person uses. Not everyone orders the same food every day, people choices may change every day. In this case, it can use your order history to make suggestions for the next order, learn your address details and much more. Customers love this smooth interaction and want all their transactions to be as simple as possible.

### 8.1.5 Cost effective

Hiring a human for a job is never a cheap affair, and it will be expensive if your revenue are not high or sales targets are not met and would create havoc in the business. Due to the boundaries of human beings, a single human can only handle one or two people at the same time. More than that would be extremely tough for the employee. Chatbots could help solve this age-old problem. As one chatbot is equal to loads of employees, it can easily communicate with thousands of customers at the same time. We would only need a handful of people to jump into conversations sometimes when necessary. Hence, it would drastically bring down the expenses and bring about a steep rise in revenue and customer satisfaction.

### 8.1.6 Faster onboarding

Before you want to accomplish a task, you first must learn how to work on the task and complete it. Only then will they be considered fit for the job. There is a continuous teaching involved in every level of hierarchy the employee will go through. Also, there will be a lot of change in the employees, some stay, some get fired, some more join in etc. What we want to say is, employees will change; it's a fact. And this would require you to allot a lot time of your employees into grooming the new joiners. Chatbots could eliminate that time to almost zero, but provide a very clean and easy to understand conversation flow and structure that needs to be maintained by the chatbot. No doubt there will be changes in this too, but it will rather take a fraction of your time to resolve as compared to human employees.

## 8.2 DISADVANTAGES

### 8.2.1 Too many functions

Most of developers strive to create a universal chatbot that will become a fully-fledged assistant to user. But in practice functional bots turn out not to cope with the majority of queries. They often do not understand the user, forget what they were told 5 minutes earlier, and have many other disadvantages. And that is no wonder, as the development of a universal bot, which would understand natural language and could evaluate context, takes years of hard work for a team of experienced programmers. And even in this case, such programs should be constantly improved while in service. However, modern technologies allow building rather useful bots to perform specific actions such as booking train tickets or providing support to bank customers.

### 8.2.2 Primitive Algorithms

There are two types of bots: based on artificial intelligence, being able to learn in the process of communication; programmed for specific behavior scenarios. Artificial intelligence chatbots are considered to be better, as they can respond depending on the situation and context. But the development of complex algorithms is required for this purpose. Meanwhile, only IT giants and few developers possess such powerful technological base. Therefore, it would be better for ordinary companies to focus on the second variant of bots, as they are more reliable and simpler. Namely for the reason they do not possess intelligence, they will not be able to adopt rude communication patterns and get beyond the control of creators.

### 8.2.3 Complex interface

Talking to a bot implies talking in a chat, meaning that a user will have to write a lot. And in case a bot cannot understand the user's request, he will have to write even more. It takes time to find out which commands a bot can respond to correctly, and which questions are better to avoid. Thus, talking to a chatbot does not save time in the majority of cases. Perhaps the efficiency of virtual assistants will increase due to the implementation of voice recognition function in the future. But for the time being their functional capabilities are very restricted, and they can be truly useful only in a few business areas.

# 9 TECHNOLOGY

## 9.1 ABOUT PYTHON

Dating from 1991, Python is a relatively new programming language. From the start, Python was considered a gap-filler, a way to write scripts that "automate the boring stuff" (as one popular book on learning Python put it) or to rapidly prototype applications that will be implemented in one or more other languages. However, over the past few years, Python has emerged as a first-class citizen in modern software development, infrastructure management, and data analysis. It is no longer a back-room utility language, but a major force in web application development and systems management and a key driver behind the explosion in big data analytics and machine perfect for IT, Python simplifies many kinds of work, from system automation to working in cutting-edge fields like machine learning. Python is easy to learn. The number of features in the language itself is modest, requiring relatively little

investment of time or effort to produce one's first programs. Python syntax is designed to be readable and straightforward. This simplicity makes Python an ideal teaching language, and allows newcomers to pick it up quickly. Developers spend more time thinking about the problem they're trying to solve, and less time thinking about language complexities or deciphering code left by others. Python is broadly used and supported. Python is both popular and widely used, as the high rankings in surveys like the Tiobe Index and the large number of GitHub projects using Python attest. Python runs on every major operating system and platform, and most minor ones too. Many major libraries and API-powered services have Python bindings or wrappers, allowing Python to interface freely with those services or make direct use of those libraries. Python may not be the fastest language, but what it lacks in speed, it makes up for in versatility. Python is not a "toy" language. Even though scripting and automation cover a large chunk of Python's use cases (more on that below), Python is also used to build robust, professional quality software, both as standalone applications and as web services.

### 9.1.1 What is Python used for

The most basic use case for Python is as a scripting and automation language. Python isn't just a replacement for shell scripts or batch files, but is also used to automate interactions with web browsers or application GUIs or system provisioning and configuration in tools such as Ansible and Salt. But scripting and automation represent only the tip of the iceberg with Python.

- Python is used for general application programming. Both CLI and cross-platform GUI applications can be created with Python and deployed as self-contained executable. Python doesn't have the native ability to generate a standalone binary from a script, but third-party packages like cx_Freeze or PyInstaller can be used to accomplish that.
- Python is used for data science and machine learning. Sophisticated data analysis has become one of fastest moving areas of IT and one of Python's star use cases. The vast majority of the libraries used for data science or machine learning have Python interfaces, making the language the most popular high-level command interface to for machine learning libraries and other numerical algorithms.
- Python is used for web services and RESTful APIs. Python's native libraries and third-party web frameworks provide fast and convenient ways to create everything

from simple REST APIs in a few lines of code, to full-blown, data-driven sites. Python's latest versions have powerful support for asynchronous operations, allowing sites to handle up to tens of thousands of requests per second with the right libraries.

- Python is used for Meta programming. In Python, everything in the language is an object, including Python modules and libraries themselves. This allows Python to work as a highly efficient code generator, making it possible to write applications that manipulate their own functions and have the kind of extensibility that would be difficult or impossible to pull off in other languages.
- Python is used for glue code. Python is often described as a "glue language," meaning it can allow disparate code (typically libraries with C language interfaces) to interoperate. Its use in data science and machine learning is in this vein, but that's just one incarnation of the general idea. Also worth noting are the sorts of tasks Python is not well-suited for. Python is a high-level language, so it's not suitable for system level programming device drivers or OS kernels are straight out. It's also not ideal for situations that call for cross standalone binaries. You could build a standalone Python app for Windows, Mac, and Linux but not elegantly or simply. Finally, Python is not the best choice when speed is an absolute priority in every aspect of the application. For that you're better off with C/C++ or another language of that caliber.

### 9.1.2 Pros and Cons of Python

Python syntax is meant to be readable and clean, with little pretense. A standard "hello world" in Python 3.x is nothing more than:

- print("Hello world!")
- Python provides many syntactical elements that make it possible to concisely express many common program flows. Consider a sample program for reading lines from a text file into a list object, stripping each line of its terminating newline character along the way:
- with open('myfile.txt') as my_file:
- file_lines = [x.strip('\n') for x in my_file]
- The with/as construction is a "context manager," which provides an efficient way to instantiate a given object for a block of code and then dispose of it outside of

that block. In this case, the object in question is my_file, instantiated with the open() function. This takes the place of several lines of boilerplate to open the file, read individual lines from it, and then close it up.

- The [x … for x in my_file] construction is another Python idiosyncrasy, the "list comprehension." It allows a given item that contains other items (here, my_file and the lines it contains) to be iterated through, and to allow each iterated element (that is, each x) to be processed and automatically appended into a list.
- You could write such a thing as a formal for… loop in Python, much as you would in another language. The point is that Python has a way to economically express things like loops that iterate over multiple objects and perform some simple operation on each element in the loop, or work with things that require explicit instantiation and disposal. Constructions like this allow Python developers to balance terseness and readability.
- Python's other language features are meant to complement common use cases. Most modern object types Unicode strings, for instance are built directly into the language. Data structures like lists, dictionaries (i.e., hash maps), tuples (for storing immutable collections of objects), and sets (for storing collections of unique objects) are available as standard-issue items.
- Like C#, Java, and Go, Python has garbage-collected memory management, meaning the programmer doesn't have to implement code to track and release objects. Normally garbage collection happens automatically in the background, but if that poses a performance problem, it can be triggered manually or disabled entirely.
- An important aspect of Python is its dynamism. Everything in the language, including functions and modules themselves, are handled as objects. This comes at the expense of speed (more on that below), but makes it far easier to write high-level code. Developers can perform complex object manipulations with only a few instructions, and even treat parts of an application as abstractions that can be altered if needed.
- Python's use of significant whitespace has been cited as both one of Python's best and worst attributes. The indentation on the second line shown above isn't just for readability; it is part of Python's syntax. Python interpreters will reject programs that don't use proper indentation to indicate control flow.

- Syntactical white space might cause noses to wrinkle, and some people do reject Python out of hand for this reason. But strict indentation rules are far less obtrusive in practice than they might seem in theory, even with the most minimal of code editors, and the end result is code that is cleaner and more readable.

### 9.1.3 Python 2 versus Python 3

- Python is available in two versions, which are different enough to trip up many new users. Python 2.x, the older "legacy" branch, will continue to be supported (i.e. receive official updates) through 2020, and it might even persist unofficially after that. Python 3.x, the current and future incarnation of the language, has many useful and important features not found in 2.x, such as better concurrency controls and a more efficient interpreter.
- Python 3 adoption was slowed for the longest time by the relative lack of third-party library support. Many Python libraries supported only Python 2, making it difficult to switch. But over the last couple of years, the number of libraries supporting only Python 2 has dwindled; most are now compatible with both versions. Today, there are few reasons against using Python 3.

# 10 CONCLUSION AND FUTURE SCOPE

## 10.1 CONCLUSION

Chatbots are the new Apps! As we have discussed in the above deliverables, this project brings the power of chatbots to Yioop and enriches its usability. Chatbots in Yioop can give a human like touch to some aspects and make it an enjoying conversation. And they are focused entirely on providing information and completing tasks for the humans they interact with. The above mentioned functionality in all the deliverables is implemented and pushed into Yioop code. By implementing the above mentioned deliverables I was able to add a basic chatbot functionality in to the Yioop. I.e., configuring and creating accounts for bot users with bot settings which is mentioned in deliverable 2, activating a bot whenever a user asks for it via post in a thread which is discussed in deliverable 3 and as I discussed in deliverable4, I have implemented a simple weather chatbot that gives weather information whenever a user ask and Fig. 3 tells that I was also able to converse with the bot in Yioop. I intend to enhance the system developed so far in CS298. Next step towards

building chatbots involve helping people to facilitate their work and interact with computers using natural language or using set of rules. Future Yioop chatbots, backed by machine learning technology, will be able to remember past conversations and learn from them to answer new ones. The challenge would be conversing with multiple bot users and multiple users.

## 10.2 FUTURE SCOPE

There are limitations to what has been currently achieved with chatbots. The limitations of data processing and retrieval are hindering chatbots to reach their full potential. It is not that we lack the computational processing power to do so. However, there is a limitation on "How" we do it. One of the biggest examples is the retail customer market. Retail customers are primarily interested in interacting with humans because of nature of their needs. They don't want bots to process their needs and respond accordingly. AI is an emerging technology and we can use it to achieve a better efficiency in chatbots.

# 11 REFERENCES


[1] Ranoliya B., Raghuwanshi N., Singh S.,"Chatbot for University related FAQs," International Conference on Advances in Computing, Communications and Informatics 2017 (ICACCI), Udupi, India, pp.1525-1530

[2] Al Mahmudar Rahman, Al Abdullah Mamun,"Programming Challenges of Chatbot: Current and Future Prospective", IEEE Region 10 Humanitarian Technology Conference 2017 (R10-HTC), Dhaka, Bangladesh, pp.75-78

[3] Albayrak N., Ozdemir A., Zeydan E., "An overview of Artificial Intelligence Based Chatbots and an Example Chatbot Application", Signal Processing and Communications Applications Conference 2018 (SIU), Izmir, Turkey

[4] Molnar G., Szuts Z., "The Role of Chatbots in Formal Education", International Symposium on Intelligent Systems and Informatics 2018 (SISY), Subotica, Serbia, pp.197-201

[5] http://e.wikipedia.org/wiki/Chatterbot

[6] http://www.alicebot.org/