

**A
MINIPROJECT REPORT
ON
"AN EXPERT SYSTEM OF HELP DESK MANAGEMENT"
OF
T.E.(Computer Engineering)
(Academic Year: 2021-2022)**

**SUBMITTED BY
OMKAR SAMEER JADHAV (T211048)**

**GUIDED BY
Prof. Balaji Chaugule**



**Department of Computer Engineering
Zeal Education Society's**

Zeal College of Engineering & Research Narhe, Pune-411041

Zeal College Of Engineering & Research, Narhe, Pune
2021-2022

CERTIFICATE



This is to certify that the project entitled “**HELP DESK MANAGEMENT USING CHATBOT**” has been carried out by:

OMKAR SAMEER JADHAV
(T211048)

Under my guidance in partial fulfilment of the engineering of Computer Engineering of Zeal College Of Engineering & Research, Narhe, Pune during Academic year 2020-2021. To the best of knowledge and belief this work has not seen submitted elsewhere for the award of any other Engineering.

Guide
Prof. Balaji Chaugule

H.O.D
Prof. Aparna Mote

INDEX

1. INTRODUCTION

2.TOOLS AND SOFTWARE USED

3.SOURCE CODE SNAPSHOTS AND OUTPUT

4.STEP INVOLVED

5.CONCLUSION

INTRODUCTION

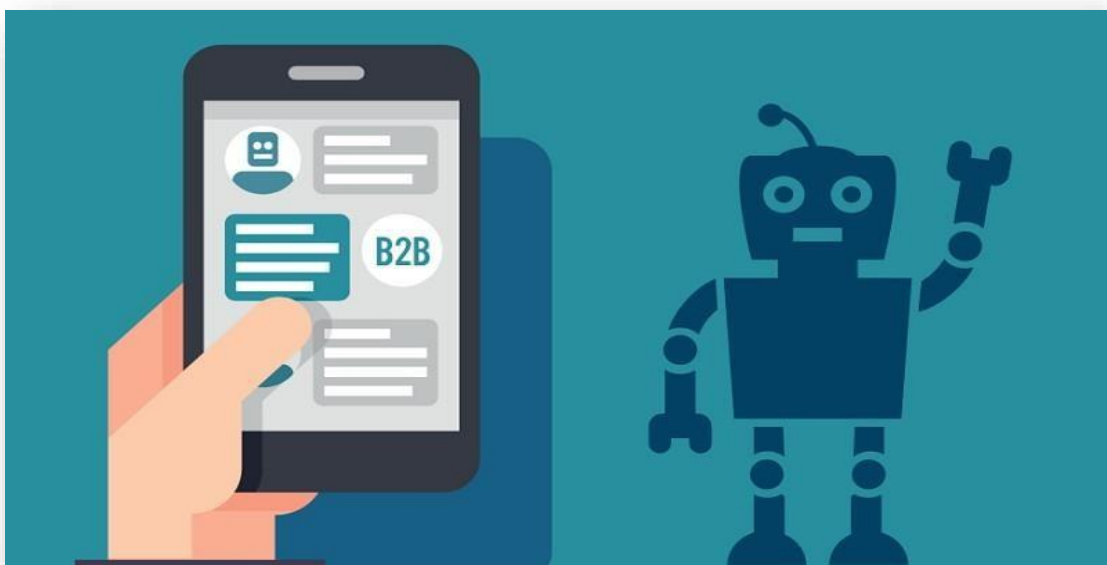
A bot is a software application that performs automated task and chatbots come under the category of bots that live in various chat platforms. A chatbot can converse with humans so the idea of conversation is primary to a chatbot.

Essentially chatbots are of two types:

Command based: Chatbots that function on predefined rules and can answer to only limited queries or questions. Users need to select an option to determine their next step.

Intelligent/AI Chatbots: Chatbots that leverage Machine Learning and Natural Language Understanding to understand the user's language and are intelligent enough to learn from conversations with their users.

[Chatbots](#) run on platforms such a Facebook Messenger, Slack, Telegram, Skype, SMS and even on websites. Each platform has its own salient features which determine the possible ways in which the chatbot can interact with the user, however, the actual behavior of the chatbot is determined by the bot itself.



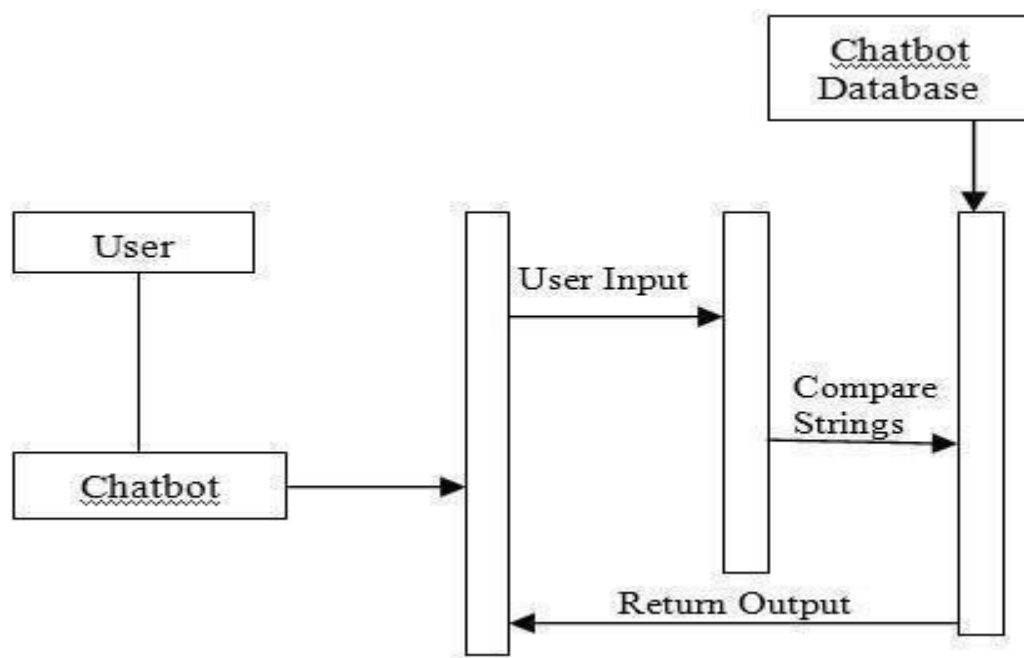
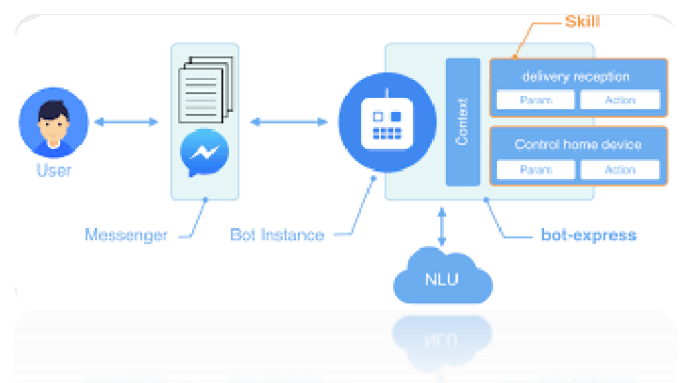
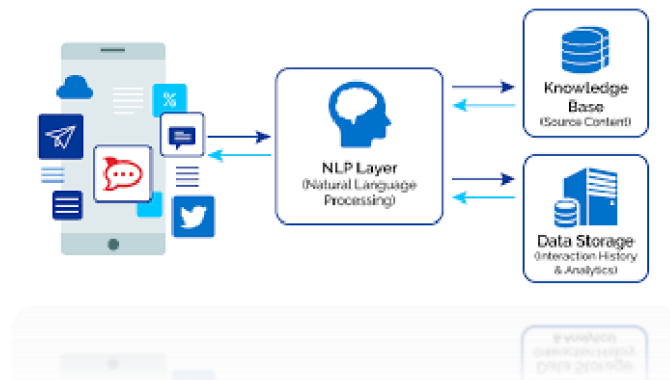


Fig. Architecture of CHATBOT working



TOOLS AND SOFTWARE USED

A. PYTHON

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected.

B. PYCHARM EDITOR

The PyCharm editor is the main part of the IDE that you use to create, read and modify code.

C. TENSORFLOW KERAS

TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow.

D. NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

SOURCE CODE SNAPSHOTS & OUTPUT

```
import random
```

```
R_EATING = "I don't like eating anything because I'm a bot obviously!"
```

```
R_ADVICE = "If I were you, I would go to the internet and type exactly what you wrote there!"
```

```
R_TEMP = "Today temperature is 38°C"
```

```
H_NO="1800-123-456"
```

```
from datetime import datetime now = datetime.now()
```

```
current_time = now.strftime("%H:%M:%S")
```

```
R_TIME=current_time
```

```
def unknown():response = ["Could you please re-phrase that? ", "Jarves", "Sounds about right.", "What  
does that mean?"]random.randrange(4)
```

```
return response
```

```
import re import
```

```
long_responses as long
```

```
def message_probability(user_message, recognised_words, single_response=False, required_words=[]):  
message_certainty =0 has_required_words = True
```

```
# Counts how many words are present in each predefined message for word in user_message: if word  
in recognised_words: message_certainty += 1
```

```
# Calculates the percent of recognised words in a user message  
percentage = float(message_certainty) / float(len(recognised_words))
```

```
# Checks that the required words are in the string  
for word in required_words: if word not in
```

```
user_message:has_required_words = False break
```

```
# Must either have the required words, or be a single response
```

```
if has_required_words or single_response: return int(percentage * 100) else:  
return 0
```

```
def check_all_messages(message): highest_prob_list = { }
```

```
# Simplifies response creation / adds it to the dict
```

```
def response(bot_response, list_of_words, single_response=False, required_words=[]):  
nonlocal highest_prob_list    highest_prob_list[bot_response] = message_probability(message,  
list_of_words, single_response, required_words)
```

```
# Responses _
```

```
-----
```

```
response('Hello!', ['hello', 'hi', 'hey', 'sup', 'heyo'], single_response=True) response('See you!', ['bye',  
'goodbye'], single_response=True) response('I\'m doing fine, and you?', ['how', 'are', 'you', 'doing'],  
required_words=['how']) response('You\'re welcome!', ['thank', 'thanks'], single_response=True)  
response('Thank you!', ['i', 'love', 'code', 'palace'], required_words=['code', 'palace'])
```

```
# Longer responses
```

```
response(long.R_ADVICE, ['give', 'advice'], required_words=['advice']) response(long.R_EATING,  
['what',  
'you', 'eat'], required_words=['you', 'eat']) response(long.R_TEMP, ['what', 'today', 'temperature'],  
required_words=['today', 'temperature']) response(long.H_NO, ['what', 'helpline', 'number'],  
required_words=['helpline',  
'number']) response(long.R_TIME, ['what', 'time'], required_words=['time'])
```

```
best_match = max(highest_prob_list, key=highest_prob_list.get)
```

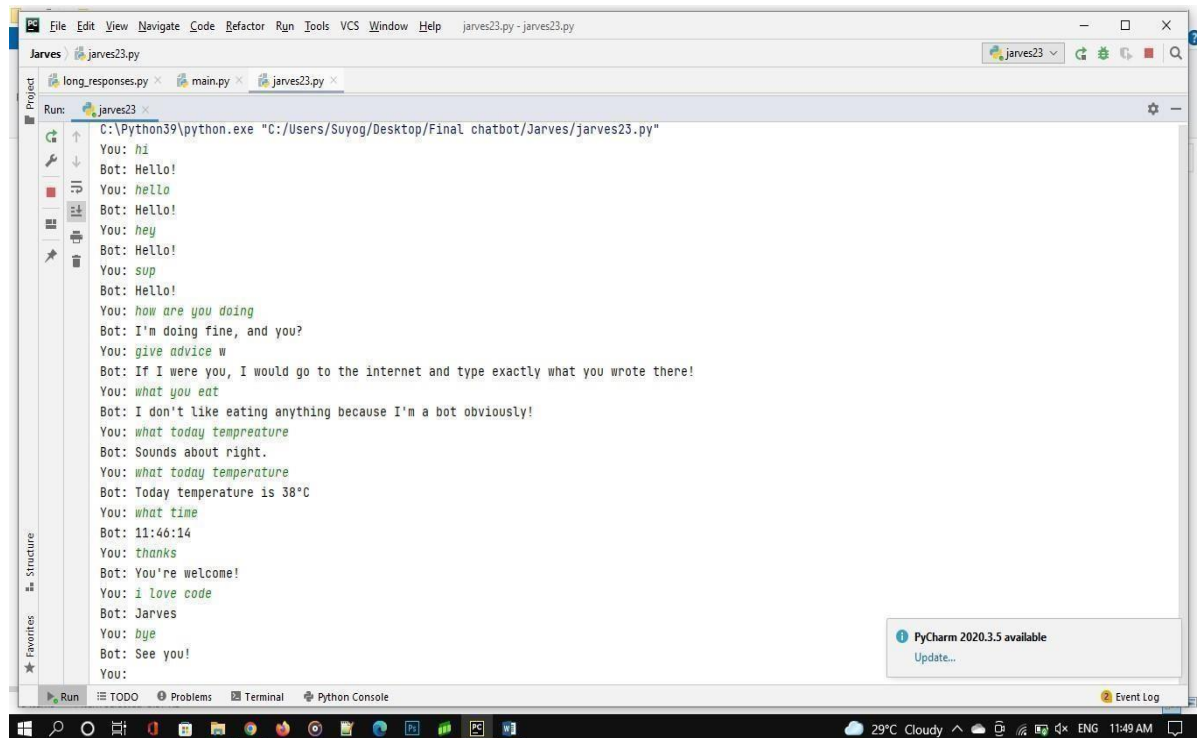
```
# print(highest_prob_list)
```

```
# print(f'Best match = {best_match} | Score: {highest_prob_list[best_match]}')
```

```
return long.unknown() if highest_prob_list[best_match] < 1 else best_match
```

```
# Used to get the response def get_response(user_input): split_message = re.split(r'\s+|[,;?!.-]\s*',  
user_input.lower()) response= check_all_messages(split_message)  
return response
```


Testing the response system while True: print('Bot: ' + get_response(input('You: ')))



The screenshot shows the PyCharm IDE interface. The top toolbar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name is 'Jarves'. The Run console shows the following conversation:

```
C:\Python39\python.exe "C:/Users/Suyog/Desktop/Final chatbot/Jarves/jarves23.py"
You: hi
Bot: Hello!
You: hello
Bot: Hello!
You: hey
Bot: Hello!
You: sup
Bot: Hello!
You: how are you doing
Bot: I'm doing fine, and you?
You: give advice w
Bot: If I were you, I would go to the internet and type exactly what you wrote there!
You: what you eat
Bot: I don't like eating anything because I'm a bot obviously!
You: what today temprature
Bot: Sounds about right.
You: what today temperature
Bot: Today temperature is 38°C
You: what time
Bot: 11:46:14
You: thanks
Bot: You're welcome!
You: i love code
Bot: Jarves
You: bye
Bot: See you!
You:
```

A notification in the bottom right corner states: 'PyCharm 2020.3.5 available' with an 'Update...' link. The bottom status bar shows '29°C Cloudy', 'ENG', and '11:49 AM'.

STEPS INVOLVED

1. Importing the libraries
2. Importing the data
3. Data pre-process-sing
4. Input Length, Output Length and Vocabulary
5. Neural Network
6. Model Analysis
7. Testing
8. Re-iteration

CONCLUSION

The chatbot has been successfully coded with the help of existing tools and software such as Pycharm, Python, Tensor Flow etc. Chatbot responds to the messages typed with higher accuracy but some discrepancies are seen when unconventional language or incomplete sentences are used. Further scope of improvement can be done where amount of data for training can be increased and Chat Bot may perform with better accuracy. Such Chat Bots can be used in various sectors where its not humanely possible to provide uptime of 24/7 round the clock, while queries and doubts can be solved by Chat Bots in absence of actual human. Hence its extremely useful Artificial Intelligent product with tremendous scope and scalability.