Chapter 1

1. Introduction

Chat-bot for HR department will be able to answer questions, query results. Chatbot will be able to answer questions related to salary, attendance, product growth as well as it will give suggestions if it doesn't get proper answers for the user's query. Chatbot will have full HR data. Chatbot will be able to answer questions based on attendance like getting the attendance of a particular company location also can give suggestions like if the user wants an employee who can work on a particular technology and if no such employee exists chatbot will give suggestions of employees who can work on similar technologies. Chatbot gives response in various formats like text, tables, graphs, etc.

1.1. Need:

Today's world is running very fast, so HRs don't have much time to look personally into the database to find answers to their queries. They need some fast approach to do their work. There is a need for an assistant who can work faster. So it is better if they have a virtual computer-based assistant which can answer fast to their queries. A simple virtual assistant is a Chatbot. In the past few years, Chatbots have been very popular because of their fast and accurate response capability and less need for any physical resources as they are not paid. Using Chatbot will reduce and make faster the work of HR people effectively increasing the productivity of them.

1.2. Problem Statement:

To overcome with these problems, we have designed a Chatbot for HR department to give virtual assistance to HR people to let them simplify their work and increase their productivity.

1.3. Aim and Objective:

- **AIM**:

The Chatbot for HR department aims to provide efficient and accurate answers for queries asked by HRs from the database using Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA).

- **OBJECTIVE:**

  To give virtual assistance to HRs so their work could be simplified and productivity could be increased.

1.4. Application and Scope:

- **APPLICATION:**

A HR can ask queries related to employee details and Chatbot will give response to those queries

- **SCOPE:**

This system answers to queries of HR people using AIML and LSA. Chatbot gives responses in different forms like texts, tables, graphs, etc. If chatbot doesn’t find a proper answer to the user’s query it gives suggestions.

Chapter 2

2.Literature Surveyed:

2.1 Literature Review:

1. Chatbot for University Related FAQs

Chatbots are programs that mimic human conversation using Artificial Intelligence (AI). It is designed to be the ultimate virtual assistant, entertainment purpose, helping one to complete tasks ranging from answering questions, getting driving directions, turning up the thermostat in smart home, to playing one's favourite tunes etc. Chatbot has become more popular in business groups right now as they can reduce customer service cost and handles multiple users at a time. But yet to accomplish many tasks there is need to make chatbots as efficient as possible. To address this problem, in this paper we provide the design of a chatbot, which provides an efficient and accurate answer for any query based on the dataset of FAQs using Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA). Template based and general questions like welcome/ greetings and general questions will be responded using AIML and other service based questions uses LSA to provide responses at any time that will serve user satisfaction. This chatbot can be used by any University to answer FAQs to curious students in an interactive fashion.

User discussion as a rule begins with welcome or general questions. User inquiries are first taken care by AIML check piece to check whether entered inquiry is AIML script or not. AIML is characterized with general inquiries and welcome which is replied by utilizing AIML formats. This operation is divided into three parts:

- User post the query on chatbot
- Processing is done on the user's query to match the predefined format by the developer
- Pattern matching is performed between user entered query and knowledge (pattern).

As discussed earlier, user can post their query on chatbot and response is generated based on pattern matching techniques presented in this paper.

2. Automated Thai-FAQ Chatbot using RNN-LSTM

In the e-commerce model that has online customer service, such as email or live chat, customers mostly use live chat because it is fast and comfortable. Thus, a company needs to hire and pay for admins. However, this incurs the problem that admins need to spend an extensive amount of time for writing an answer and customers have to wait for the answers. Several chatbots are available, but they require users to set up key phrases manually. In this article, we propose and develop a Frequently Asked Questions (FAQs) Chatbot which automatically responds to customers by using a Recurrent Neural Network (RNN) in the form of Long Short- Term Memory (LSTM) for text classification. The experimental results have shown that chatbot could recognize 86.36% of the questions and answer with 93.2% accuracy.

1) Preparing data

In our case, we have 2,636 pairs of questions and answers. We then manually categorized such pairs into 80 classes (according to the number FAQ types) and labeled them with an integer. Then split questions and answers. Questions were used for training AI while the answer would be prepared for replying to customers

2) Pre-processing

Pre-processing consists of three main components: tokenization, mapping dictionary, and zero padding. Tokenization or word segmentation is an essential task in natural language processing (NLP) for the Thai language that does not have word boundaries. After the text was segmented into words, each word would be mapped to an integer by dictionary index for processing. We would get the length of the list of integers equal to the number of words of the text. However, the classification model needed a fixed length of the input, so we used zero padding to make every input have the same length.

3) Classification Model

The classification model is a neural network that takes an input from pre-processing for learning to categorize the questions. It consists of three layers. First, the embedding layer is the NLP module where words (an integer) are mapped to vectors of real numbers that learn representation for predefined fixed sized vocabulary from a corpus of text. The embedding visualization in two dimensions after training is shown in Fig. 3. Second, the long short-term memory (LSTM) layer is a particular kind of recurrent neural network (RNN), which is capable of learning sequential data such as text and video. LSTM enables RNN to remember inputs over a long period. Third, Dense layer (Output layer) with softmax activation function is used in various multiclass classification methods. The softmax activation function in the output layer represents a categorical distribution over class labels and obtaining the probabilities of each input belonging to a label. Because of softmax activation function is used at the output layer, we have to encode the label of questions to one-hot format for the learning process of the model.

3. A Pilot Study Integrating an AI-driven Chatbot in an Introductory Programming Course

As AI software tools become more commonplace, their potential to transform the student experience has greatly increased. By integrating these tools into a course, students can begin to receive real-time, around the clock, instructor moderated support. The potential benefits for students are significant. In addition to increasing their ability to work constructively on their schedule, it has the potential to engage students who might otherwise avoid directly interacting with another person. Before these tools can be meaningfully actualized into a course, work must be done to build a knowledge base about the course.

For this research, the team sought to develop an intelligent chatbot interface for an introductory computer programming course. The interface had an initially limited knowledge base with the intent that it would be populated based on students' interactions with the chatbot. This model allowed the bot to evolve with the needs of the students. This paper seeks to present the methodology for how the chatbot was developed and integrated into the course, how the knowledge base was developed, the usage during the pilot, and the next steps for improving the chatbot’s interface. Additionally, the paper will discuss the mechanisms added to handle issues such as false-positive responses and how faculty may be able to integrate such tools into their own courses as supplementary assistance.

EduBot was used for 48 interactions by 21 unique users (31% of possible users) in 33 different usage sessions. Six of the 21 unique users initiated more than one session, with the other 15 only using the chatbot once. A majority of these interactions occurred within the first third of the academic term.The initial knowledge database for EduBot was only populated with common MATLAB functions used in the Introduction to Computing for Engineers course. As a result, it was expected that there would be a high percentage of “No- Answer Found” responses from EduBot. Overall, these responses comprised 33.3% of the chatbot’s responses.

2.2: Existing System:

The previous systems contain a large database but it makes difficult for HR to find the data of employee.

Previously when chatbots are not present HR's uses the forms, Excel. It was complicated for finding the data of employee. Because of chatbot getting record are easy now.

2.3: Summary of Literature Review:

Sr.no	Title	Author	Publication	Approach
1	Chatbot for University Related FAQs	Bhavika R. Ranoliya , NidhiRaghuwanshi and Sanjay Singh	International Conference on Advances In Computing Communications and Informatics 2017	They developed a interactive chatbot for University related Frequently Asked Questions (FAQs). User discussion as a rule begins with welcome or general questions. User inquiries are first taken care by AIML check piece to check whether entered inquiry is AIML script or not. AIML is characterized with general inquiries and welcome which is replied by utilizing AIML formats.
2	Automated Thai-FAQ Chatbot using RNN-LSTM	PanitanMuangkammuen,NarongIntiruk ,Kanda RunapongsaSaikaew	22nd International Computer Science and Engineering Conference 2018	They proposed and developed a Frequently Asked Questions (FAQs) Chatbot which automatically responds to customers by using a Recurrent Neural Network (RNN) in the form of Long ShortTerm Memory (LSTM) for text classification.
3	Intelligent Chatting Service Using AIML	Saqib G, Faizan , Ghatte N	IEEE International Conference on Current Trends toward Converging Technologies, 2018	AIML limits the bots to perform mathematical calculations, provide information about weather, news, recent updates etc. This paper describes the development of chat bots hosting service. This hosting website also

				provide API's for Weather, News, Dictionary, web encyclopedia, Mathematical Calculations and much more and a global file to keep all bots updated that are hosted on the website.

Table 2.3. Summary of Literature Review

Chapter 3

3.Requirement Analysis

User:-

Employee working in the HR department is the user of this system. Employee enters queries in application and application gives the response to the user.

Application:-

This system works using AIML and LSA technologies. This application gives inputs from the user and processes it to find out the proper meaning of the user's queries. Then it gives responses in various formats such as text, graphs, tables, etc.

Chapter 4

4.System Design

4.1: UML diagram

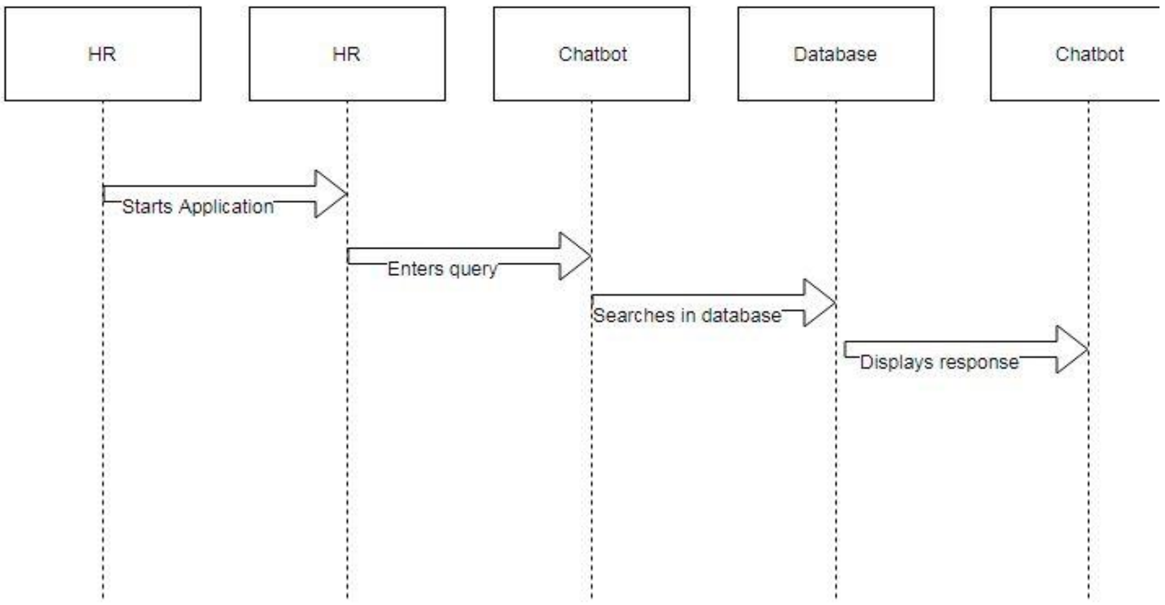


Figure 4.1. UML diagram

4.2:DFD

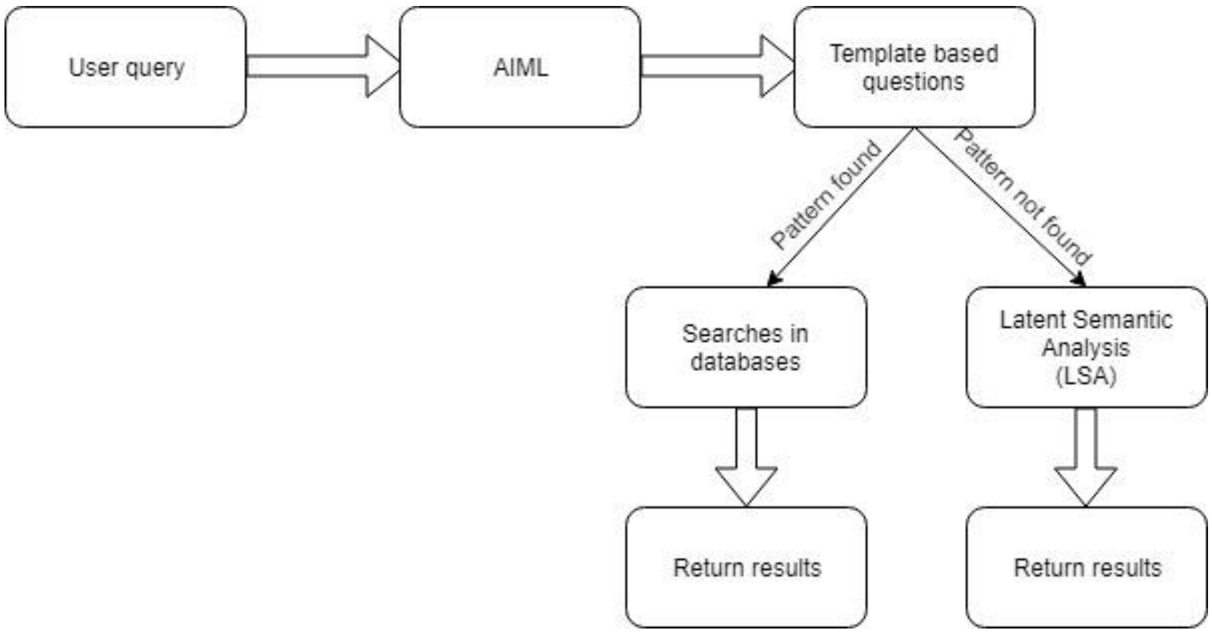


Figure 4.2. DFD

4.3: ER diagram

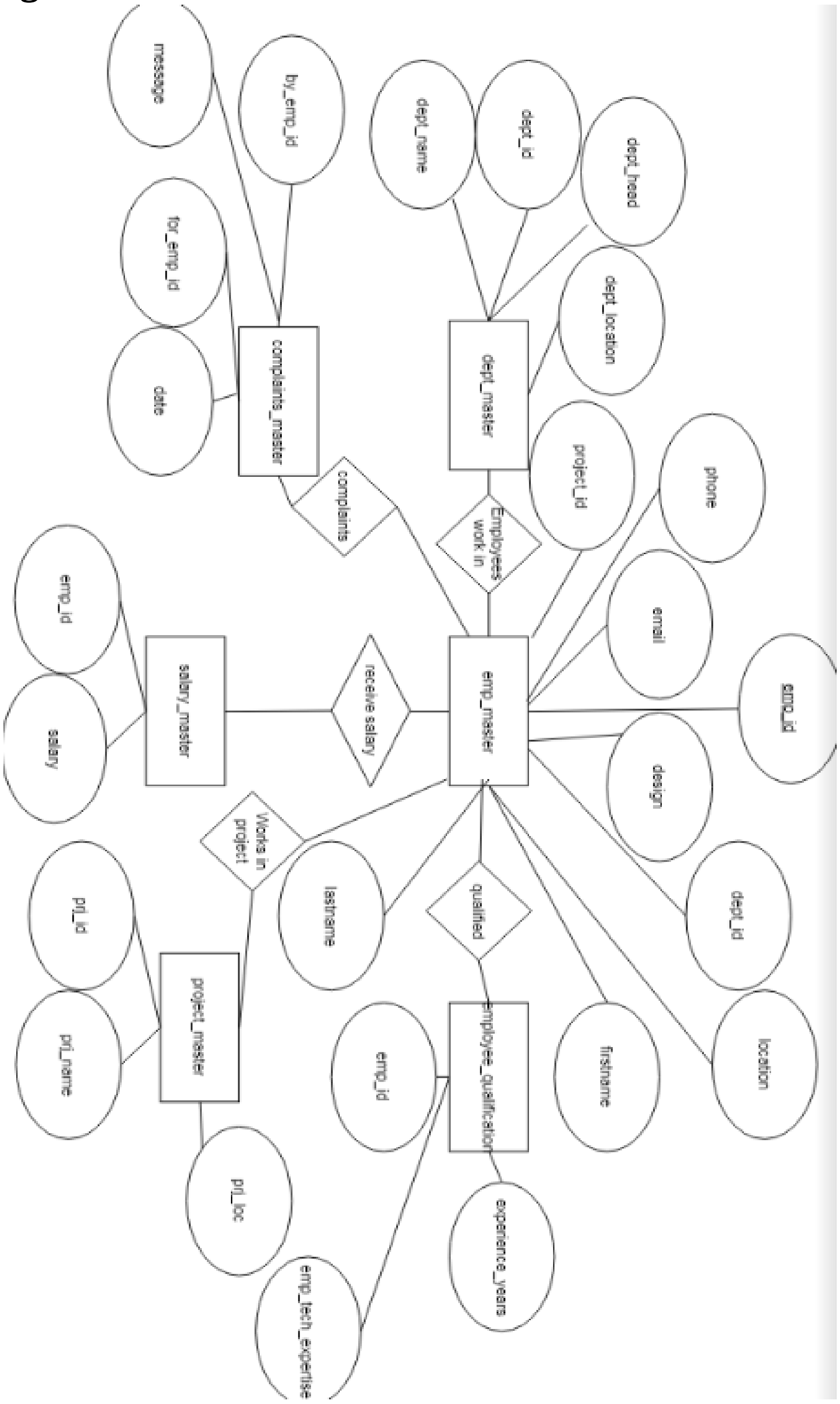


Figure 4.3. ER diagram

Chapter 5

5.Hardware and Software Used

5.1. Software Requirements:

1. **AIML**
2. **LSA**
3. **Python 2.7**
4. **Django**

5.2. Hardware Requirements:

1. **Computer with minimum 4GB RAM**

Chapter 6

6.Implementation

This project proposed a chatbot that will help the HR department to find answers to their queries by just typing in the chat window. This system uses technologies Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA). The workflow of the proposed system is shown in Figure 1. Workflow of Chatbot for HR department.

First, the bot-user has to enter a query in the chatbox. After that AIML developed chatbot will match the pattern and will give a proper response. If the response is to be fetched from the database, it will be handled by LSA. LSA will catch query and try to identify the correct requirements of the user and after that, it will fetch results from the database using python. Once chatbot got the response for the user entered query it will show it as a response in the chatbox.

Section 1) Template based responses:

Template based queries like greetings would be handled by Artificial Intelligence Markup Language (AIML). AIML will match the user entered query with the pattern defined in the AIML files and if pattern matches it will give the stored response.

For example,

Human: Hello

Bot: Hi

Section 2) General queries:

General queries cannot be handled by only AIML, it needs the help of Latent Semantic Analysis (LSA), as we cannot store every pattern user can enter. So we generalize some part of query and other part will be handled by LSA.

For example,

Human: Give me salary of Prakash Parmar

Bot: 65000 Rs.

It can be seen that to get salary of employee different user can type various queries. So we store pattern for **Give** and **salary** by using LSA which will give same result for all salary requests.

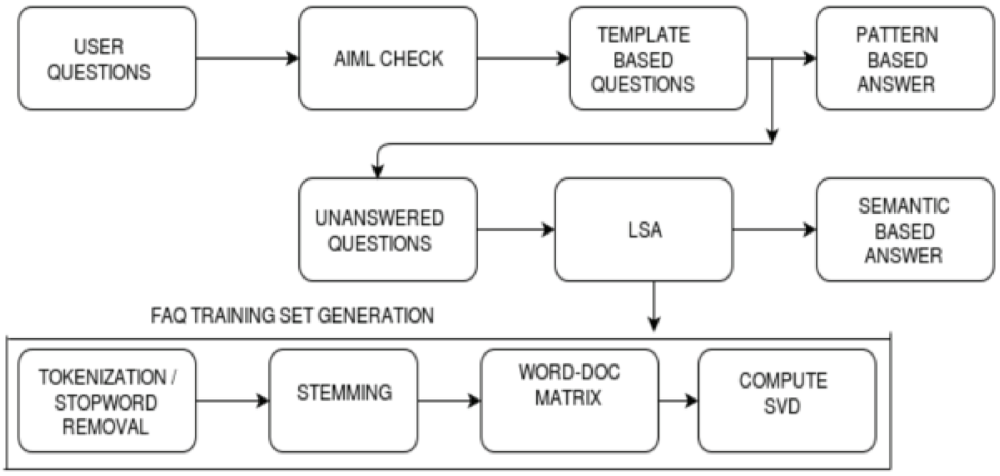
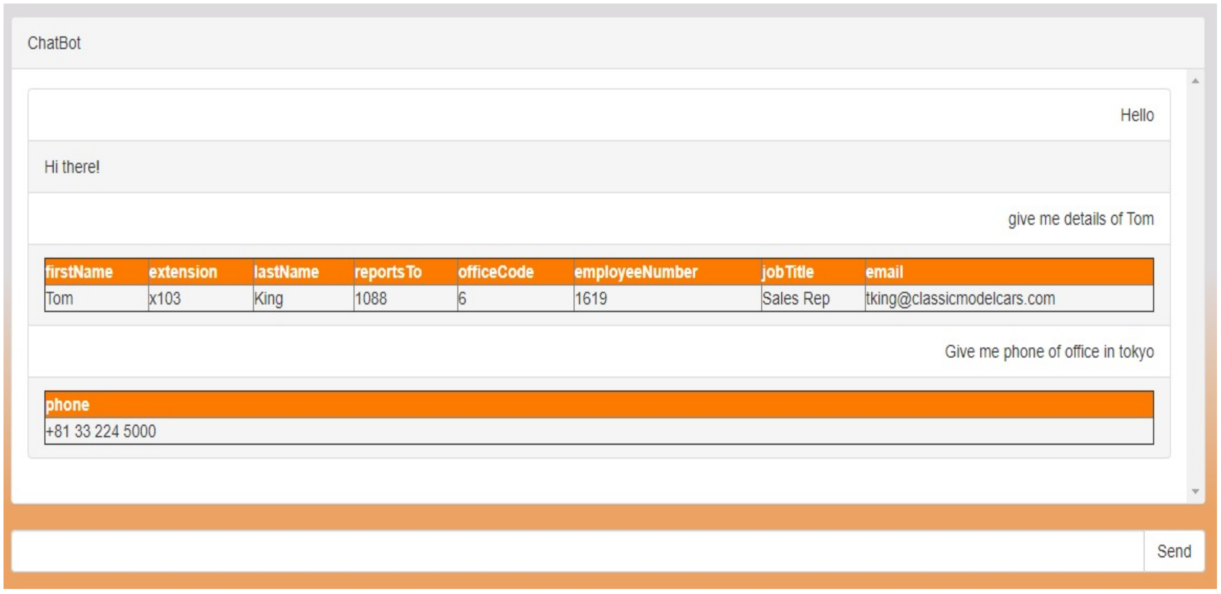


Figure 6.1. Workflow of Chatbot for HR department

Chat window


ChatBot

Hello

Hi there!

give me details of Tom

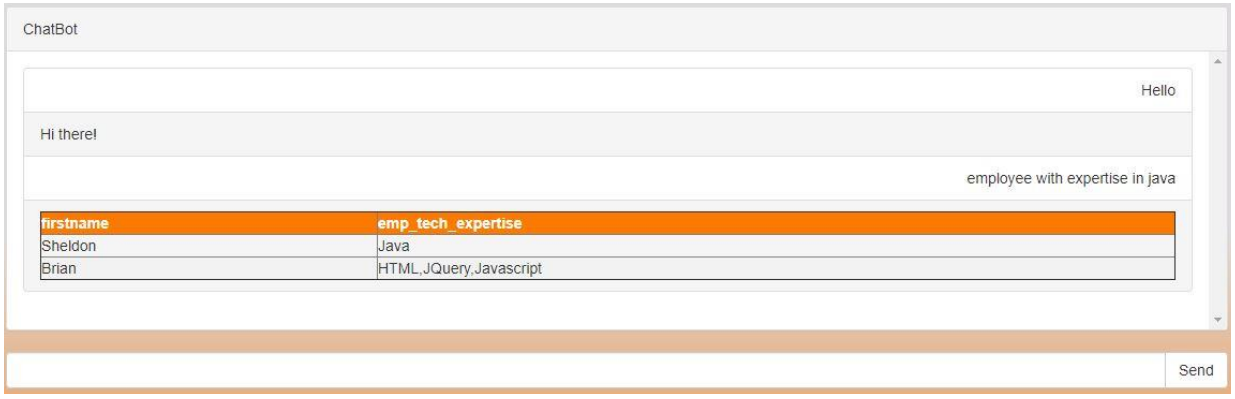
firstName	extension	lastName	reportsTo	officeCode	employeeNumber	jobTitle	email
Tom	x103	King	1088	6	1619	Sales Rep	tking@classicmodelcars.com

Give me phone of office in tokyo

phone
+81 33 224 5000

Send

Figure 6.2. chat Window


ChatBot

Hello

Hi there!

employee with expertise in java

firstname	emp_tech_expertise
Sheldon	Java
Brian	HTML,JQuery,Javascript

Send

Figure 6.3 Employee Expertise

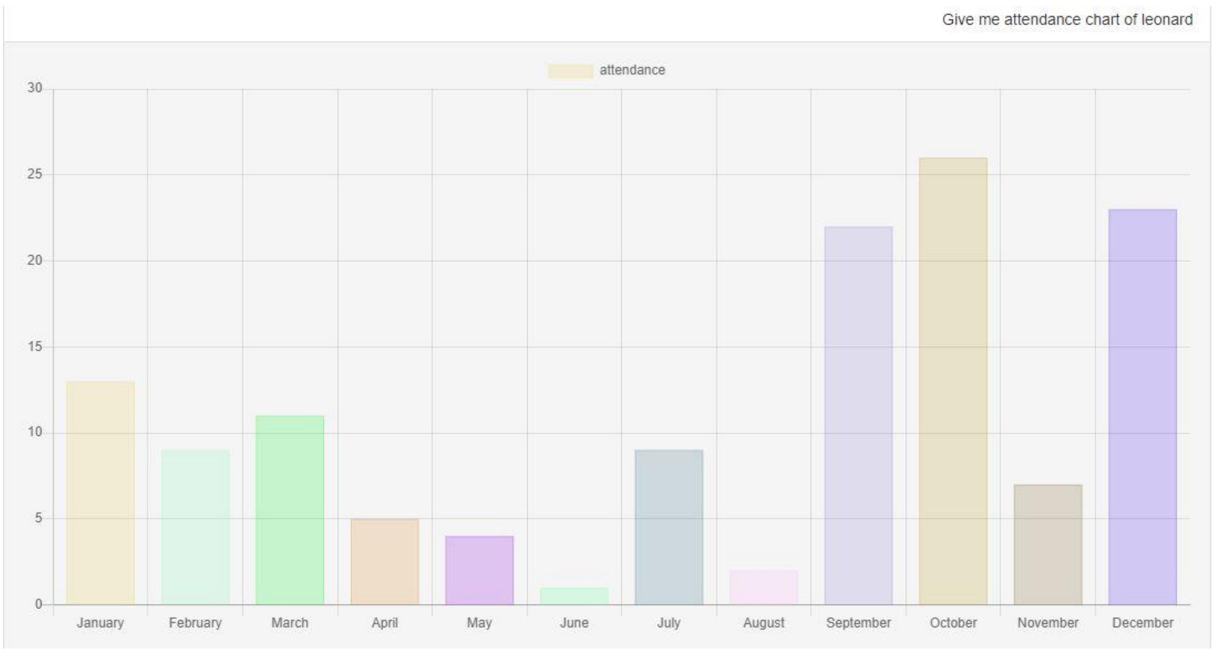
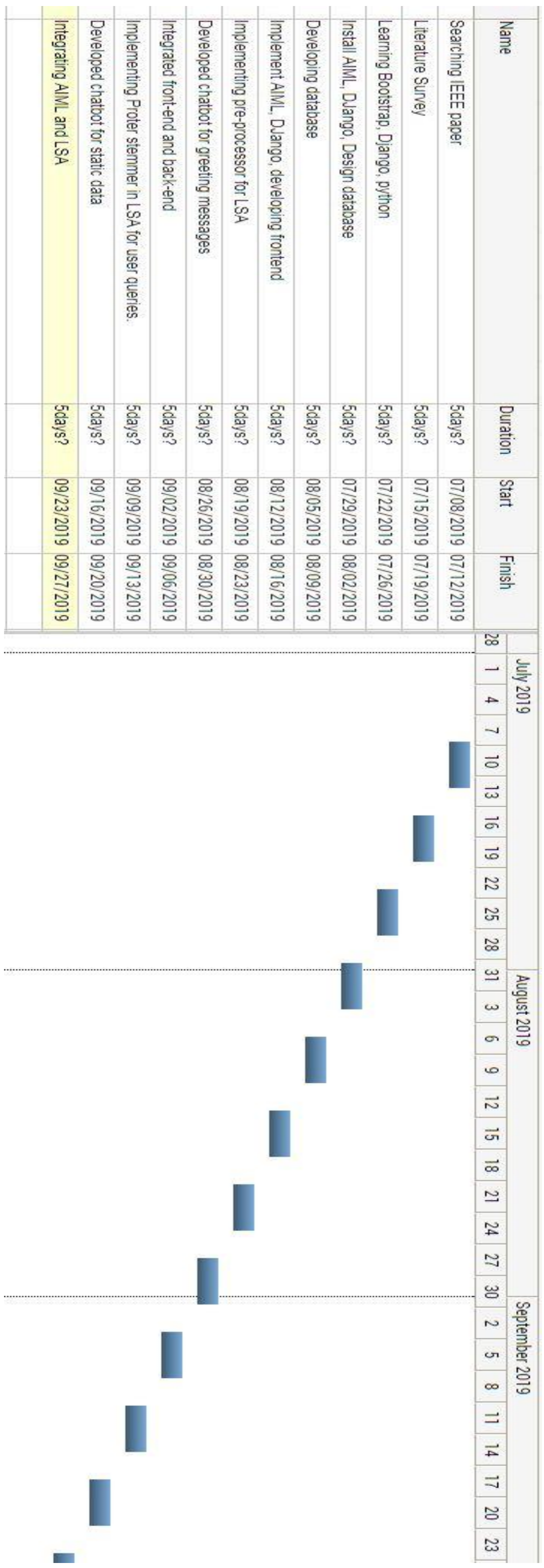

Give me attendance chart of leonard

Figure 6.4 Attendance Chart

Chapter 7

7.Gantt Chart



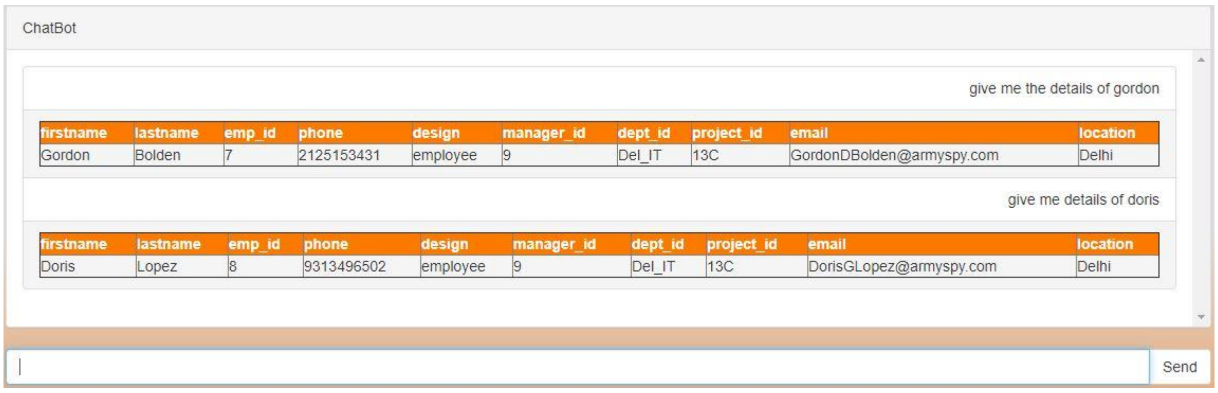
Name	Duration	Start	Finish
Searching IEEE paper	5days?	07/08/2019	07/12/2019
Literature Survey	5days?	07/15/2019	07/19/2019
Learning Bootstrap, Django, python	5days?	07/22/2019	07/26/2019
Install AIML, DJango, Design database	5days?	07/29/2019	08/02/2019
Developing database	5days?	08/05/2019	08/09/2019
Implement AIML, DJango, developing frontend	5days?	08/12/2019	08/16/2019
Implementing pre-processor for LSA	5days?	08/19/2019	08/23/2019
Developed chatbot for greeting messages	5days?	08/26/2019	08/30/2019
Integrated front-end and back-end	5days?	09/02/2019	09/06/2019
Implementing Proter stemmer in LSA for user queries.	5days?	09/09/2019	09/13/2019
Developed chatbot for static data	5days?	09/16/2019	09/20/2019
Integrating AIML and LSA	5days?	09/23/2019	09/27/2019

Figure 7.1 : Gantt chart

Chapter 8

8. Result and Discussion

Chatbot interacts with users giving responses in various forms like text, tables, graphs, etc. If chatbot does not understand the proper meaning of what the users want to ask it asks back users based on how many parts of it chatbot had understood. If chatbot does not get a correct response it gives suggestions to the user.



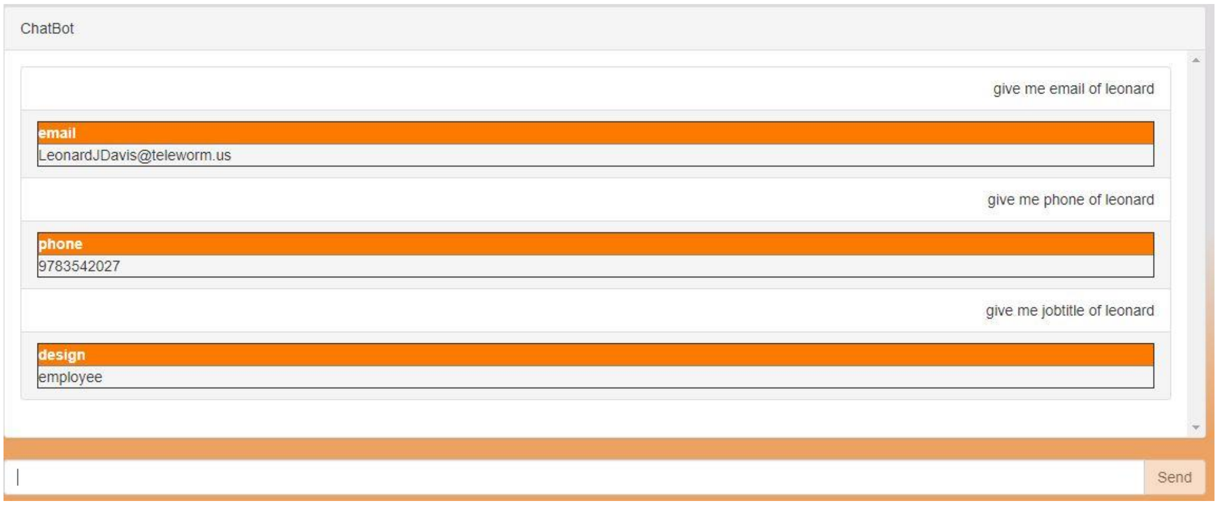
give me the details of gordon

firstname	lastname	emp_id	phone	design	manager_id	dept_id	project_id	email	location
Gordon	Bolden	7	2125153431	employee	9	Del_IT	13C	GordonDBolden@armyspy.com	Delhi

give me details of doris

firstname	lastname	emp_id	phone	design	manager_id	dept_id	project_id	email	location
Doris	Lopez	8	9313496502	employee	9	Del_IT	13C	DorisGLopez@armyspy.com	Delhi

Figure 8.1 Result in table format



give me email of leonard

email
LeonardJDavis@teleworm.us

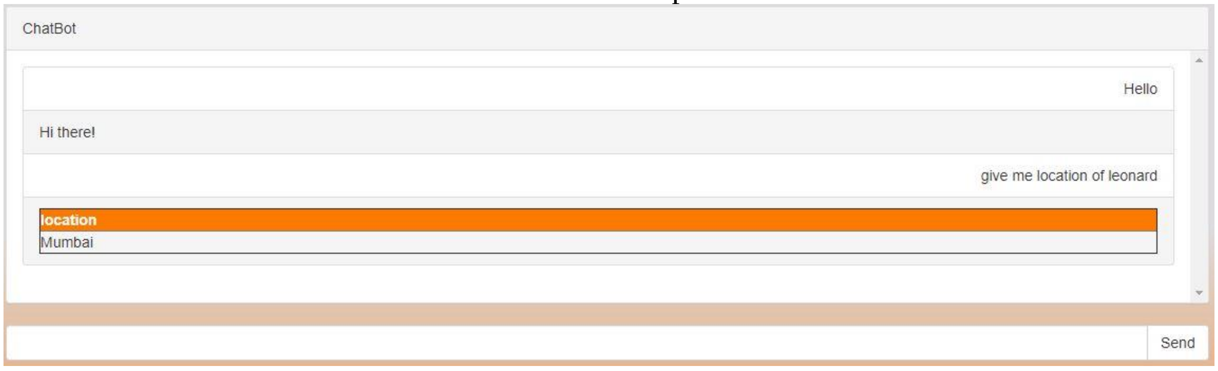
give me phone of leonard

phone
9783542027

give me jobtitle of leonard

design
employee

Figure 8.2 Result in text format



Hello

Hi there!

give me location of leonard

location
Mumbai

Figure 8.3 Result

Chapter 9

9. Conclusion

This project will definitely simplify the work of HR department by finding accurate answers to their questions. Chatbot developed will be continuously developing and learning from the data which is new to the system.

Chapter 10

10. Future Scope

This project can be further modified for other departments as this project only works for the HR department. Frontend can be improved to grab maximum attention of the user. The model can be trained to analyze data and then display results.

References

1] “Chatbot for University Related FAQs”, Bhavika R. Ranoliya ,NidhiRaghuwanshi and Sanjay Singh, International Conference on Advances In Computing Communications and Informatics 2017

2] “Automated Thai-FAQ Chatbot using RNN-LSTM”, PanitanMuangkammuen,NarongIntiruk ,Kanda RunapongsaSaikaew , 22 nd International Computer Science and Engineering Conference 2018

3] “Intelligent Chatting Service Using AIML”, Saqib G, Faizan , Ghatte N , IEEE International Conference on Current Trends toward Converging Technologies, 2018

Appendix

Code:

chat.html

```
{% extends 'base.html' %} {% block content %}
<div class="container">
<div id="chat-body" class="panel panel-default">
<div class="panel-heading">ChatBot</div>
<div id="msg-list-div" class="panel-body">
<ul id="msg-list" class="list-group">
        {% for obj in chat %} {% if obj.user == request.user %}
<li class="text-right list-group-item">{{ obj.message | safe
}}</li>
        {% else %}
<li class="text-left list-group-item">{{ obj.message | safe
}}</li>
        {% endif %} {% empty %}
<li class="text-right list-group-item">No messages yet!</li>
        {% endfor %}
</ul>
</div>
</div>

<form
    id="chat-form"
    method="post"
    action="{% url 'chat:post' %}"
    data-spy="affix"
    data-offset-bottom="10"
>
    {% csrf_token %}
<div id="chat-bottom" class="input-group">
<input type="text" id="chat-msg" name="chat-msg"
class="form-control" />
<span class="input-group-btn">
<input class="btnbtn-default" id="send" type="submit"
value="Send" />
</span>
</div>
</form>
</div>
{% endblock content %}
```

message.html

```
{% for obj in chat %} {% if obj.human_bot %}
<li class="text-right list-group-item">{{ obj.message | safe }}
</li>
  {% else %}
<li class="text-left list-group-item">{{ obj.message | safe
}}</li>
  {% endif %} {% empty %}
<li class="text-right list-group-item">No messages yet!</li>
  {% endfor %}
</div>
```

login.html

```
{% extends "account/base.html" %}
{% load crispy_forms_tags %}

{% load i18n %}
{% load account socialaccount %}

{% block head_title%}{% trans "Sign In" %}{% endblock %}

{% block content %}
<div class="container">
<div class="row">
<div class="col-md-6 col-md-offset-3">
<h1 class="text-center">{% trans "Sign In" %}</h1>

    {% get_providers as socialaccount_providers %}


    {% if socialaccount_providers %}
<p class="text-center">{% blocktrans with site.name as site_name %}Please sign in with one
    of your existing third party accounts. Or, <a href="{{ signup_url }}">sign up</a>
    for a {{ site_name }} account and sign in below:{% endblocktrans %}</p>

<div class="socialaccount_ballot">

<ul class="socialaccount_providers list-inline text-center">
        {% include "socialaccount/snippets/provider_list.html" with process="login" %}
</ul>

<div class="login-or">{% trans 'or' %}</div>

</div>

    {% include "socialaccount/snippets/login_extra.html" %}

    {% else %}
```

```
<p class="text-center">{% blocktrans %}If you have not created an account yet, then please
<a href="{{ signup_url }}">sign up</a> first.{% endblocktrans %}</p>
   {% endif %}


<form class="login" method="POST" action="{% url 'account_login' %}">
     {% csrf_token %}
{{ form|crispy }}
     {% if redirect_field_value %}
<input type="hidden" name="{{ redirect_field_name }}" value="{{ redirect_field_value }}"
/>
     {% endif %}
<button class="primaryActionbtnbtn-default" type="submit">{% trans "Sign In"
%}</button>
<a class="button secondaryAction" href="{% url 'account_reset_password' %}">{% trans
"Forgot Password?" %}</a>
</form>

</div>
</div>
</div>

{% endblock %}
```

signup.html

```
{% extends "account/base.html" %}
{% load crispy_forms_tags %}

{% load i18n %}

{% block head_title%}{% trans "Signup" %}{% endblock %}

{% block content %}
<div class="container">
<div class="row">
<div class="col-md-6 col-md-offset-3">
<h1 class="text-center">{% trans "Sign Up" %}</h1>

<p class="text-center">{% blocktrans %}Already have an
account? Then please <a href="{{ login_url }}">sign
in</a>.{% endblocktrans %}</p>

<form class="signup" id="signup_form" method="post"
action="{% url 'account_signup' %}">
        {% csrf_token %}
{{ form|crispy }}
        {% if redirect_field_value %}
```

```
<input type="hidden" name="{{ redirect_field_name }}"
value="{{ redirect_field_value }}" />
        {% endif %}
<button type="submit" class="btnbtn-primary">{% trans "Sign
Up" %} &raquo;</button>
</form>
</div>
</div>
</div>

{% endblock %}
```

urls.py

```
from django.conf.urls import url
from . import views

urlpatterns = [
url(r'^$', views.home, name='home'),
url(r'post/$', views.post, name='post'),
url(r'^messages/$', views.messages, name='messages'),
url(r'^sample/$', views.sample, name='sample'),
]
```

view.py

```
from django.http import HttpResponse
from django.http import JsonResponse
from django.shortcuts import render
from my_sql_connect import database_fetch
from chat.models import Chat
from aiml_try import aiml_generate
from lsa import lsa_fetch
import aiml

from django.contrib.auth import get_user_model
from django.views.generic import View

from rest_framework.views import APIView
from rest_framework.response import Response

aiml_handle = aiml.Kernel()
response_queue = ""
cant_answer_count = 0

def home(request):
    global aiml_handle
Chat.objects.all().delete()
    chats = Chat.objects.all()
```

```
ctx = {
        'home': 'active',
        'chat': chats
    }
aiml_generate(aiml_handle)
    if request.user.is_authenticated:
        return render(request, 'chat.html', ctx)
    else:
        return render(request, 'base.html', None)



def post(request):
    global aiml_handle,response_queue,cant_answer_count
    if request.method == "POST":
msg = request.POST.get('msgbox', None)
print('Our value = ', msg)

chat_message = Chat(user=request.user,
message=msg,human_bot=True)
        if msg != '':
chat_message.save()

        response = aiml_handle.respond(msg)
if  response == '':
response_queue = response_queue + " " + msg
cant_answer_count = cant_answer_count + 1
            found= 0
            if cant_answer_count> 2:
lsa_Keywords = lsa_fetch(response_queue)
                #print lsa_Keywords
                for keyword_Lsa in lsa_Keywords:
                    response = aiml_handle.respond("Lsa
"+keyword_Lsa)
                    print ("Lsa = "+ response)
                    if(response!=''):
                        found = 1
response_queue = ""
cant_answer_count = 0
                        break
            if found == 0:
                    result = ":( Sorry , response Not Found Can you
Elaborate !!!"

        if(response!=''):
response_queue = ""
cant_answer_count = 0
            result = response
            if response[0] == '@':
                result= database_fetch(response[1:]);
        chat_message2 = Chat(user=request.user,
message=result,human_bot=False)
        chat_message2.save()

        return JsonResponse({'msg': msg, 'user':
chat_message.user.username})
    else:
        return HttpResponse('Request must be POST.')


def messages(request):
    chat = Chat.objects.all()
    return render(request, 'messages.html', {'chat': chat})


class ChartData(APIView):
authentication_classes = []
permission_classes = []

defsample(self, request, format=None):
        salary = salaries.objects.all().count()
        labels = ["Users", "Blue", "Yellow", "Green", "Purple",
"Orange"]
default_items = [qs_count, 23, 2, 3, 12, 2]
        data = {
                "labels": labels,
                "default": default_items,
        }
        return Response(data)
```

urls.py

```
from django.conf.urls import url, include
from django.contrib import admin

# from .views import ChartData

urlpatterns = [
url(r'^admin/', admin.site.urls),
url(r'^accounts/', include('allauth.urls')),
url(r'^', include('chat.urls', namespace="chat")),
]
```