1. **INTRODUCTION:**

In industry, users use database applications to get information related to employees. This needs knowledge about the database to non-technical employees such as the HR department. Other applications developed are not much user friendly and they are not smart enough to evolve on themselves and to understand what user actually wants. Other applications give programmed options restricting user to explore all aspects. 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). Chatbot 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.

**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:**

Searching for details of employees in a huge database is a tedious and hectic task. Also, it is difficult to customize searches and results. User has to type various queries to get different results. The appearance of an application is not interactive.

**1.3. Aims and Objectives**

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

**1.4. Application and Scope:**

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. Thisproject 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.

**2. Literature Surveyed**:

**2.1 Literature review**

* **Chatbot for University Related FAQs[1]**

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. AIMLis 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[1].

* **Automated Thai-FAQ Chatbot using RNN-LSTM[2]**

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

1. 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.

1. 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. 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[2].

* **A Pilot Study Integrating an AI-driven Chatbot in an Introductory Programming Course[3]**

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**

* HRM Employee Data Management
* Recruitment Management Attendance & Leave Management
* Grievance Management
* Salary Management
* Employee Self Service Report Management

HR work is more complex and involved today than it ever has been. There’s so much that goes into the management of employee information, which is used for everything from recruiting and hiring to training, evaluations, and so much more. The importance and manpower behind these tasks make it critical for human resources professionals to have HR management software for more efficient management of HR information.

This is why many companies are now using an HRMS (Human Resource Management System) — a combination of systems and processes that connect human resource management and information technology through HR software. An HRMS can be used in candidate recruiting, payroll management, leave approval, succession planning, attendance tracking, career progression, performance reviews, and the overall maintenance of employee information within an organization.

The automation of repetitive and time consuming tasks associated with human resources management frees up some of the companies most valuable employees and allows the focus to shift to culture, retention, and other highly impactful areas.

Employee general information, photo, note attachment files, medical information, tax information, monthly timesheet, payroll reports (based on the tax information and timesheet), and medical insurance and emergency contacts. Employee benefits from helps you record all medical and detail insurance programs that your company has for your employees. Create an HR payroll report, phone list, employee list, department list reports, and you can add your own report later based on your need. Employee vacation and sick leave management. Employee job history information management. Employee certification and awards management. Employee performance evaluations information management. Employee company asset tracking. Employee resume and training records management.The database records all employee information and you can review it by name list, phone list, or department lists. Benefits information management. Grievance history information (e.g. complaints, dispute etc.) management. Complaint Link is available for raising the complaint. Auto-generated mail will be sent to the employer from HR id stating that your complaint had been received and resolved ASAP. Auto-generated mail will be sent to the employer from HR id stating that your dispute had been received and mention the solution provided to the employee which had to follow mandatory.

**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.

**4. System Design**

**4.1: UML diagram**

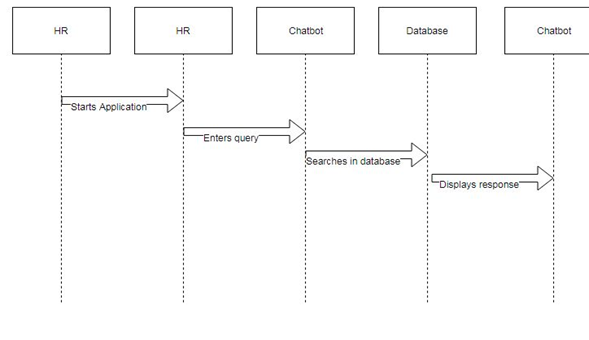
****

Figure 4.1. UML diagram

**4.2:DFD**

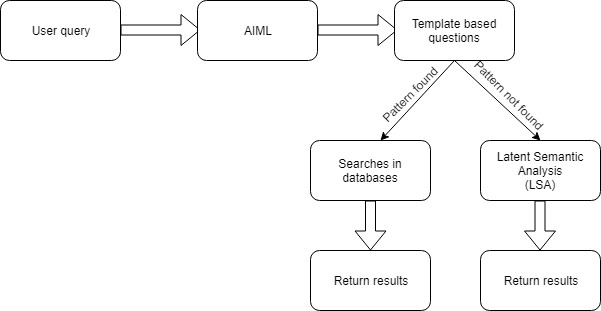
****

Figure 4.2. DFD

**4.3: ER diagram**

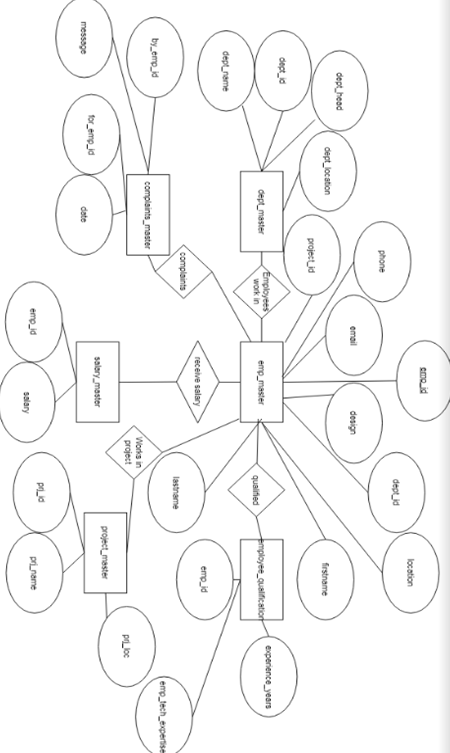
****

Figure 4.3. ER diagram

**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**

**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 6.1. 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[1]. 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[1].

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.

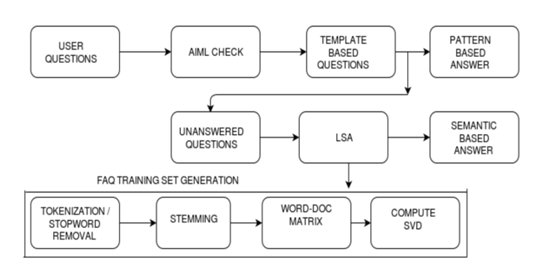
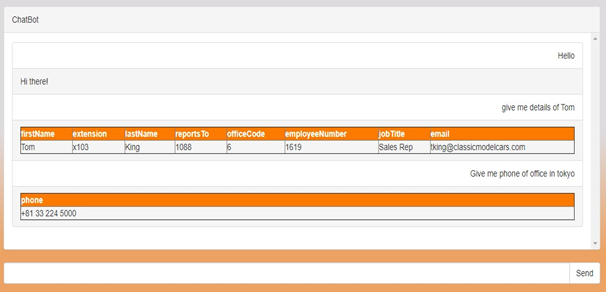
****

Figure 6.1. Workflow of Chatbot for HR department

**Chat window**

****

|  |
| --- |
|  |

Figure 6.2. chat Window

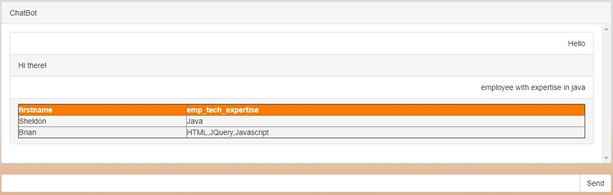
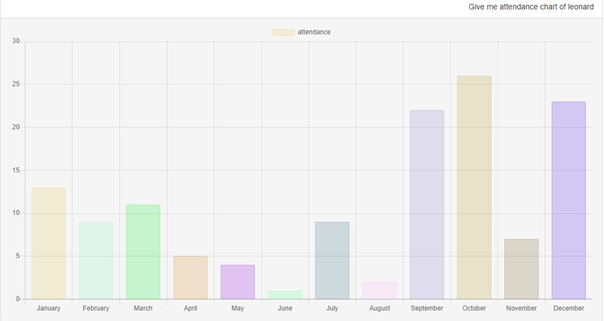
****

Figure 6.3 Employee Expertise



|  |
| --- |
| Figure 6.4 Attendance Chart |

**7. Testing**

**7.1 Test Cases:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S No | Action | Inputs | Expected Output | Actual Output | Test Platform | Test Result | Test Comments |
| 1. | Enter Template Based Message | Template Based Query | Set Response For Entered Query | Set Response For Entered Query | Windows OS,  Chrome Version 79.0. | Pass | Expected Response Returned by Chatbot |
| 2. | Enter Data Retrieval Query | Data Retrieval Query | Data Retrieved From Database Based on Input Query | Data Retrieved From Database Based on Input Query | Windows OS,  Chrome Version 79.0. | Pass | Expected Data Reurned By Chatbot |
| 3 | Enter query not related to employee | Random query | Chatbot asks to re-enter | Chatbot asks to re-enter | Windows OS,  Chrome Version 79.0. | Pass | Expected action taken by chatbot. |
| 4 | Enter non-related query three times | Non-related query | Chatbot understands what user wants and gives proper response | Chatbot understands what user wants and gives proper response | Windows OS,  Chrome Version 79.0. | Pass | Expected action taken by chatbot. |

**7.2 Type of Testing Used:**

For this system, black box testing is used. In black-box testing, the user need not know the technical aspects of the system. Since end-users of this system are employees working in the HR department, they might be unable to understand the internal working of the chatbot. When using an application it is rare for the user to have access to the source code[9]. The test is performed by working in a group and giving different inputs to the chatbot. The response generated by the chatbot is compared with the expected output. For this list of questions were prepared and tested against chatbot. Anomalies were identified and corrected.

**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 has understood. If a chatbot does not get a correct response it gives suggestions to the user.

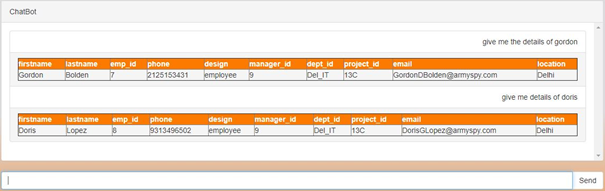
****

Figure 8.1 Result in table format

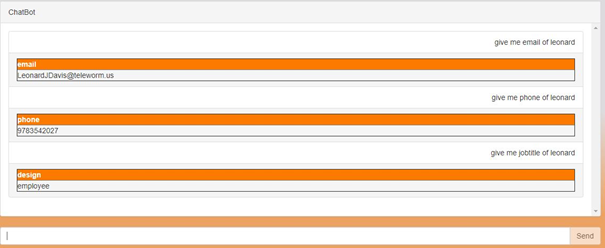


Figure 8.2 Result in text format

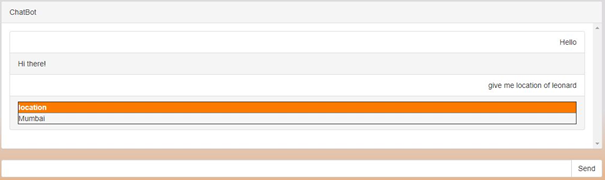


Figure 8.3 Result

**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.

**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 ConvergingTechnologies, 2018

4] "An Efficient Search for Context-Based Chatbots", Ayah Atiyah, Shaidah Jusoh, Sufyan Almajali, Computer Science and Information Technology (CSIT) 2018 8th International Conference on, pp

5] "Implementation of Chatbot for ITSM Application Using IBM Watson", Neha Atul Godse, Shaunak Deodhar, Shubhangi Raut, Pranjali Jagdale, Computing Communication Control and Automation (ICCUBEA) 2018 Fourth International Conference on, pp.

6] "AI Based Student Bot for Academic Information System using Machine Learning", Vijayakumar R, Bhuvaneshwari B, Adith S, Deepika M, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2019.

7] “An e-business chatbot using AIML and LSA”, N T Thomas

Amrita Vishwa Vidyapeetham University, 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)

8] “AI and Web-Based Human-Like Interactive University Chatbot (UNIBOT)”, Neelkumar P. Patel, Devangi R. Parikh, Darshan A. Patel, Ronak R. Patel, 2019 3rd International Conference on Electronics, Communications and Technology.

9]https://d1qmdf3vop2l07.cloudfront.net/new-violin.cloudvent.net/compressed/163d7341cec7c150a8a456fb4a00bb6d.pdf

**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")),

]