1. INTRODUCTION

1.1 Need

This system is useful for people as it allows them to order food from their home without having to go to restaurants. They can save time due to this system and do something more productive in the time saved. The system is also interactive which makes the customer more comfortable. With chatbots, your customers no longer need to make a call to reserve a table, wait for staff to attend to them or wait in line for tables to free up. Restaurants don't need to have a exclusive service executive for the customers either. Bots can be programmed to carry out a myriad of tasks ranging from answering FAQs, making a reservation, ordering food or processing payment. The bot can carry out these tasks in manner similar to a service executive, difference being — it can execute round the clock with zero downtime.

1.2 Problem Statement

Our chatbot is a window into a future where a friend is always on standby just in case you feel hungry. All you have to do is message him and he'll get it delivered right to you.

1.3 Aim and Objectives

1.3.1 Aim

With chatbot your customers no longer need to make a call to reserve a table, wait for staff to attend to them or wait in line for tables to free up. Restaurants don't need to have an exclusive service executive for the customers either.

1.3.2 Objectives

The objective of this system is to:

1. Bots can be programmed to carry out tasks like answering FAQs.
2. Bots can be programmed to make a reservation
3. Bots can be programmed to order food

1.3.3 Applications and Scope

While the staff focuses on preparing and serving food, chatbots can engage with the customers by answering questions related to open and close times, reward points or whether if the restaurant is open on a public holiday. The use cases of chatbot in restaurants rely heavily on the kind of experience restaurants want to offer their visitors.

For millennials, the generation that actively prefers not speaking with others, they can be the perfect fit as they are the ones who, apart from food, also expect a digital experience. This is where restaurants needs to evolve by understanding modern day customer behaviors and expectations with the advent of digital technology.

2. HARDWARE AND SOFTWARE USED

2.1 Software Details

1. AIML

AIML stands for Artificial Intelligence Modelling Language. AIML is an XML based markup language meant to create artificial intelligent applications. AIML makes it possible to create human interfaces while keeping the implementation simple to program, easy to understand and highly maintainable.

2. Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

3. MySQL

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

3. CODING IMPLEMENTATION

Views.py

```
def home(request):

    global aiml_handle
    Chat.objects.all().delete()
    aiml_generate(aiml_handle)
    print "in home successfull"
    chats = Chat.objects.all()
    ctx = {
        'home': 'active',
        'chat': chats
    }
    if request.user.is_authenticated:
        intro = "Welcome to KR food delivery. Order and eat Great food.Please Pick a Food
Category"
        chat_message = Chat(user=request.user, message=intro, human_bot=False)
        chat_message.save()
        msg = "show me the categories"
        response = aiml_handle.respond(preprocess_data(msg))
        if response[0] == '@':
            result_list = database_fetch(response[1:])
            result = json_html(result_list)
        chat_message = Chat(user=request.user, message=result, human_bot=False)
        chat_message.save()
        return render(request, 'order_message.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(preprocess_data(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(
                        preprocess_data("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_list = database_fetch(response[1:])
                result = json_html(result_list)
            elif response[0] == '$':
                result = chart_display(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, 'just_message_only.html', {'chat': chat})
```

Load_Message.html

```
{% extends 'order_page.html' %} {% block content %}
{% load staticfiles%}
 <form id="chat-form"
  method="post"
  action="{% url 'chat:post' %}"
  data-spy="affix"
  data-offset-bottom="10" class="form-container"  style="height: 100%">
    {% csrf_token %}
   <h1>Chat</h1>

   <label for="msg"><b>Message</b></label>

   <div id="mymsglist" class="well" style="overflow: auto;max-height: 250px;text-align:
center;">

{% for obj in chat %} {% if obj.human_bot %}

<div class="row container lighter " style="width: inherit;">
  <div class="col-sm-11" style="text-align:right">{{ obj.message | safe }}</div>
  <div class="col-sm-1" style="padding:0px"><img src='{% static "images/user.png" %}'
alt="Avatar" ></span></div>
</div>

{% else %}

<div class="row container lighter " style="width: inherit;">
  <div class="col-sm-1" style="padding:0px"><img src='{% static "images/chatbot.png" %}'
alt="Avatar" ></span></div>
  <div class="col-sm-11" style="text-align:left">{{ obj.message | safe }}</div>
</div>

{% endif %} {% empty %}
<div class="row container lighter " style="width: inherit;">
  <div class="col-sm-1" style="padding:0px"><img src='{% static "images/chatbot.png" %}'
alt="Avatar" ></span></div>
  <div class="col-sm-11" style="text-align:left">Welcome to KR food delivery. What would you
like to order</div>
</div>
```

```
{% endfor %}


</div>
    <input type="text" id="chat-msg" name="chat-msg" class="form-control" placeholder="Place
an Order" style="margin-bottom:5%">
   <button type="submit" class="btn" id="send" form="chat-form"
value="Submit">Send</button>

   <button type="button" class="btn cancel" onclick="closeForm()">Close</button>
  </form>


  {% endblock content %}
```

Messages.aiml

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

<category>
   <pattern>show menu</pattern>
   <template>@SELECT * FROM menu ;</template>
</category>

<category>
   <pattern>show categori</pattern>
   <template>@SELECT * FROM category;</template>
</category>

<category>
   <pattern>sandvich</pattern>
   <template>@select Name,price from menu where f_c_code = 'sandvich';</template>
</category>

<category>
   <pattern>bread</pattern>
   <template>@select Name,price from menu where f_c_code = 'breads';</template>
</category>
```

```
<category>
   <pattern>dal</pattern>
   <template>@select Name,price from menu where f_c_code = 'dals';</template>
</category>

<category>
   <pattern>dosa</pattern>
   <template>@select Name,price from menu where f_c_code = 'dosas';</template>
</category>


<category>
   <pattern>IceCream</pattern>
   <template>@select Name,price from menu where f_c_code = 'IceCream';</template>
</category>


<category>
   <pattern>Maincourse</pattern>
   <template>@select Name,price from menu where f_c_code = 'Maincourse';</template>
</category>


<category>
   <pattern>pizza</pattern>
   <template>@select Name,price from menu where f_c_code = 'pizza';</template>
</category>

<category>
   <pattern>rice</pattern>
   <template>@select Name,price from menu where f_c_code = 'rice';</template>
</category>

<category>
   <pattern>salad</pattern>
   <template>@select Name,price from menu where f_c_code = 'salads';</template>
</category>

<category>
   <pattern>Starter</pattern>
```

```
   <template>@select Name,price from menu where f_c_code = 'starters';</template>
</category>

<category>
   <pattern>subziyan</pattern>
   <template>@select Name,price from menu where f_c_code = 'subziyan';</template>
</category>

<category>
   <pattern>noodl</pattern>
   <template>@select Name,price from menu where f_c_code = 'noodles';</template>
</category>
```

4. RESULTS AND DISCUSSION

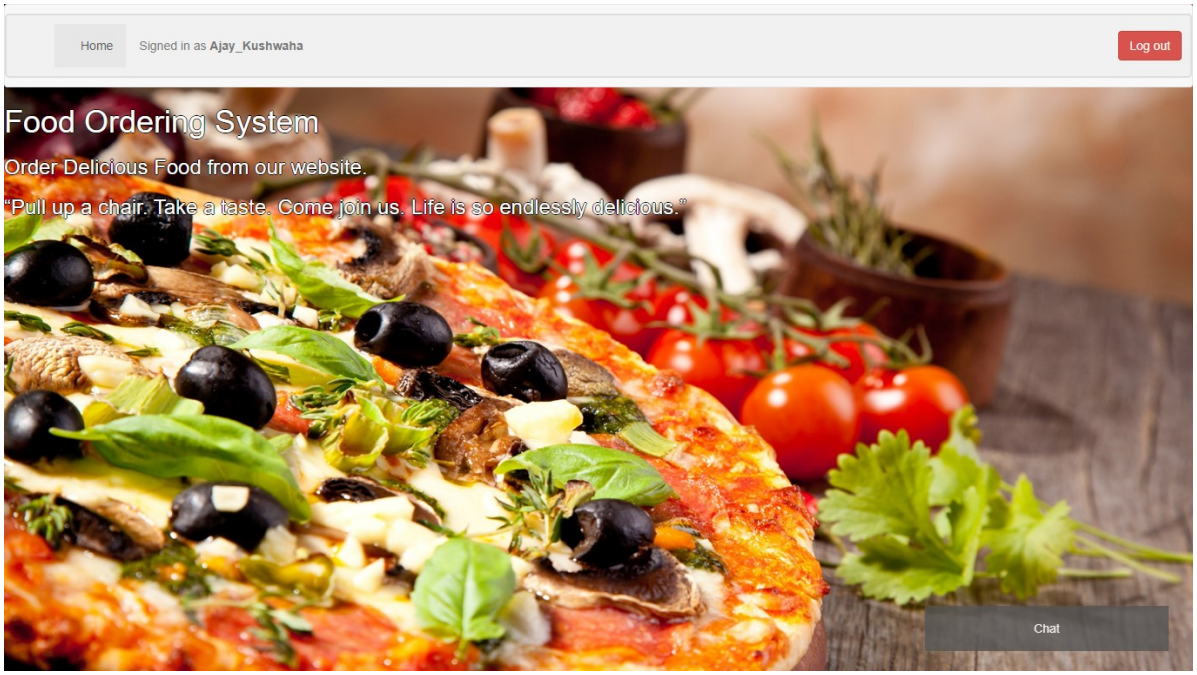

Figure 4.1.Home Page

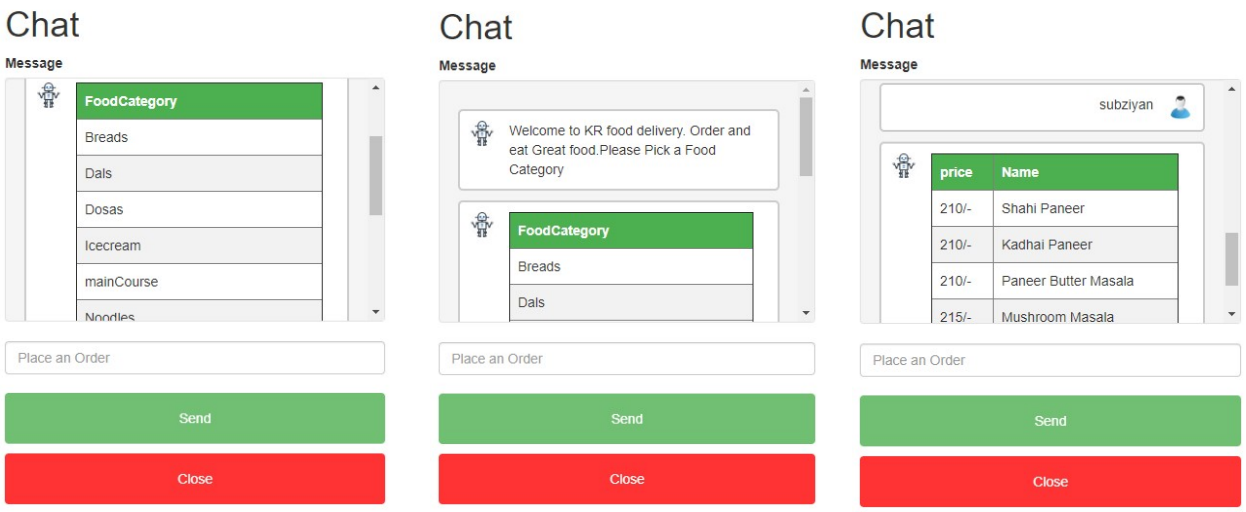

Figure 4.2 Chat ScreenShots

5. CONCLUSION

The idea of this system is to help people order food from home in an interactive way as they normally do from real life restaurants without having to actually go there. The system shows the capabilities of Artificial Intelligence in creating a system that helps in our daily tasks and saving our time in the process.

REFERENCES

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

[2] Panitan Muangkammuen, NarongIntiruk, Kanda Runapongsa Saikaew, “Automated Thai-FAQ Chatbot using RNN-LSTM”, 22nd International Computer Science and Engineering Conference 2018.

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

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

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

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

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