“Hostel Hero: An AIML Chatbot”

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Navneet Thakur (161215)

Under the supervision of

(Dr. Hemraj Saini)

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Hostel Hero: An AIML Chatbot"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2019 to May 2020 under the supervision of **Dr. Hemraj Saini** (Associate Professor, Computer Science &Engineering Department).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]
(Navneet Thakur)

Navneet Thakur (161215)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible] (HemRaj Saini)

Dr. Hemraj Saini

Associate Professor

Computer Science and Engineering and Information Technology

Dated: 20th June 2020

ACKNOWLEDGEMENT

I owe my profound gratitude to my project supervisor **Dr. Hemraj Saini**, who took keen interest and guided me all along in my project work titled - **"Hostel Hero: An AIML Chatbot"** till the completion of the project by providing all the necessary information for developing the project. The project development helped me in research and I got to know a lot of new things in my domain. I am really thankful to him for his support in completion of this project.

TABLE OF CONTENT

ABSTRACT

Interaction with machine or a software has become very common these days as most of the businesses are getting automated and this query related work is also being automated alongside with other technologies. Now in various software related companies' websites, one can see chatbot windows through which you are actually interacting with a machine or software and that software can resolve most of your queries very easily by providing you sufficient and most accurate information with respect to your query.

This project is based on AIML which stands for artificial Intelligence Mark-up Language which is a language used for creating the chatbots. The chatbot created at the end of the project will be able to resolve any kind of query related to JUIT Hostels, whether it is about chief warden or warden or the hostel fee, it can tell the client about it if asked. To build this, I will be using python language to implement the aiml interpreter and then we have to create a form to input the queries of clients, this will be done using HTML. Now to transfer the data entered in the form to python file we need to use a Flask app which can move data between the HTML page and python file. When this is done, we will have to create the personalised aiml files to train our chatbot on them as the chatbot learn from database or previous conversation that are recorded in database and then answer to the query accordingly. once we have the personalized aiml file then we can train out chatbot and the it will be able to answer any query asked by the clients. The query response will be reverted on the html form page.

At the end when the chatbot is supplied with sufficient information and has a well balanced and precise database of knowledge, it can answer all the queries related to JUIT Hostels within a second.

As a bonus, I will be adding the gTTS which is a python module to implement the text-to-speech functionality with the help of which along with presenting the response to the client, the chatbot will also read the response produced by it.

Chapter 01

Hostel Hero: An AIML Chatbot

Introduction

1.1 Introduction:

In today’s world, everything is becoming automated. Every part of society is being automated day by day in order to reduce the human efforts and Artificial Intelligence is playing the most important role in all this. Now a days, we have lots of machines/software that respond to our commands and do certain specific operation. We just need to tell them what to do and they do it very precisely and in vey less time and this is all because of the Artificial Intelligence. For example, we have Google Assistant, Bixby and Siri etc. These all are Artificial Intelligence based chat bots that can chat with us and can do certain specific tasks for us if we tell them to do. In this project, I have also created a chatbot similar to these above examples yet different.

This chatbot can help anyone to understand the rules and regulations of JUIT hostels and can provide them all the necessary information they need about all the hostels of JUIT, like where the hostel buildings are located, who are the respective wardens of each hostel what is their contact numbers, and what are the various do’s and don’ts of the hostels.

For developing this chatbot, I have used the technologies like AIML, python, flask, HTML, CSS, GTTS and xml.

1.2 Motivation:

Whenever someone visit university campus for the first time or someone has been in the campus since a long time, no one can remember all the things about university.

Being a Hosteler student, I have seen that we don’t know many things about our hostels and we need to search for that information on the internet.

So, this motivated me to build a chatbot that can help anyone to get the required information about the hostels very easily and can guide him through the hostel rules.

1.3 Problem Statement:

This project aims for creating a chatbot using AIML and python, that can guide people through all the information related to hostels of JUIT.

This report contains all the information regarding this project like how it is created, what are various challenges faced and what is the final product etc.

1.4 Technical Requirements:

This project has certain generic requirements for its implementation as follow:

a) Windows 10 (64-bit)
b) 4GB RAM
c) Web Browser like Chrome latest version
d) Editor like Visual Studio Code
e) Installed Python
f) Installed Flask
g) Installed GTTS
h) Installed python-aiml Library

1.5 Deliverables:

As a result, I have built a chatbot named "Hostel Hero" through which you can get any kind of information related to JUIT hostels.

You just have to type your query and the chatbot will give the respective information back to you and as a bonus it will also read the whole response for you.

Chapter 02

Literature Survey

2.1 AIML Introduction:

AIML stands for Artificial Intelligence Mark-up Language. AIML is an XML language, which is used for creating natural language software agents called chatbots. It was because of AIML that we were able to create very advanced ELIZA called A.L.I.C.E. There are AIML interpreters available in various languages like in Java, Ruby, python, C++, C#, pascal etc.

AIML is motivated by two things:

1. Creating a chatbot which can interact with us, requires a large amount of writing content, as a reply during conversation.
2. Developing a chatbot requires a great imagination and blend of expected conversation flow and programming skills.

2.2 AIML Tags:

1. **aiml tag**: This tag contains the content of aiml file.

```
E.g.:  <aiml version= “2.0” encoding = “UTF-8”>
        ………Content……
       </aiml>
```

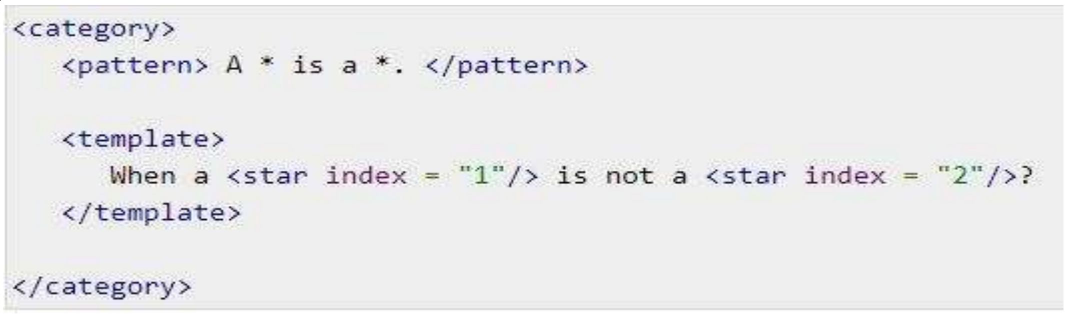
2. **category tag:** This tag defines the basic unit of knowledge in AIML. In this tag, there are generally two tags, first is pattern tag and second is template tag.

```
E.g.: <category>
              ……
      </category>
```

3. **pattern tag:** This tag is used for comparison with the user input. A user can input different type of text and that text is matched with different patterns in aiml files.
   Note: This tag must be the first tag inside the category tag.
   Also, all the things within the pattern tag must be in capital letters.
   Pattern tag is case sensitive.

4. **template tag:** This tag is used for storing specific reply for specific inputs that are given by user. For a given pattern, the chatbot will reply the corresponding template.
   It must be the second tag inside the category tag.

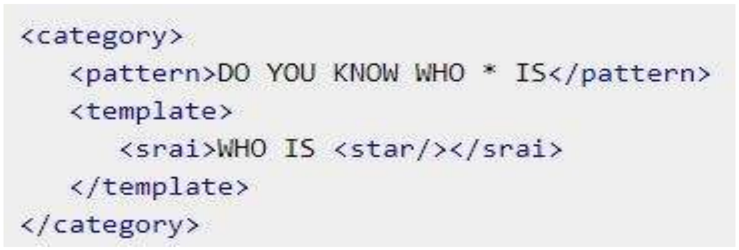
5. **star tag:** This tag is used to get the wildcard input entered by users. The star wildcard denotes one or more words. This helps us to match any kind of input entered by user to be processed and give the output accordingly.
   E.g.:



```
<category>
   <pattern> A * is a *. </pattern>

   <template>
      When a <star index = "1"/> is not a <star index = "2"/>?
   </template>

</category>
```

Fig. 2.1: code snap showing use of star tag and start wildcard

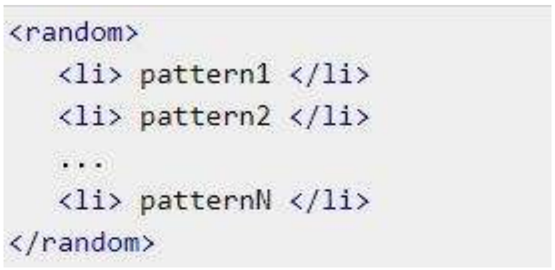
6. **srai tag:** This tag is used for multiple purpose. With the help of this tag, we can direct the same template as a response for different user inputs. This is useful in cases when a user can ask same question in different ways and formats and the answer is same for all those inputs. When the pattern is matched with the input then the srai tag is replaced with the corresponding template.
   E.g.:



```
<category>
   <pattern>DO YOU KNOW WHO * IS</pattern>
   <template>
      <srai>WHO IS <star/></srai>
   </template>
</category>
```

Fig. 2.2: code snap showing use of srai tag

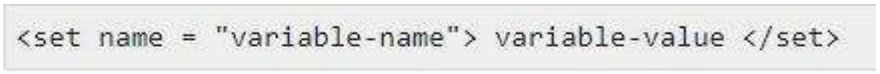
7. **random tag:** This tag is used define a list of responses for a given input and the interpreter will randomly pick one of those responses when the user input that particular pattern.
   E.g.:



```
<random>
   <li> pattern1 </li>
   <li> pattern2 </li>
   ...
   <li> patternN </li>
</random>
```

Fig. 2.3: code snap showing use of random tag

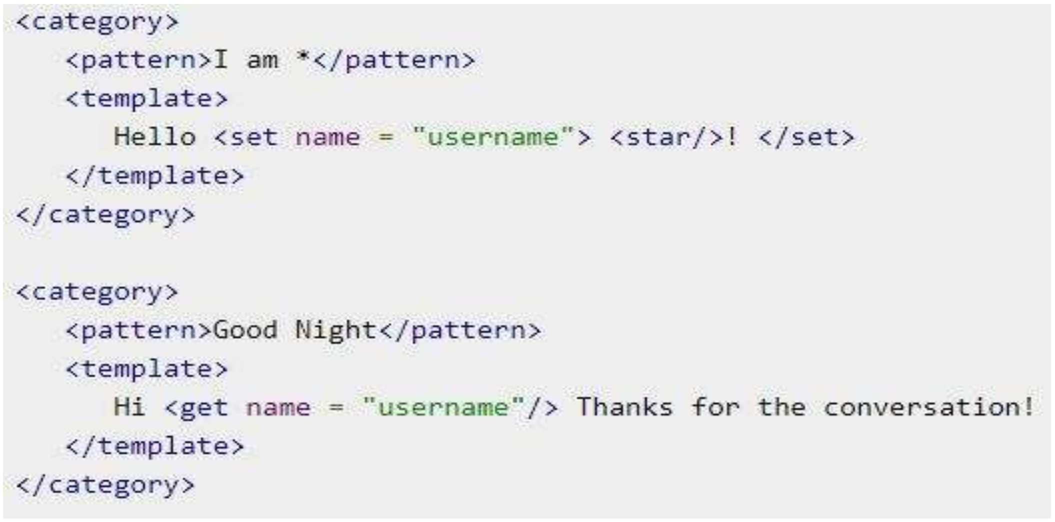
8. **set tag:** This tag is used to set a local variable known as predicates and it is specific to the client whom our chatbot is interacting. A variable can be predefined or can be created by programmer.
   E.g.:



```
<set name = "variable-name"> variable-value </set>
```

Fig. 2.4: code snap showing syntax of set tag

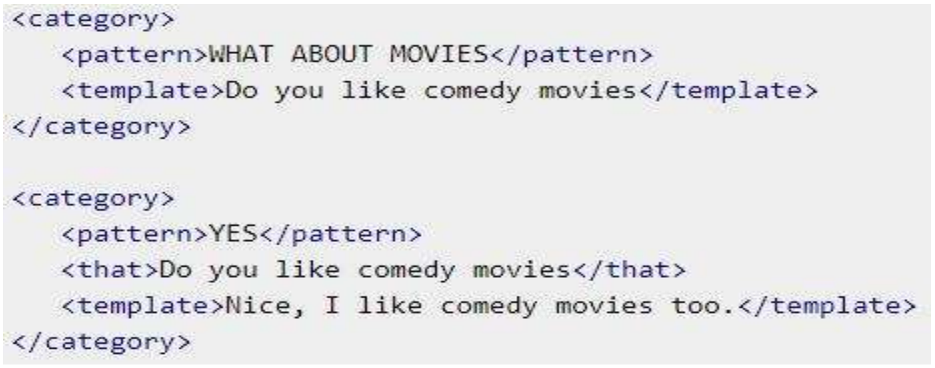
9. **get tag:** This tag is used to get the value of a local variable which is set previously while interacting with a specific client.
   E.g.:



```
<category>
   <pattern>I am *</pattern>
   <template>
      Hello <set name = "username"> <star/>! </set>
   </template>
</category>

<category>
   <pattern>Good Night</pattern>
   <template>
      Hi <get name = "username"/> Thanks for the conversation!
   </template>
</category>
```

Fig. 2.5: code snap showing use of set tag and get tag

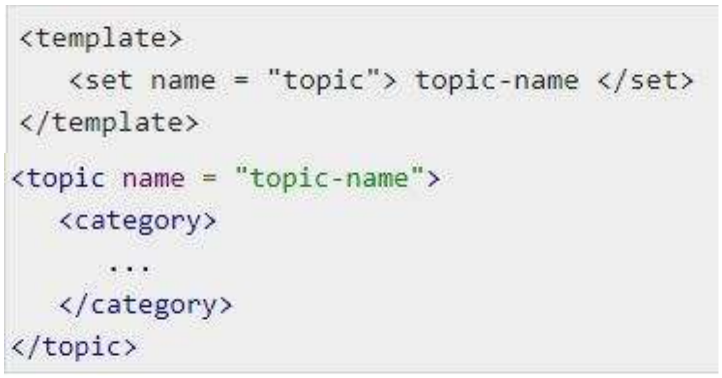
10. **that tag:** This tag is helpful when chatbot is interacting with a client with respect to a particular context. Inside a that tag, last sentence output given by the chatbot so that we can create the new template for responding according to the context. This tag is generally useful in case of interrogative conversation.
    E.g.:



```
<category>
   <pattern>WHAT ABOUT MOVIES</pattern>
   <template>Do you like comedy movies</template>
</category>

<category>
   <pattern>YES</pattern>
   <that>Do you like comedy movies</that>
   <template>Nice, I like comedy movies too.</template>
</category>
```

Fig. 2.6: code snap showing use of that tag

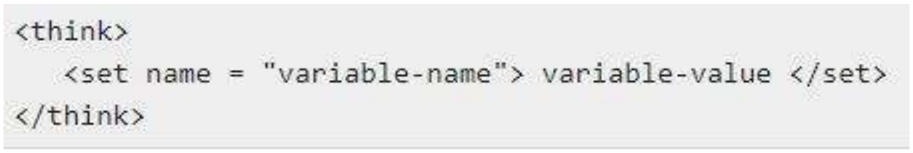
11. **topic tag:** This tag helps us to set context so that all the following conversion can be done with respect to that context. This tag is generally used in case of Yes or No conversation.
    E.g.:



```
<template>
   <set name = "topic"> topic-name </set>
</template>
<topic name = "topic-name">
   <category>
      ...
   </category>
</topic>
```

Fig. 2.7: code snap showing use of topic tag

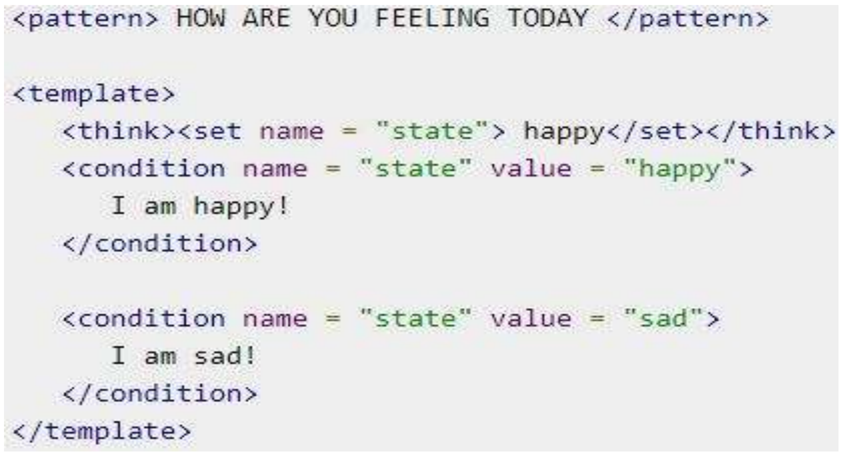
12. **think tag:** This tag is used to set variable without telling the client. Actually, in case of set tag, it also tells the user whatever the variables are set, but here it's not the case.
    E.g.:



```
<think>
   <set name = "variable-name"> variable-value </set>
</think>
```

Fig. 2.8: code snap showing syntax of think tag

13. **condition tag:** This tag works same as like a switch case in programming languages. It helps the chat bot to respond according to certain scenario depending on variable and its corresponding values.
    E.g.:



```
<pattern> HOW ARE YOU FEELING TODAY </pattern>

<template>
   <think><set name = "state"> happy</set></think>
   <condition name = "state" value = "happy">
      I am happy!
   </condition>

   <condition name = "state" value = "sad">
      I am sad!
   </condition>
</template>
```

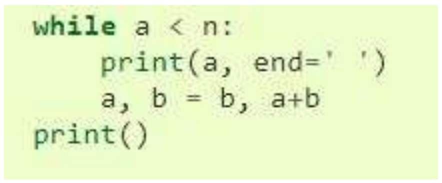
Fig. 2.9: code snap showing use of condition tag

2.3 Python:

Python is a very powerful object-oriented language. Python also has an interpreter for aiml language and it is very easy to use so that's why I have used python in this project.

In this project I have used python control structures like while loop, if else statements and some other things also.

I have use python here because working in python is very easy for project like this where you have to data processing because there are so many in-built functionalities that makes our task very easy and interesting.



```
while a < n:
    print(a, end=' ')
    a, b = b, a+b
print()
```

Fig. 2.10: code snap showing use while loop

2.4 python-aiml:

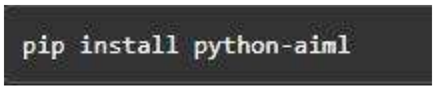
To use aiml with python we need a module called "python-aiml". While using python2, the module used is "aiml" and for python3, its is "python-aiml". I have used python3 in this project so I have used "python-aiml" module.

What "python-aiml" module do is, it implements an interpreter for aiml language.

Installing python-aiml:

Go to the root directory of your project and open the command prompt.

Now write the command:



```
pip install python-aiml
```

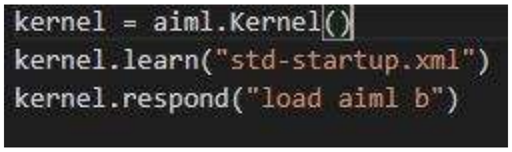
Fig. 2.11: command to install python-aiml module

Now we can import the module in our python file of our chatbot.

Kernel:

Kernel is something that we get with "python-aiml" module. This kernel is the main working body behind the use of aiml with python. This kernel makes the bot to learn different pattern and their respective responses as well as with the help of this the chat bot respond to different questions asked by client.

How to use kernel:



```
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
```

Fig. 2.12: code snap showing various kernel functions

First line in above snap shows the command to create a kernel and next two lines shows how kernel is used to learn aiml and xml files and how kernel respond to particular query or pattern respectively.

2.5 Flask:

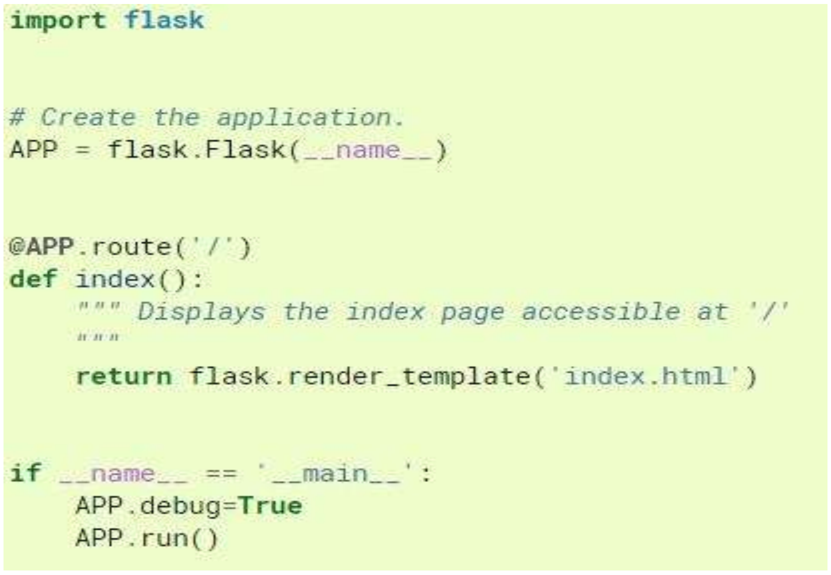
This is a web framework that provides us various tools and libraries to create web applications. It is written in python. This belongs to the category of micro frameworks which is a framework with very little dependencies on external libraries. This is a very light framework to work with.

Template Engine:

These are used to create application using flask. We create a **template folder** where various templates are kept like html documents and a **static folder** is created where various static files like images, CSS files and JS files are kept.

Create the application file which will be a python file.

Now in this file I have created an using flask and through that app I have accessed the content or query entered by the user and then that text query is fed to kernel which in turn provide response to that query according to the patterns that are learnt by it.



```
import flask


# Create the application.
APP = flask.Flask(__name__)


@APP.route('/')
def index():
    """ Displays the index page accessible at '/'
    """
    return flask.render_template('index.html')


if __name__ == '__main__':
    APP.debug=True
    APP.run()
```

Fig. 2.13: code snap showing creation of app using flask

render_template: This is the function used for rendering the content of a template like html file etc. This function also used to send data or update data of html file dynamically.

2.6 HTML:

HTML stands for Hyper Text Mark-up Language and is a standard language used for creating web pages. It is easy to learn and implement. There are number of elements in here and each element will tell how the web page will look at the end. We need a web browser to run html file.

HTML Elements:

1. **html Element:** This element defines the html file.
   E.g.: <html> content </html>
   Note: Never skip the end tag in html file.

2. **Head Element:** This element contains the data about the data i.e. metadata.
   This element is placed between html element and body element.
   E.g.: <head> content </head>

3. **Title Element:** This element contains the title of the document and is placed inside the head tag.
   E.g.: <title> content </title>

4. **Body Element:** This element contains the actual body of the html document inside it.
   E.g.: <body> content </body>

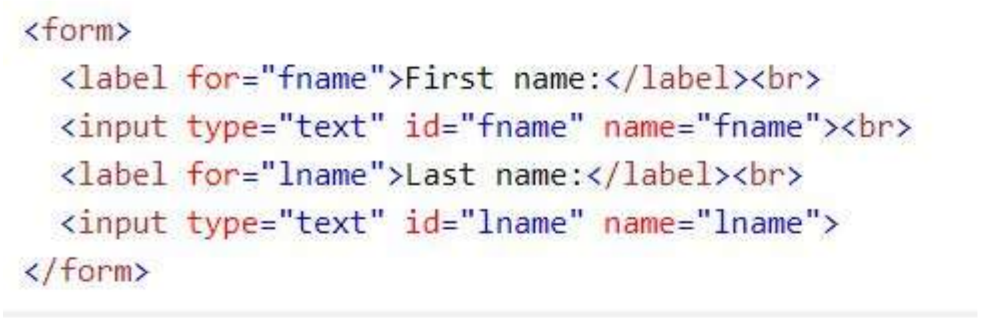
5. **Heading Element:** There are total 6 headings from h1 to h6. These elements are used to define the heading of the section inside the html page.
   E.g.: <h1> heading </h1>

6. **Paragraph Element:** This element defines a new paragraph inside the html web page.
   E.g.: <p> paragraph </p>

7. **Division Element**: This element defines a new division inside the html document.
   E.g.: <div> content </div>

8. **Style Element:** This element defines certain styles or apply certain styles to different elements inside the html document. This Element is present inside the head element.
   E.g.: <style> different styles </style>

9. **Form Element:** This form is used to define a form. This is used for getting the user input and then later on submit it to may be stored in database or may be to do certain computations and revert back the result.
   Syntax: <form> form elements </form>
   E.g.:



```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

Fig. 2.14: code snap showing code for a basic html form

2.7 CSS:

CSS stands for cascading style sheet. CSS defines the styles for html documents. This actually take the main part in defining how the html document will looks like. It can control how multiple pages will looks like at the same time.

CSS Syntax:

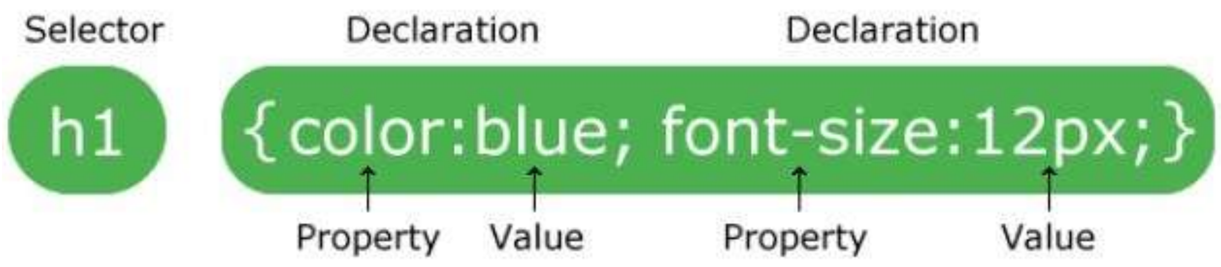
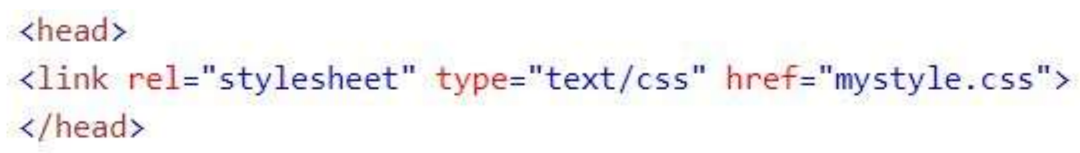


Fig. 2.15: figure showing CSS syntax

The selector element tells to which element the following style is going to affect.
The declaration section contains all the properties that are to be applied to that particular element.
In each declaration we have multiple properties and their respective values.
Multiple properties are separated by semi colon.
The declaration section is covered by curly braces.
In this project I have used Inline CSS, where properties are defined alongside the html element definition.

Types of CSS:

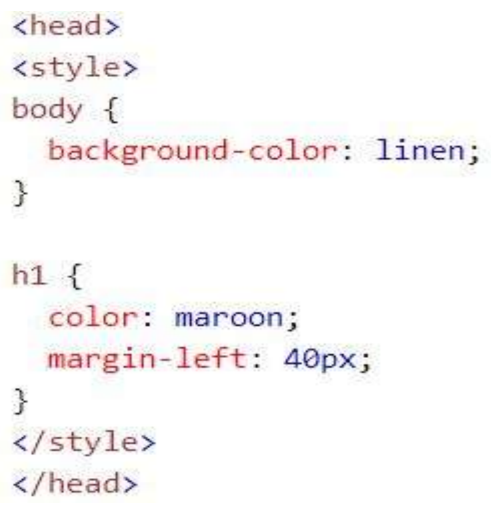
1. External CSS



```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Fig. 2.16: figure showing how to link external CSS file

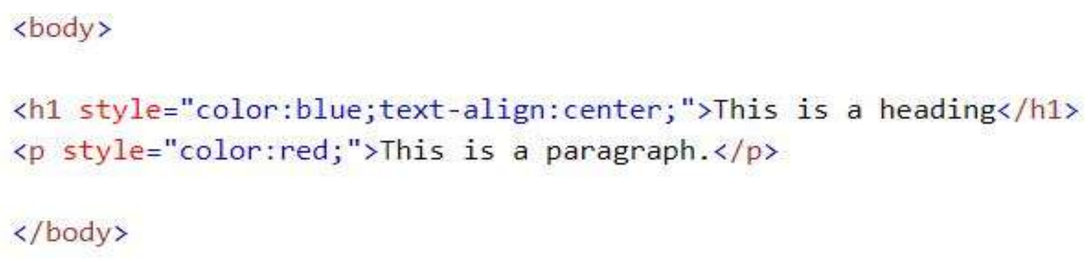
2. Internal CSS



```
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
```

Fig. 2.17: figure showing how use CSS internally

3. Inline CSS



```
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
```

Fig. 2.18: figure showing how use Inline CSS

Note:
While using Inline CSS, we are not able to take full benefits and advantages offered by CSS. So, Inline CSS is used sparingly.

CSS Properties:

1. **Background-color:**
   This property is used to set the background-color of the text inside a particular element.
2. **Color:**
   This property is used to set the text color of the particular element.
3. **Margin:**
   This property is used to keep one element far from other by putting some space between them after a defined area.
4. **Padding:**
   This property is used to create a space between the content inside a particular element but inside the border.
5. **Height:**
   This property is used to set the height of a particular element.
6. **Width:**
   This property is used to set width of a particular element of html.
7. **Text-align:**
   This property is used to set the text alignment in horizontal direction.
8. **Font-family:**
   This property of CSS is used to set the font family of the text.

2.8 gTTS:

It stands for “Google Text-to-Speech”. It is a python library which is used for interfacing with Google’s text to speech API.

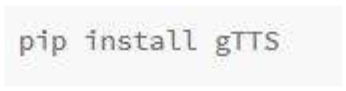
When we supply an input text to gTTS, it converts it into an audio file in the specified language.

gTTS requires internet connectivity for converting text to speech.

In this project I have also used gTTS as sometime output of a chatbot and nobody want to read it so the chatbot will read all its responses for you.

Installing gTTS:

Go to the root folder of your project and the type the command shown in figure 2.19.



```
pip install gTTS
```

Fig. 2.19: figure showing command to install gTTS

2.9 XML:

XML stands for Extensible Mark-up language. This is very similar to HTML. This language was mainly designed to store and transport data. It is a self-descriptive language. The difference between HTML and XML is that XML was developed to store and carry data while HTML was developed to display data.

XML document Structure:

XML documents have very interesting structure. It is like a tree which has a root, branches and leaves.

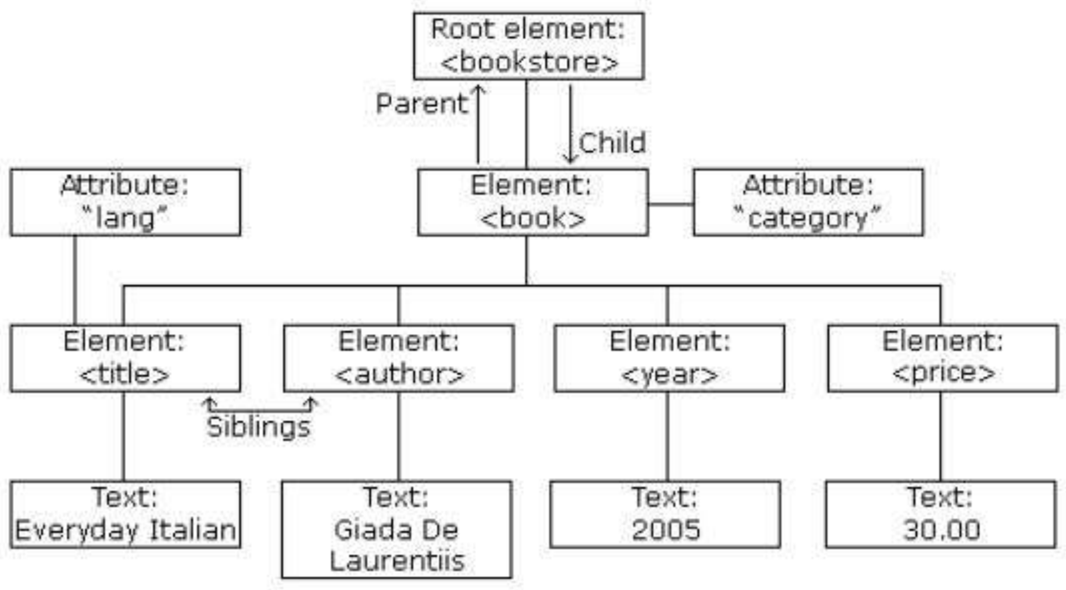
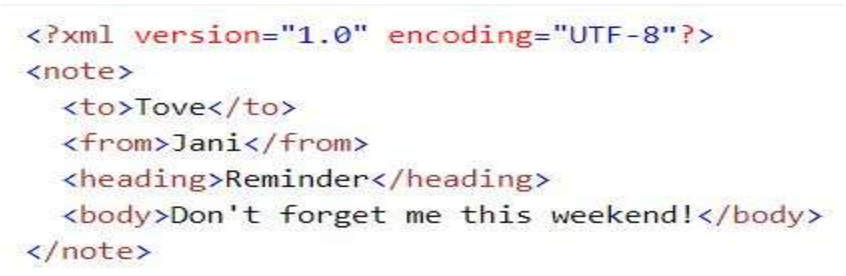


Fig. 2.20: Diagram showing XML document structure

XML Document Syntax:

The syntax of xml language is very simple and easy to learn. An xml document always have a root which act as apparent of all the elements.

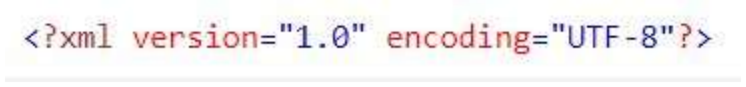


```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Fig. 2.21: code snap showing a sample of xml document

In the figure 2.21, <note> element is acting as a root element for all other elements present in the document.

XML Prolog:



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

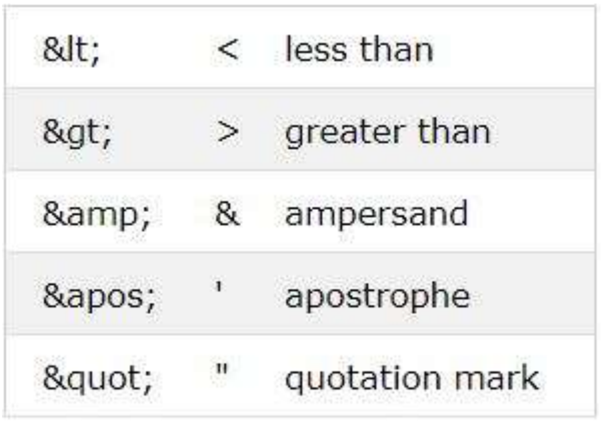
Fig. 2.22: code snap showing xml prolog

Though the xml prolog is option but still if it is present then it must be the first line of the xml document.

Note: All elements present in xml must have the closing tag as it is illegal to remove the closing tag. And, all the xml tags are case sensitive in nature.

In XML, like HTML we have elements who have attributes as name-value pairs.

Note: We cannot use symbols like <, >, &,',” directly in xml documents as they have special meaning there so instead, we have to use then like:



&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Fig. 2.22: figure showing the syntax for using different characters in xml document

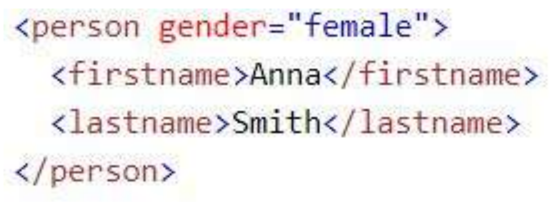
An xml element can contain, attributes, text, other element or all of these at the same time.

XML does not have any inbuilt specific elements.

Element name can start with a letter or an underscore and the element name cannot contain spaces.

Element name can contain letters, digits, hyphens and underscores and cannot start with letters “xml”.

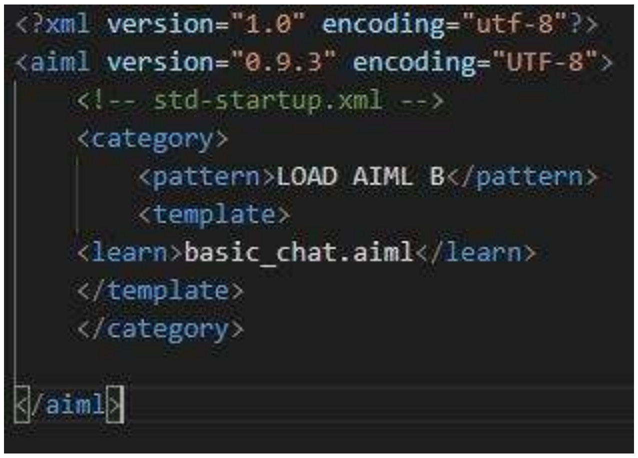
We can use attributes with xml elements like shown below:



```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

Fig. 2.23: figure showing the syntax for attributes

Though we know that AIML is another extension of xml but still we need to have basic Idea of xml for developing the chatbot because the starting point for any aiml chatbot is a standard xml start-up file called as "std-startup.xml". This file act as a main entry point for working of any aiml chatbot. This file will contain only a single pattern and a single response for that as shown in below code snapshot.



```
<?xml version="1.0" encoding="utf-8"?>
<aiml version="0.9.3" encoding="UTF-8">
    <!-- std-startup.xml -->
    <category>
        <pattern>LOAD AIML B</pattern>
        <template>
    <learn>basic_chat.aiml</learn>
    </template>
    </category>

</aiml>
```

Fig. 2.24: std-startup.xml file

As we can see that there is only single pattern and in response to that the aiml interpreter will command to learn the "basic_chat.aiml" file.

Chapter 03

System Development

The chatbot is developed using AIML which stands for Artificial Intelligence Mark-up Language. AIML use **pattern matching algorithm** to find the best match for entered query thus enables the chatbot to respond with best possible response. Various steps involved in development of this chatbot are shown in next section which is development plan.

3.1 Development plan:

Step-1: Create an HTML form to take in the input query of client.

Step-2: Setup the python-aiml module and create the basic chatbot python file.

Step-3: Creating flask app to import the input from html form.

Step-4: Personalise the aiml file to the specific topic to customise your chatbot.

Step-5: Now add the gTTS to the project to make it much easier to use.

3.2 Creating HTML Form:

The html form is created which will act as an interface for interacting with chatbot.

In this html form, we basically will have a block at the centre of the page where on the top left we will have JUIT logo. Below the logo we will see the name of the chatbot and below that we will have the input bar to enter the query. Below the input query bar, we have the search button and below the search button we will have the response section where response will be printed or reverted to the client.

I have used various CSS properties to style that html web page and I have used he inline CSS in developing this webpage.

Here, I basically have created two divisions, one for the background and other for the centred box where the form will be given to enter the query and display responses.

The background is of dark purple color and the centre box is of white color.

The place holder for query bar is set to “Query” and the search button is of light purple color.

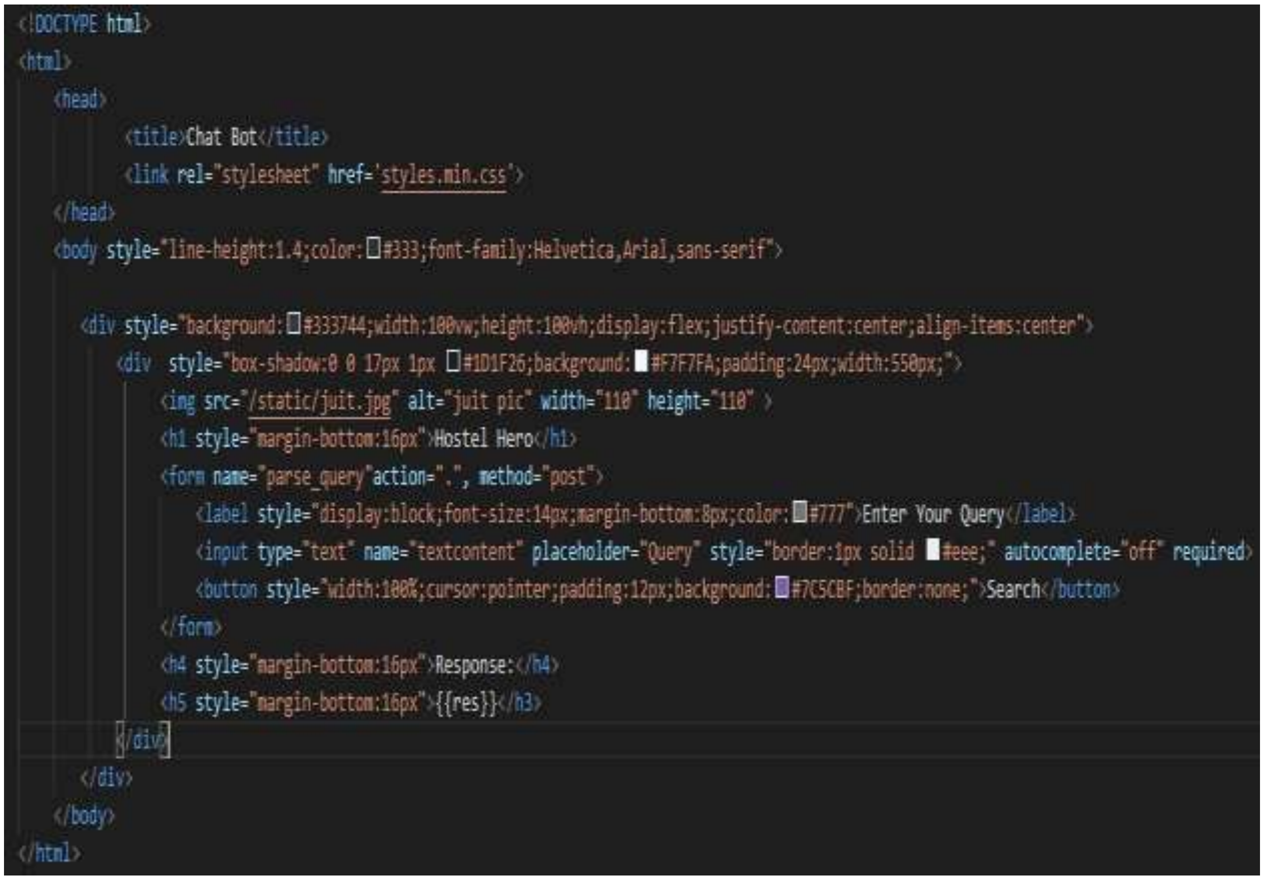
Both the input bar and the search button have the same width in order to bring the uniformity to the form.

Autocomplete property is set to off and the user can ask query in English language and in any form he/ she likes.

Centre box is made a little wider as there are some large responses to certain queries so those responses can fill I the space appropriately without affecting much of the uniformity of the web page.

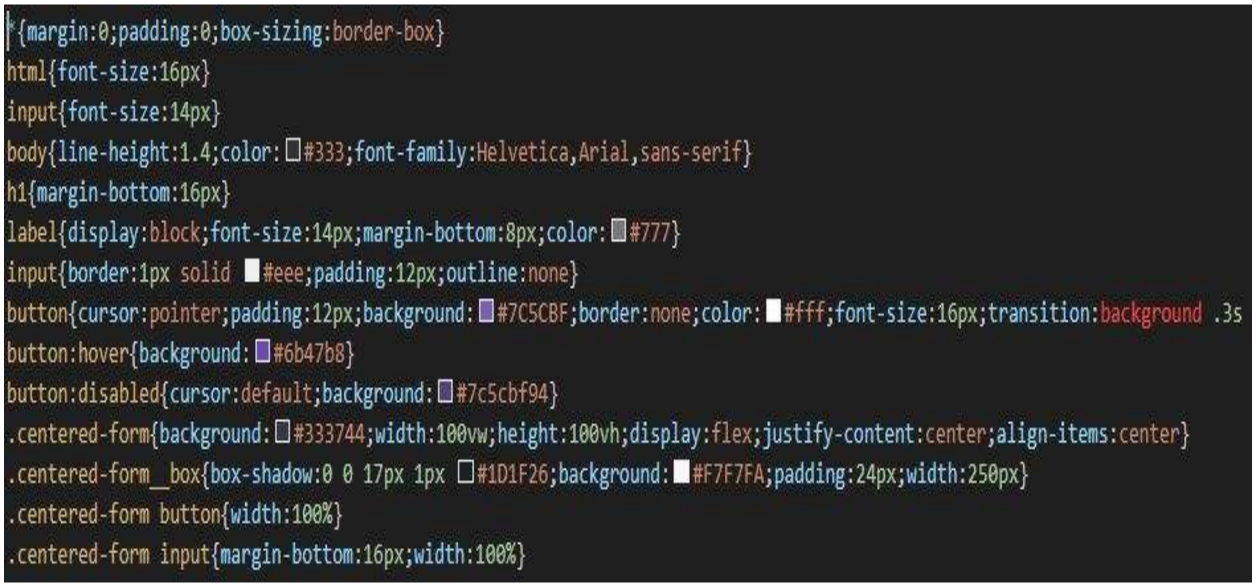
The name of the input field is set to “textcontent” as it will also be used in the flask app to render the input.

Some of the CSS properties used are also imported from external CSS file which is placed in the static folder which is present in the root directory of our project.



```html
<!DOCTYPE html>
<html>
    <head>
            <title>Chat Bot</title>
            <link rel="stylesheet" href='styles.min.css'>
    </head>
    <body style="line-height:1.4;color:#333;font-family:Helvetica,Arial,sans-serif">

      <div style="background:#333744;width:100vw;height:100vh;display:flex;justify-content:center;align-items:center">
          <div  style="box-shadow:0 0 17px 1px #1D1F26;background:#F7F7FA;padding:24px;width:550px;">
              <img src="/static/juit.jpg" alt="juit pic" width="110" height="110" >
              <h1 style="margin-bottom:16px">Hostel Hero</h1>
              <form name="parse_query"action=".", method="post">
                  <label style="display:block;font-size:14px;margin-bottom:8px;color:#777">Enter Your Query</label>
                  <input type="text" name="textcontent" placeholder="Query" style="border:1px solid #eee;" autocomplete="off" required>
                  <button style="width:100%;cursor:pointer;padding:12px;background:#7C5CBF;border:none;">Search</button>
              </form>
              <h4 style="margin-bottom:16px">Response:</h4>
              <h5 style="margin-bottom:16px">{{res}}</h3>
          </div>
      </div>
    </body>
</html>
```

Fig. 3.1: code snap showing the code of HTML form



```
*{margin:0;padding:0;box-sizing:border-box}
html{font-size:16px}
input{font-size:14px}
body{line-height:1.4;color:#333;font-family:Helvetica,Arial,sans-serif}
h1{margin-bottom:16px}
label{display:block;font-size:14px;margin-bottom:8px;color:#777}
input{border:1px solid #eee;padding:12px;outline:none}
button{cursor:pointer;padding:12px;background:#7C5CBF;border:none;color:#fff;font-size:16px;transition:background .3s
button:hover{background:#6b47b8}
button:disabled{cursor:default;background:#7c5cbf94}
.centered-form{background:#333744;width:100vw;height:100vh;display:flex;justify-content:center;align-items:center}
.centered-form__box{box-shadow:0 0 17px 1px #1D1F26;background:#F7F7FA;padding:24px;width:250px}
.centered-form button{width:100%}
.centered-form input{margin-bottom:16px;width:100%}
```

Fig. 3.2: code snap showing various CSS properties for styling different elements

3.3 Chatbot Python File:

In this section, I have used python language to implement the aiml interpreter. This is because the python language has a very clean grammar and the syntax of the language is also very-very easy to learn as well as implement. Also, the language has so many library and inbuilt functions as well as API the it makes the development in this language a very easy task to do. That is why I have chosen python as a language to implement the aiml interpreter.
Now for this, first of all I have installed "python-aiml" module. "python-aiml" module is a module which implements the aiml interpreter for python language. "python-aiml" is the module for implementing the aiml interpreter while using the python3 version of python. If we are using python2 then the module we require is "aiml" not "python-aiml".

First go to the root directory of project and open the command prompt and type the following command:

```
$ pip install python-aiml
```

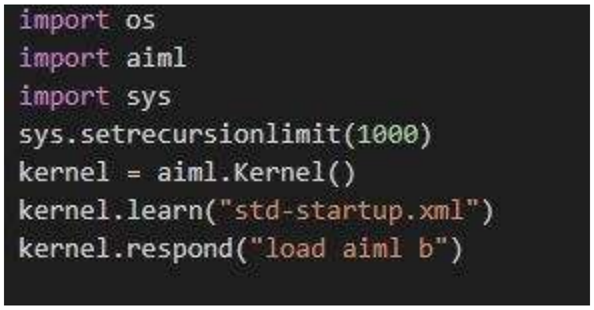
After it gets installed completely, we can import the aiml module to our python file.
Now, we need to create the kernel which is a functionality provided by the aiml module.
This kernel will do al the work starting from learning file to responding to a particular query of the client.
We can create the kernel as follow:

```
kernel= aiml.Kernel()
```

Kernel respond to a query as follow:

Response=kernel.respond(query)

Now what is happening is that kernel is taking in the query and use the pattern matching algorithms to match the asked query to a particular pattern that it has learnt from aiml files. We have used a “while loop” which will keep on listening to the queries of the clients.



```
import os
import aiml
import sys
sys.setrecursionlimit(1000)
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
```

Fig. 3.3: code snap showing the kernel functionalities and use

3.4 Integrating Flask App in chatbot python file:

Now, we need to create the flask app through which we can link our html form to the python file means we can send data to/from html form from/to python file. This is the main requirement as we take the input query entered through the html form but we need it to supply to our kernel and interpreter so that it can respond to it. Similarly, we also want the response sent by the kernel to be presented to user and that is presented through html form. This all need for the flow of data from here to there gave rise to the need of flask.

I have also tried doing this using CGI but that was very complex and there were so many things and problems because of which it would have taken so much time to complete this project so that’s why I shifted to flask for doing this task. And using flask has made this work so much simpler and easier and it was a great relief.

Installing Flask:

To install Flask, type the following command in your command prompt in root directory of your project:

```
$ pip install Flask
```

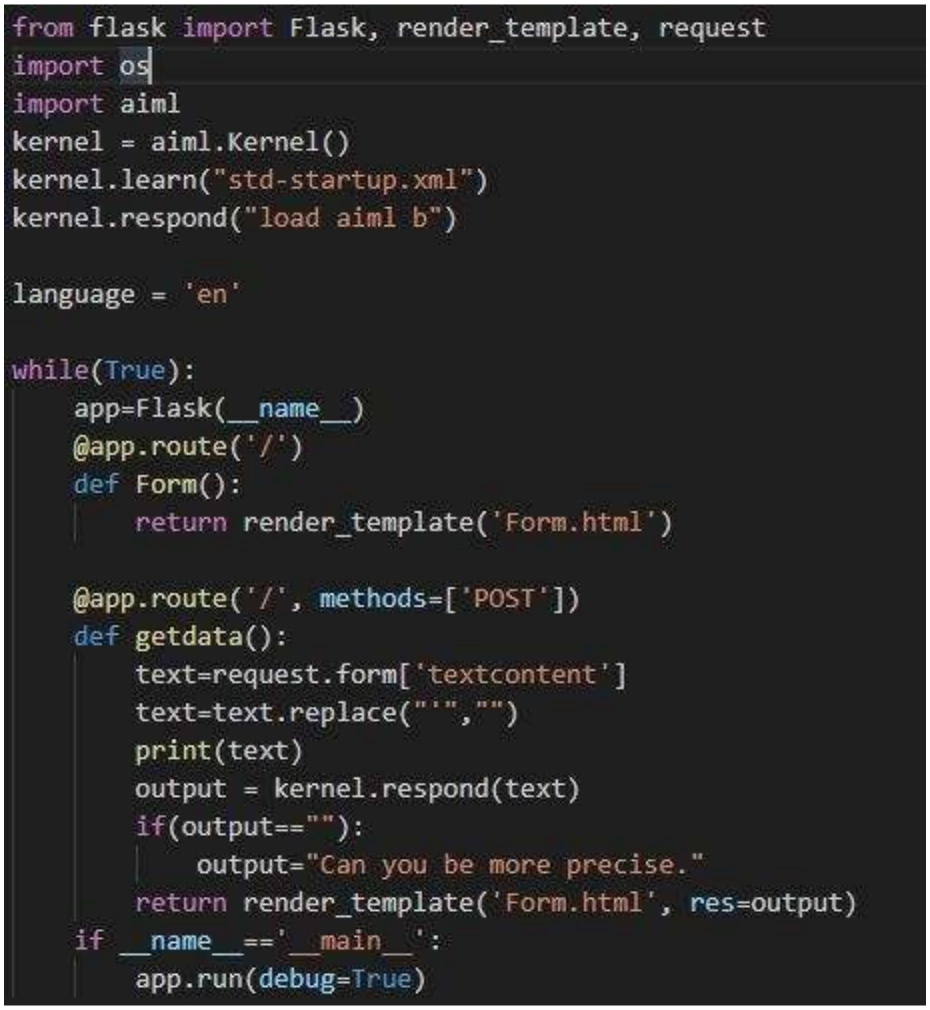
Now from flask, we need three things to import:

1. **Flask:** To create the flask app.
2. **render_template:** To render the templates i.e. html form. It is used to get the content from different input fields of the form.
3. **request:** This is used to extract content from different elements of the rendered template.

Note: In case of rendering the template, we need to specify the route to the app so that it can render the particular temple by implementing a function and this is done using "render_template".

After rendering a template, we need to make another function which will extract the data from different elements of the rendered template and this is done using "request".

Now, our chatbot file will looks like:



```
from flask import Flask, render_template, request
import os
import aiml
kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")

language = 'en'

while(True):
    app=Flask(__name__)
    @app.route('/')
    def Form():
        return render_template('Form.html')

    @app.route('/', methods=['POST'])
    def getdata():
        text=request.form['textcontent']
        text=text.replace("'","")
        print(text)
        output = kernel.respond(text)
        if(output==""):
            output="Can you be more precise."
        return render_template('Form.html', res=output)
    if __name__=='__main__':
        app.run(debug=True)
```

Fig. 3.4: code snap showing chatbot file after integrating flask app

After integrating the above flask app to the python file of chatbot, the prototype of chatbot is ready but it doesn't have any knowledge to interact with client. It is like a new born baby, who doesn't know how to talk and walk but can learn very easily to do so if we supply it with sufficient knowledge. A chatbot learns from the previously stored information and in the next section we will provide it with sufficient information so that it can learn how to interact and respond according to the question asked and this all is done by providing it with aiml files which act as a database for the chatbot from where it can use the information to do its task efficiently.

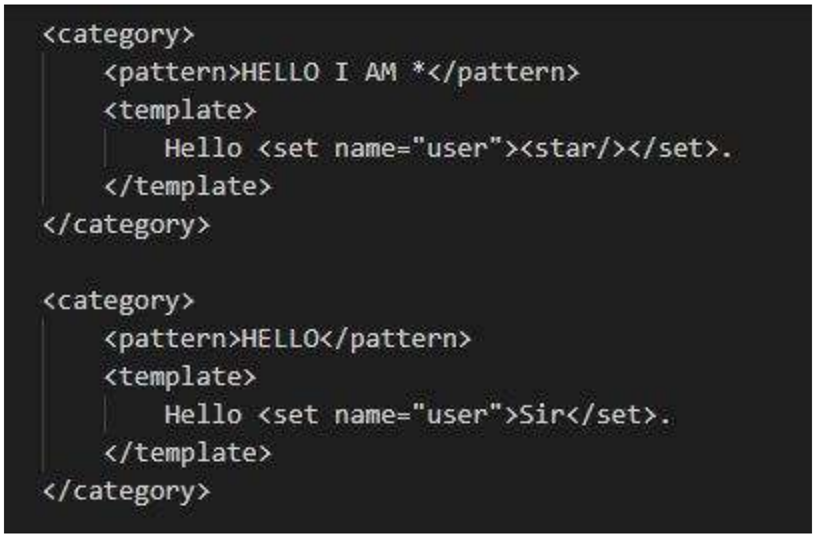
3.5 Creating personized AIML file:

The aiml files act as a database for the chatbot and the chatbot use the information stored in these files to do the conversation with client. These are the aiml files which helps us to make our chatbot to be centred to a specific organisation or purpose. These aiml files determines the nature of chatbot e.g. whether it is for a software company or a law firm etc.

We can personalize our chatbot and make it field specific with the help of these aiml files.

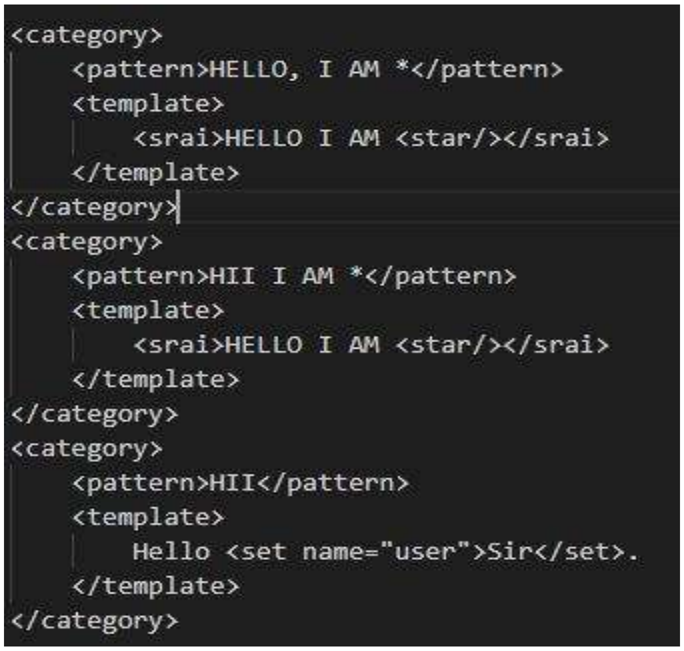
In this project as we are making our chatbot to be responsible for any kind of query regarding the Hostels of JUIT, so the aiml file here contain all the information regarding the JUIT Hostels.

These files here include all patterns in which any query can be generated and the corresponding responses regarding the hostel of JUIT.



```
<category>
    <pattern>HELLO I AM *</pattern>
    <template>
        Hello <set name="user"><star/></set>.
    </template>
</category>

<category>
    <pattern>HELLO</pattern>
    <template>
        Hello <set name="user">Sir</set>.
    </template>
</category>
```

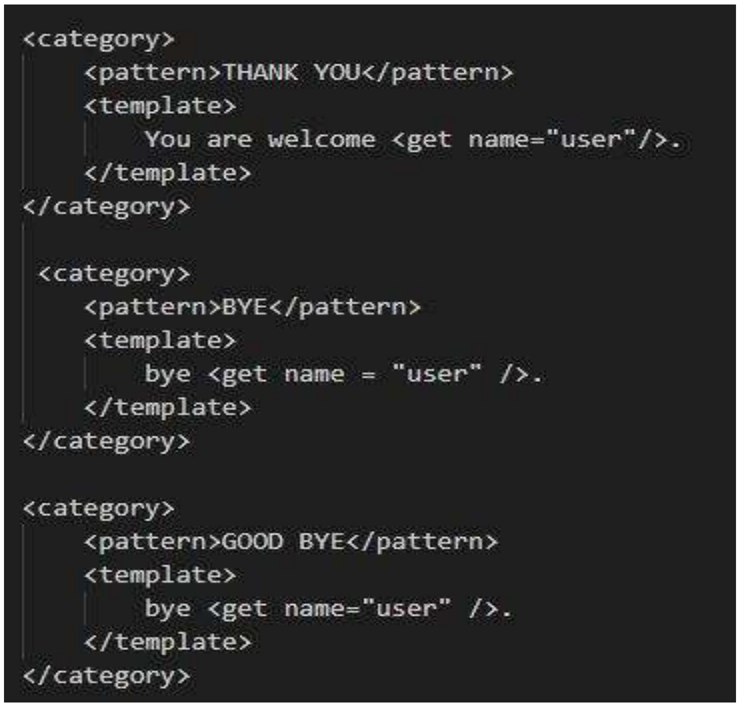
Fig. 3.5: code snap showing aiml code for starting conversation part-1



```
<category>
    <pattern>HELLO, I AM *</pattern>
    <template>
        <srai>HELLO I AM <star/></srai>
    </template>
</category>
<category>
    <pattern>HII I AM *</pattern>
    <template>
        <srai>HELLO I AM <star/></srai>
    </template>
</category>
<category>
    <pattern>HII</pattern>
    <template>
        Hello <set name="user">Sir</set>.
    </template>
</category>
```

Fig. 3.6: code snap showing aiml code for starting conversation part-2

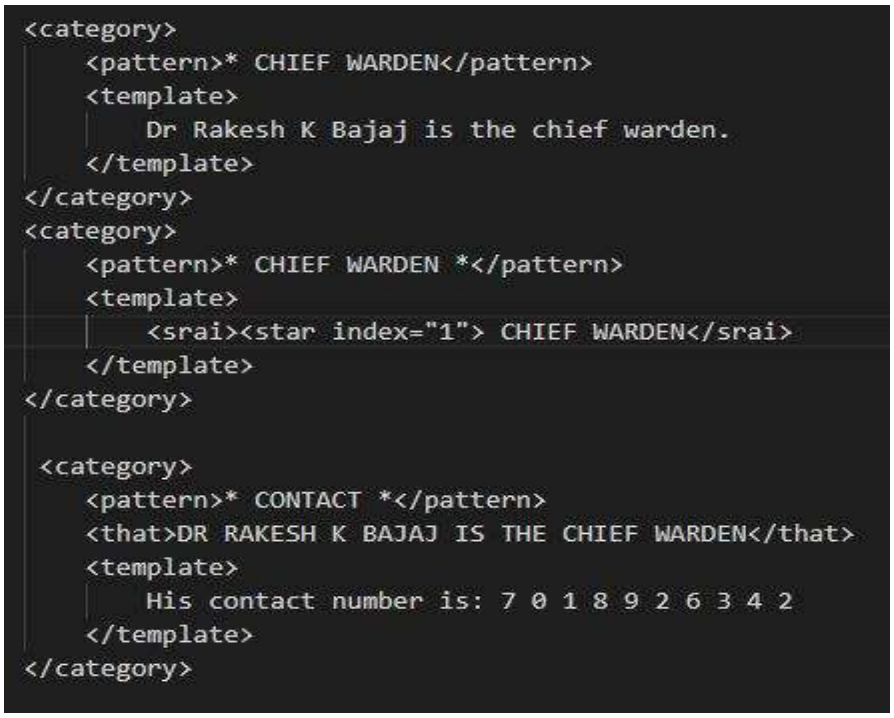
Figure 3.5 and 3.6 shows the aiml code for the beginning of conversation with chatbot and here I have used <srai> tag also. Now, following figures will show the aiml code for the end of the conversation.



```
<category>
    <pattern>THANK YOU</pattern>
    <template>
        You are welcome <get name="user"/>.
    </template>
</category>

 <category>
    <pattern>BYE</pattern>
    <template>
        bye <get name = "user" />.
    </template>
</category>

<category>
    <pattern>GOOD BYE</pattern>
    <template>
        bye <get name="user" />.
    </template>
</category>
```

Fig. 3.7: code snap showing aiml code for end conversation

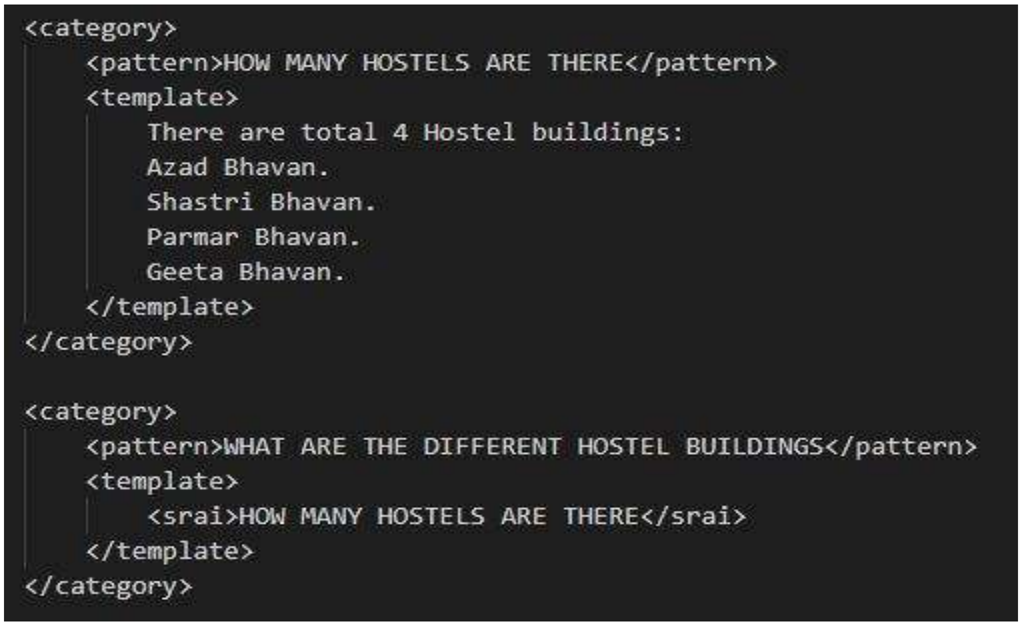
Now, Head of all the hostels is chief warden. The following aiml code shows patterns and templates related to chief warden.



```
<category>
    <pattern>* CHIEF WARDEN</pattern>
    <template>
        Dr Rakesh K Bajaj is the chief warden.
    </template>
</category>
<category>
    <pattern>* CHIEF WARDEN *</pattern>
    <template>
        <srai><star index="1"> CHIEF WARDEN</srai>
    </template>
</category>

 <category>
    <pattern>* CONTACT *</pattern>
    <that>DR RAKESH K BAJAJ IS THE CHIEF WARDEN</that>
    <template>
        His contact number is: 7 0 1 8 9 2 6 3 4 2
    </template>
</category>
```

Fig. 3.8: code snap showing aiml code for query related to chief warden

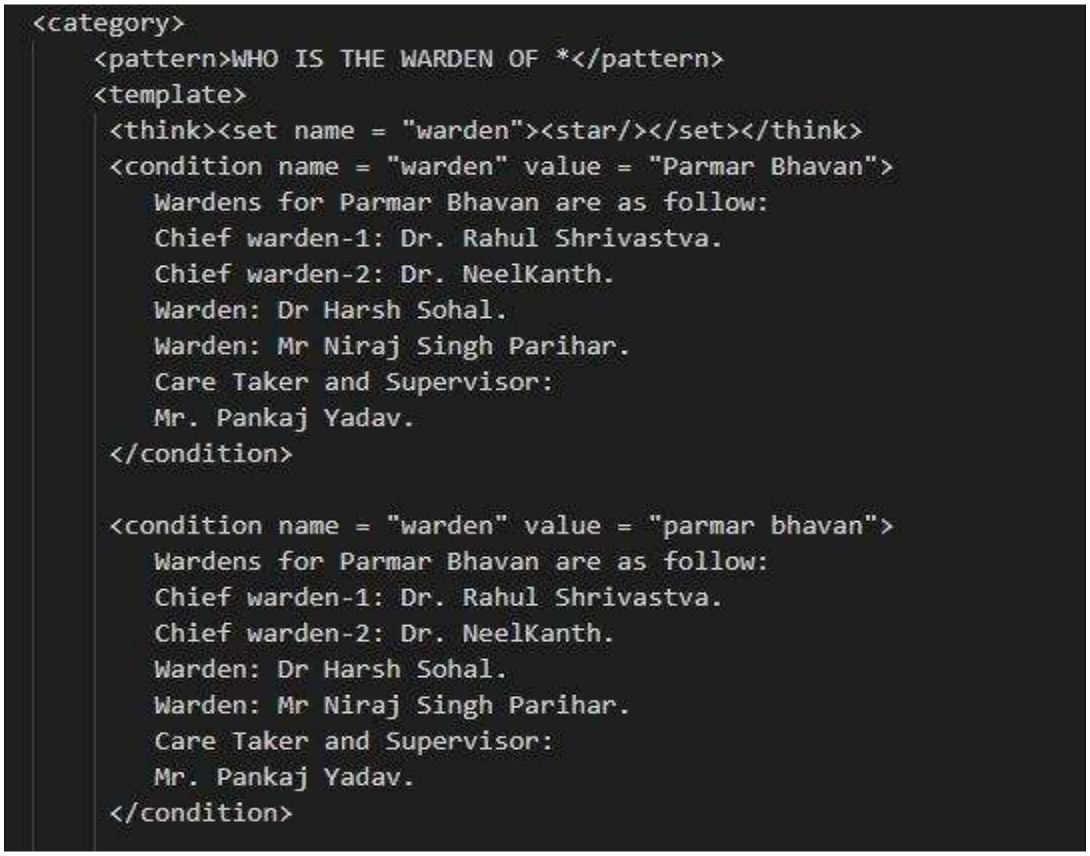
Now, following figures will show the aiml code for the query related to Hostel Buildings of JUIT.



```
<category>
    <pattern>HOW MANY HOSTELS ARE THERE</pattern>
    <template>
        There are total 4 Hostel buildings:
        Azad Bhavan.
        Shastri Bhavan.
        Parmar Bhavan.
        Geeta Bhavan.
    </template>
</category>

<category>
    <pattern>WHAT ARE THE DIFFERENT HOSTEL BUILDINGS</pattern>
    <template>
        <srai>HOW MANY HOSTELS ARE THERE</srai>
    </template>
</category>
```

Fig. 3.9: code snap showing aiml code for knowing the hostel buildings

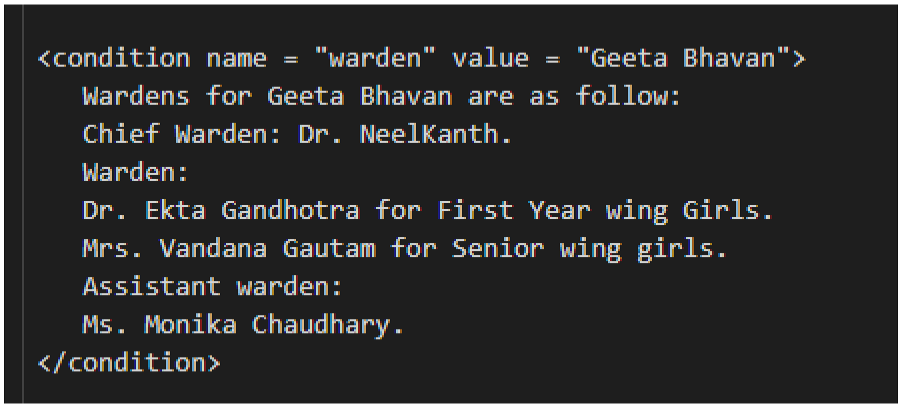
After Knowing about the hostel names, one can ask for the respective wardens of each hostel. The following code snaps shows the related code for this type of queries.



```
<category>
    <pattern>WHO IS THE WARDEN OF *</pattern>
    <template>
     <think><set name = "warden"><star/></set></think>
     <condition name = "warden" value = "Parmar Bhavan">
        Wardens for Parmar Bhavan are as follow:
        Chief warden-1: Dr. Rahul Shrivastva.
        Chief warden-2: Dr. NeelKanth.
        Warden: Dr Harsh Sohal.
        Warden: Mr Niraj Singh Parihar.
        Care Taker and Supervisor:
        Mr. Pankaj Yadav.
     </condition>

     <condition name = "warden" value = "parmar bhavan">
        Wardens for Parmar Bhavan are as follow:
        Chief warden-1: Dr. Rahul Shrivastva.
        Chief warden-2: Dr. NeelKanth.
        Warden: Dr Harsh Sohal.
        Warden: Mr Niraj Singh Parihar.
        Care Taker and Supervisor:
        Mr. Pankaj Yadav.
     </condition>
```

Fig. 3.10: code snap showing aiml code for knowing the warden of Parmar Bhavan

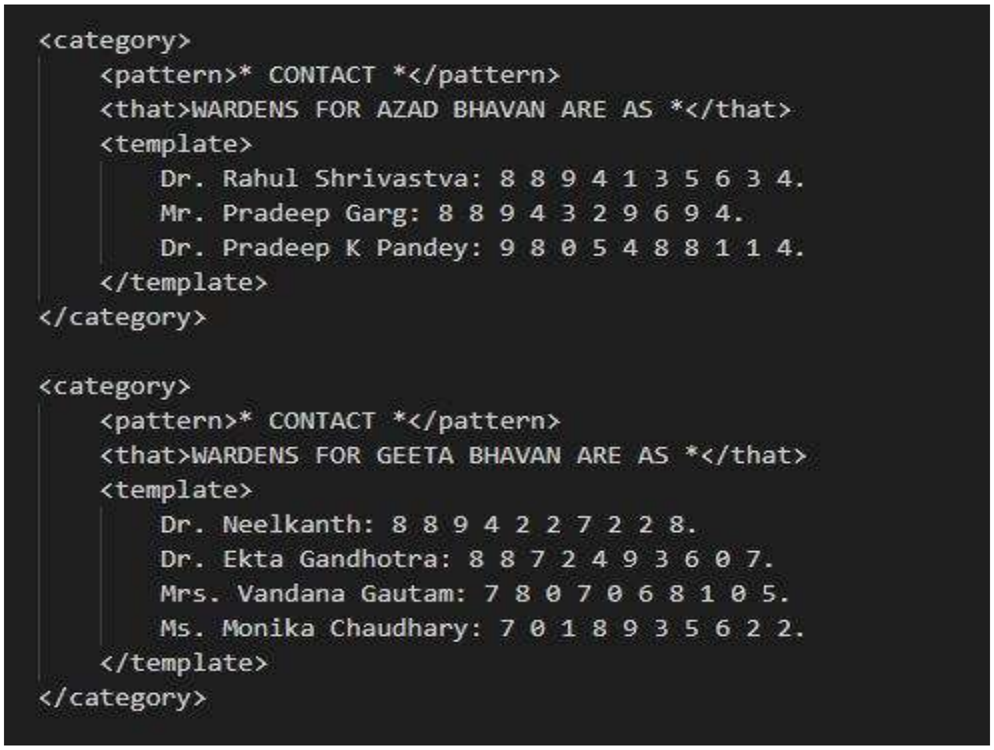
Parallelly, the next figure show query about warden of Geeta Bhavan.



```
<condition name = "warden" value = "Geeta Bhavan">
   Wardens for Geeta Bhavan are as follow:
   Chief Warden: Dr. NeelKanth.
   Warden:
   Dr. Ekta Gandhotra for First Year wing Girls.
   Mrs. Vandana Gautam for Senior wing girls.
   Assistant warden:
   Ms. Monika Chaudhary.
</condition>
```

Fig. 3.11: code snap showing aiml code for knowing the warden of Geeta Bhavan

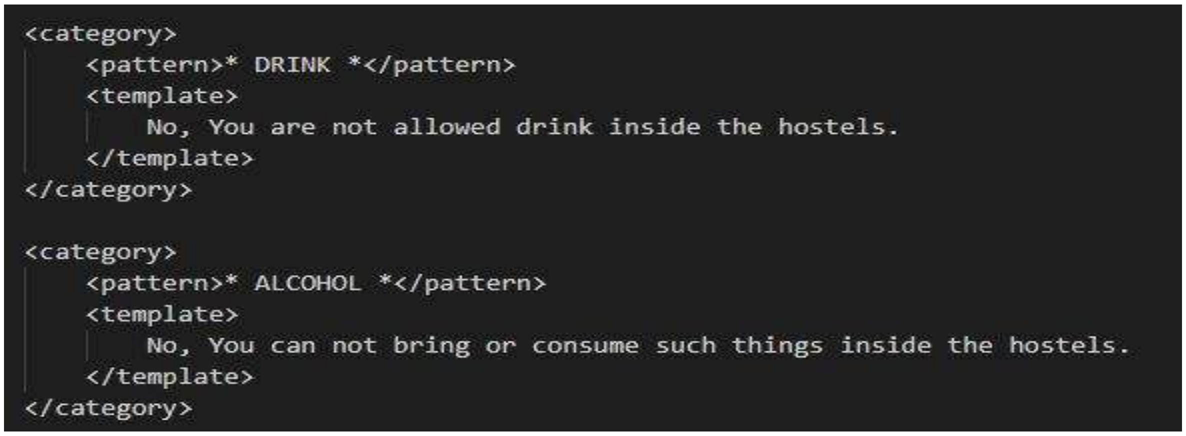
Now, after knowing about the hostel warden one may ask for the contact number of the warden. The following code snap shows the aiml code for this case.



```
<category>
    <pattern>* CONTACT *</pattern>
    <that>WARDENS FOR AZAD BHAVAN ARE AS *</that>
    <template>
        Dr. Rahul Shrivastva: 8 8 9 4 1 3 5 6 3 4.
        Mr. Pradeep Garg: 8 8 9 4 3 2 9 6 9 4.
        Dr. Pradeep K Pandey: 9 8 0 5 4 8 8 1 1 4.
    </template>
</category>

<category>
    <pattern>* CONTACT *</pattern>
    <that>WARDENS FOR GEETA BHAVAN ARE AS *</that>
    <template>
        Dr. Neelkanth: 8 8 9 4 2 2 7 2 2 8.
        Dr. Ekta Gandhotra: 8 8 7 2 4 9 3 6 0 7.
        Mrs. Vandana Gautam: 7 8 0 7 0 6 8 1 0 5.
        Ms. Monika Chaudhary: 7 0 1 8 9 3 5 6 2 2.
    </template>
</category>
```

Fig. 3.12: code snap showing aiml code for contact of the warden of Geeta Bhavan and Azad Bhavan

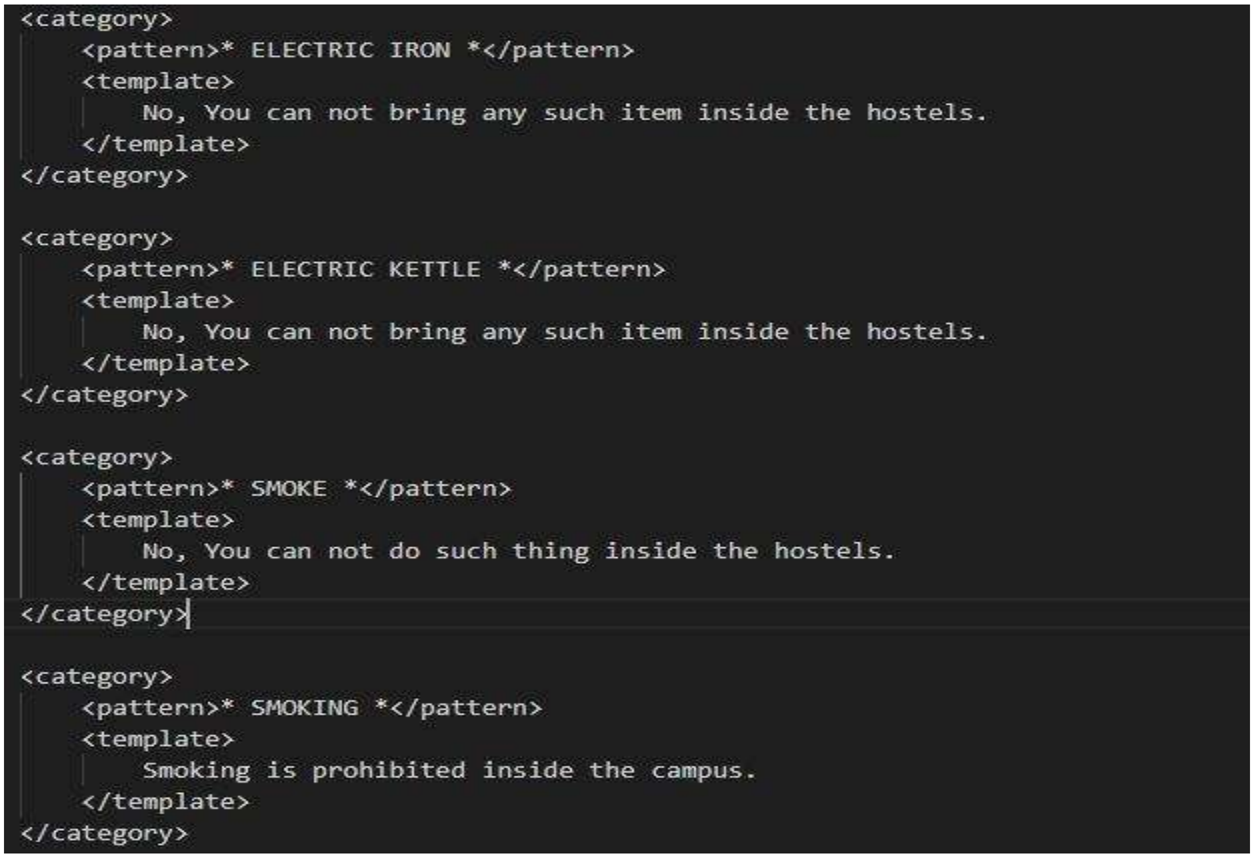
In the figure 3.12, we can also see the use of that tag which used to communicate with respect to a context. In the next figure, the code snap shows the code for query related to alcohol.



```
<category>
    <pattern>* DRINK *</pattern>
    <template>
        No, You are not allowed drink inside the hostels.
    </template>
</category>

<category>
    <pattern>* ALCOHOL *</pattern>
    <template>
        No, You can not bring or consume such things inside the hostels.
    </template>
</category>
```

Fig. 3.13: code snap showing aiml code for query related to alcohol and drinking

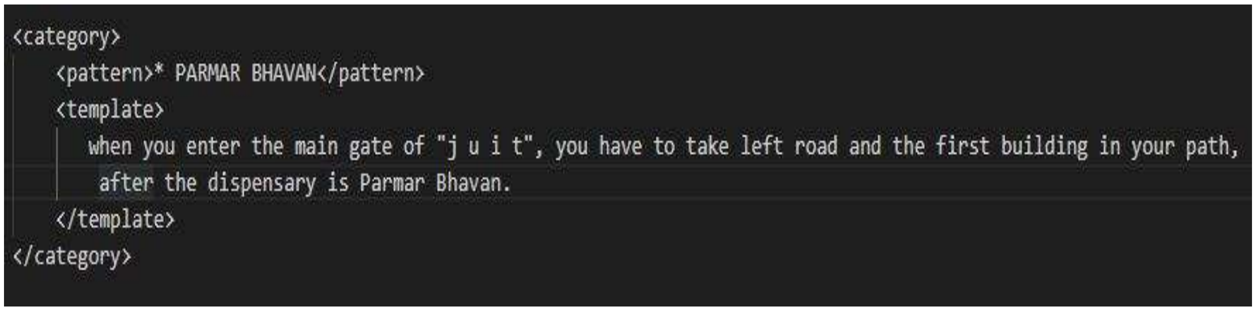
The next code snap shows the code for some other related queries like smoking, electrical appliances etc.



```
<category>
    <pattern>* ELECTRIC IRON *</pattern>
    <template>
        No, You can not bring any such item inside the hostels.
    </template>
</category>

<category>
    <pattern>* ELECTRIC KETTLE *</pattern>
    <template>
        No, You can not bring any such item inside the hostels.
    </template>
</category>

<category>
    <pattern>* SMOKE *</pattern>
    <template>
        No, You can not do such thing inside the hostels.
    </template>
</category>

<category>
    <pattern>* SMOKING *</pattern>
    <template>
        Smoking is prohibited inside the campus.
    </template>
</category>
```

Fig. 3.14: code snap showing aiml code for query related to smoking and appliances etc.

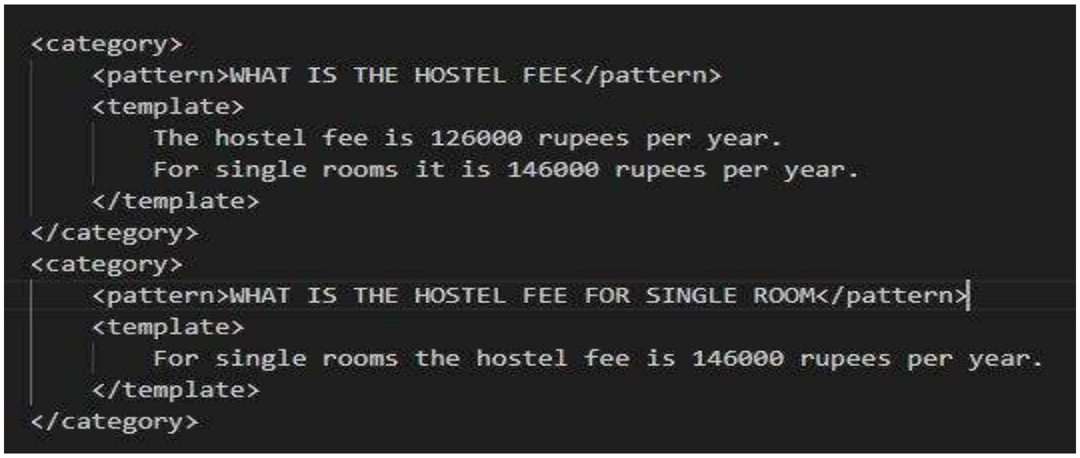
The next code snap shows the code for query related to reaching Parmar Bhavan.



```
<category>
    <pattern>* PARMAR BHAVAN</pattern>
    <template>
      when you enter the main gate of "j u i t", you have to take left road and the first building in your path,
       after the dispensary is Parmar Bhavan.
    </template>
</category>
```

Fig. 3.15: code snap showing aiml code for query related to reaching Parmar Bhavan.

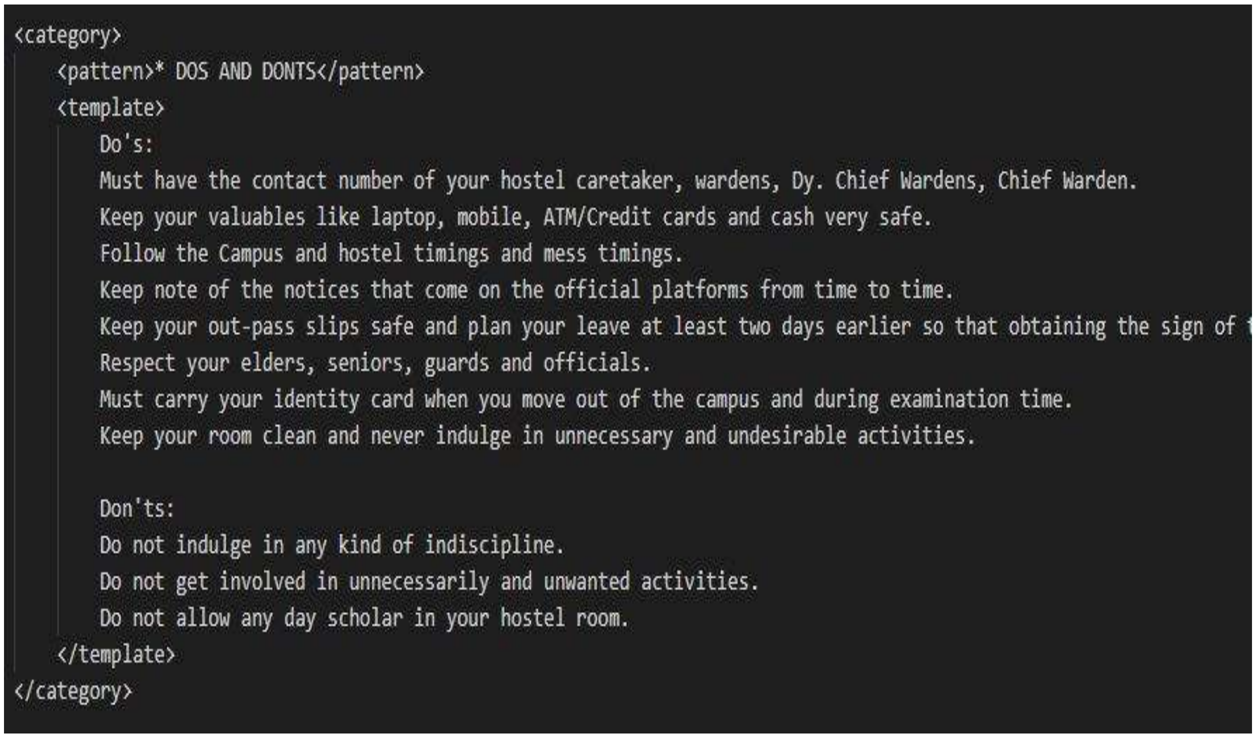
The next figure shows the code snap for queries related to hostel fee.



```
<category>
    <pattern>WHAT IS THE HOSTEL FEE</pattern>
    <template>
        The hostel fee is 126000 rupees per year.
        For single rooms it is 146000 rupees per year.
    </template>
</category>
<category>
    <pattern>WHAT IS THE HOSTEL FEE FOR SINGLE ROOM</pattern>
    <template>
        For single rooms the hostel fee is 146000 rupees per year.
    </template>
</category>
```

Fig. 3.16: code snap showing aiml code for query related hostel fee.

Same as shown in the previous figures, I have designed the chatbot to give response related to any matter mentioned above for all hostels and now at the end the query related what are the do’s and don’ts of the hostels is shown in the below figure.



```
<category>
    <pattern>* DOS AND DONTS</pattern>
    <template>
        Do's:
        Must have the contact number of your hostel caretaker, wardens, Dy. Chief Wardens, Chief Warden.
        Keep your valuables like laptop, mobile, ATM/Credit cards and cash very safe.
        Follow the Campus and hostel timings and mess timings.
        Keep note of the notices that come on the official platforms from time to time.
        Keep your out-pass slips safe and plan your leave at least two days earlier so that obtaining the sign of
        Respect your elders, seniors, guards and officials.
        Must carry your identity card when you move out of the campus and during examination time.
        Keep your room clean and never indulge in unnecessary and undesirable activities.

        Don'ts:
        Do not indulge in any kind of indiscipline.
        Do not get involved in unnecessarily and unwanted activities.
        Do not allow any day scholar in your hostel room.
    </template>
</category>
```

Fig. 3.17: code snap showing aiml code for query related to do’s and don’ts.

3.6 gTTS: The Bonus

As we know that gTTS is a python library which is used to convert text to speech by using internet.

It is a bonus as some time we don't want to read the results so to help with this the chatbot is capable of reding the output for you.

After installing gTTS library, we can import it to our python file.

Now the gTTS function take three arguments:

1. **text:** which is to be spoken.
2. **Language:** in which language to be spoken which I have set to English "en".
3. **Slow:** whether to read it slow which I have set to false.

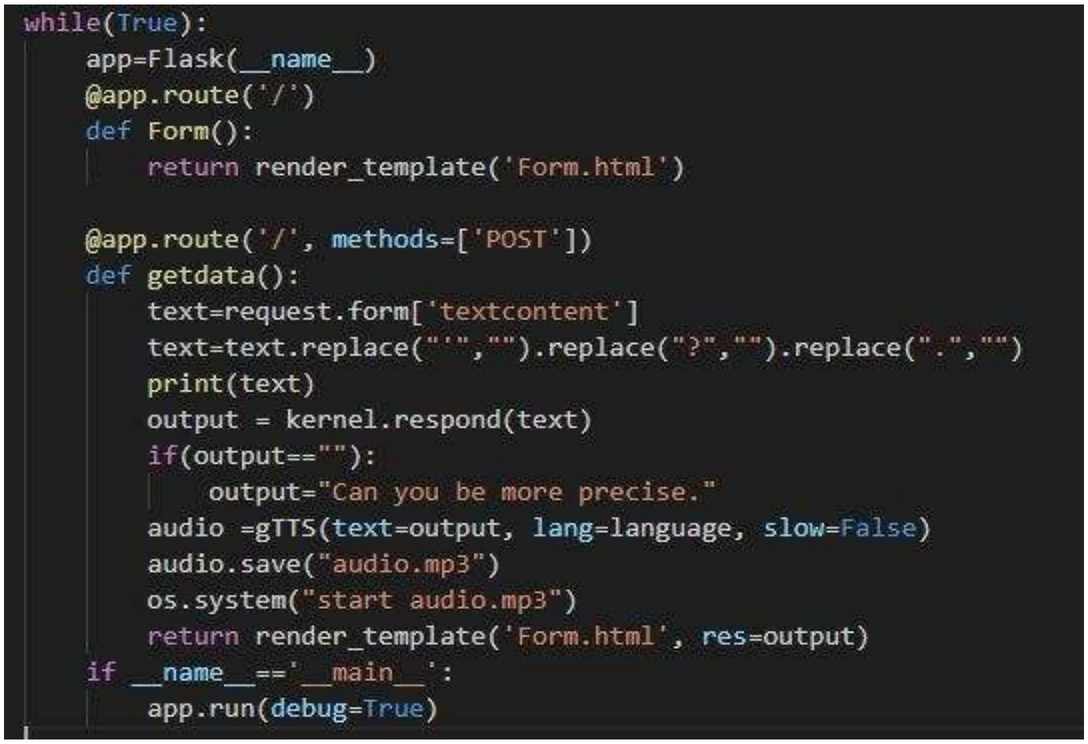
Now, the output of gTTS functions is then saved in a .mp3 file as:

```
Output.save("audio.mp3")
```

After saving the audio file, we can run it using "os" module as:

```
os.system("start audio.mp3")
```

These all changes are only made inside the while loop. Now, after adding the gTTS, finally the while loop looks like:



```
while(True):
    app=Flask(__name__)
    @app.route('/')
    def Form():
        return render_template('Form.html')

    @app.route('/', methods=['POST'])
    def getdata():
        text=request.form['textcontent']
        text=text.replace("'","").replace("?","").replace(".","")
        print(text)
        output = kernel.respond(text)
        if(output==""):
            output="Can you be more precise."
        audio =gTTS(text=output, lang=language, slow=False)
        audio.save("audio.mp3")
        os.system("start audio.mp3")
        return render_template('Form.html', res=output)
    if __name__=='__main__':
        app.run(debug=True)
```

Fig. 3.18: while loop after adding gTTS functionality

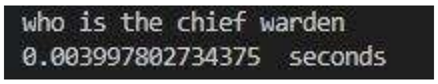
Chapter 04

Performance Analysis

As we know the chatbot gives response by using pattern matching technique for identifying the appropriate and precise response to a particular query. If we talk about the performance, the chatbot is performing very well. It gives answer very quickly to a query asked by a client.

To check its response time, I have used the "time" module of python, using which we can get the execution time of the task.

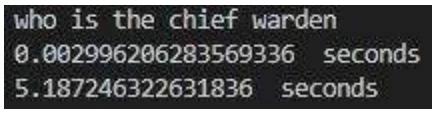
Figure 4.1 shows the execution time of the chatbot to produce the response without using gTTS.



```
who is the chief warden
0.003997802734375  seconds
```

Fig. 4.1: the execution time to produce response without gTTS

The figure 4.2 shows the execution time to produce response with and without audio using gTTS respectively.



```
who is the chief warden
0.002996206283569336  seconds
5.187246322631836  seconds
```

Fig. 4.2: the execution time to produce response with and without gTTS

As we can see in figure 4.1 and 4.2, the chatbot took approximately 0.003 to 0.004 seconds to respond to a particular query, which is very fast. But it is taking a little longer in case while we are using gTTS, this is because gTTS module use internet to convert text to speech and this is the reason why it is taking approx. 5 seconds to give response in case when we are using gTTS.

Further the performance of the chatbot can be checked based on how it performs in the real environment while interaction with the real clients. It can me measured according to how many queries it has resolved out of 100 to check how much perfect and personalised it is.

Number of queries it isn't able to resolve will show us the scope of improvement along with the performance measures of the chatbot which we can improve very easily by adding a few more lines of code with respect to the queries not resolved by the chatbot.

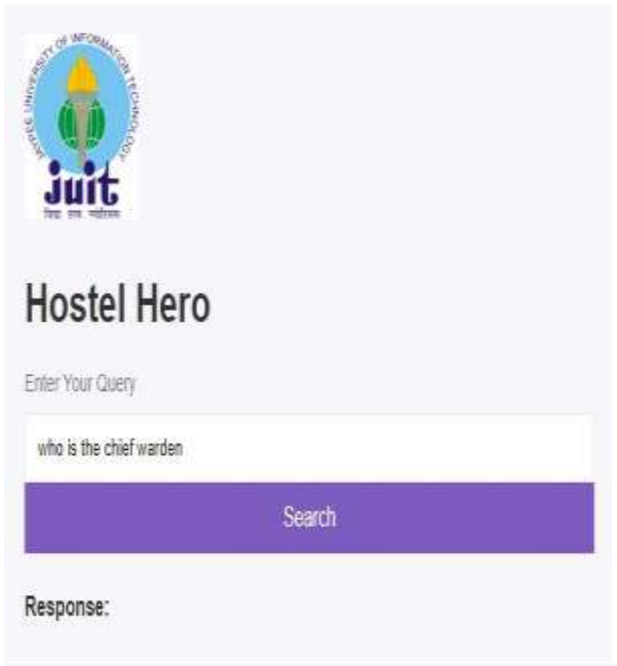
Chapter 05

Conclusion

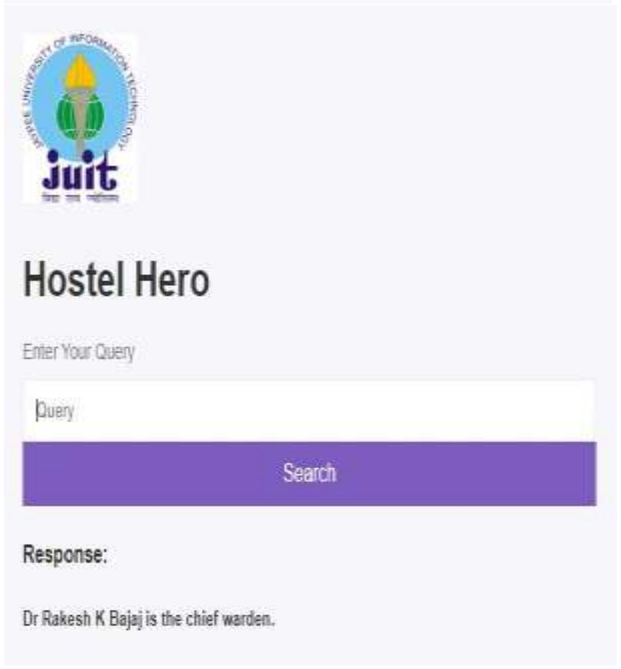
In the end, I can say the doing this project was a nice experience as at the end, as a result we have successfully created a chatbot which is developed using HTML, XML, AIML, Python, CSS, gTTS. This chatbot can resolve any query related to the JUIT Hostels. You can ask it about chief warden, fee, wardens, hostels, do's and don'ts or anything that is related to hostels and it will most probably solve your problem by providing you the sufficient information you need.

For future works, we can make it a larger scale project by making it to cover the full university means it can answer all the queries related to the academics, the laundry, the mess and the hostels. This pretty much include each and every aspect of a campus that one visiting the campus may want to know about the campus.

It takes a great effort to make a chatbot personalized to a specific topic but thanks to aiml, we have achieved it. Below given figures shows the working example our chatbot "Hostel Hero".



who is the chief warden



Dr Rakesh K Bajaj is the chief warden.

Fig. 5.1: Query Asked

Fig. 5.2: Response to the query

REFERENCES

1. *A.L.I.C.E Artificial Intelligence Foundation*, [online] Available: http://alice.pandorabots.com.

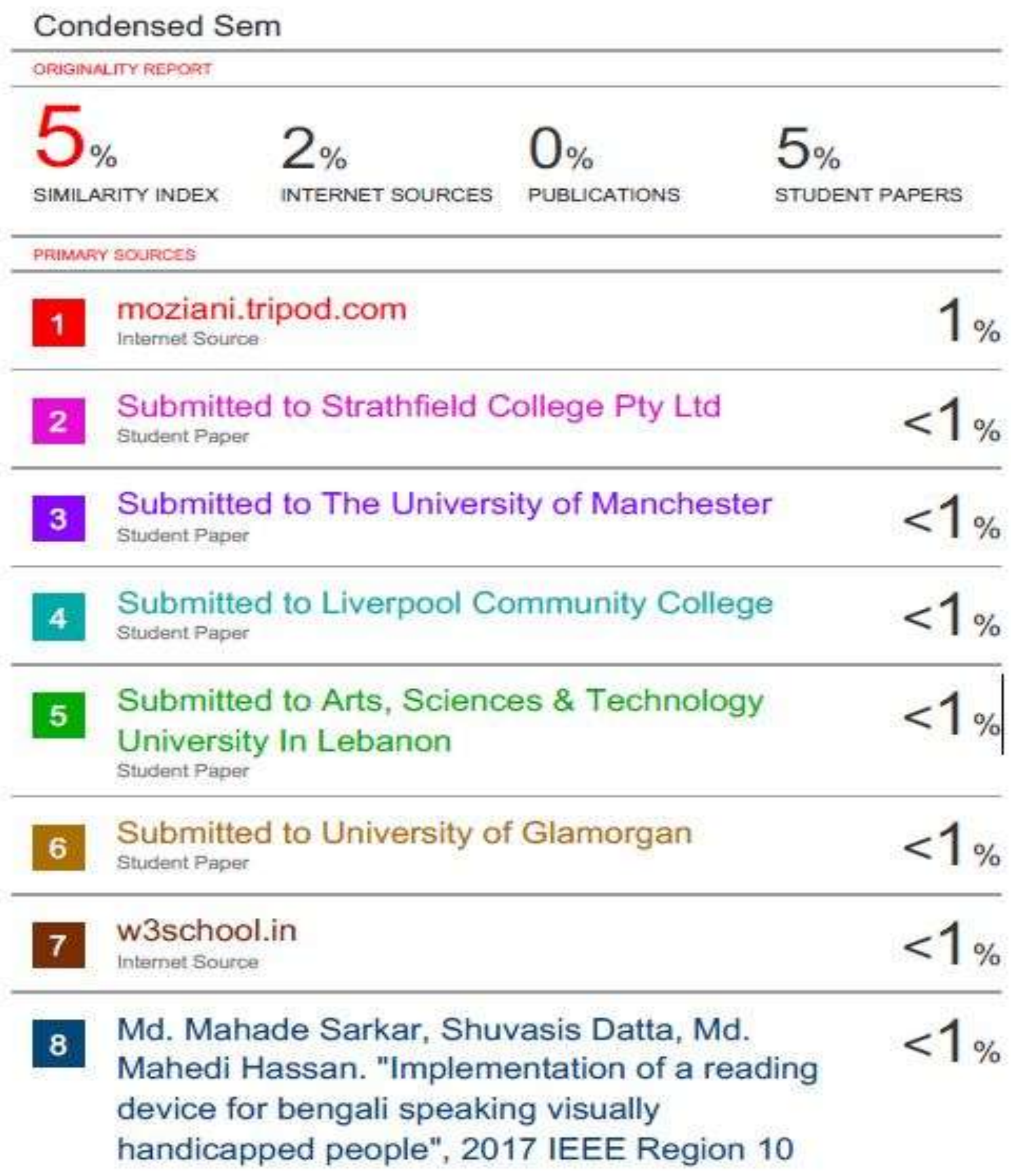
2. Bhavika R. Ranoliya, Nidhi Raghuwanshi, Sanjay Singh, "Chatbot for university related FAQs", *Advances in Computing Communications and Informatics (ICACCI) 2017 International Conference on*, pp. 1525-1530, 2017.

3. Shahriare Satu, Hasnat Parvez and Shamim-Al-Mamun, "Review of integrated applications with AIML based chatbot", *2015 International Conference on Computer and Information Engineering (ICCIE)*, pp. 87-90, 2015.

4. J. Weizenbaum, "Elizaa computer program for the study of natural language communication between man and machine", *Communications of the ACM*, vol. 9, no. 1, pp. 36-45, 1966.

5. 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. 125-130, 2018.

Plagiarism Report



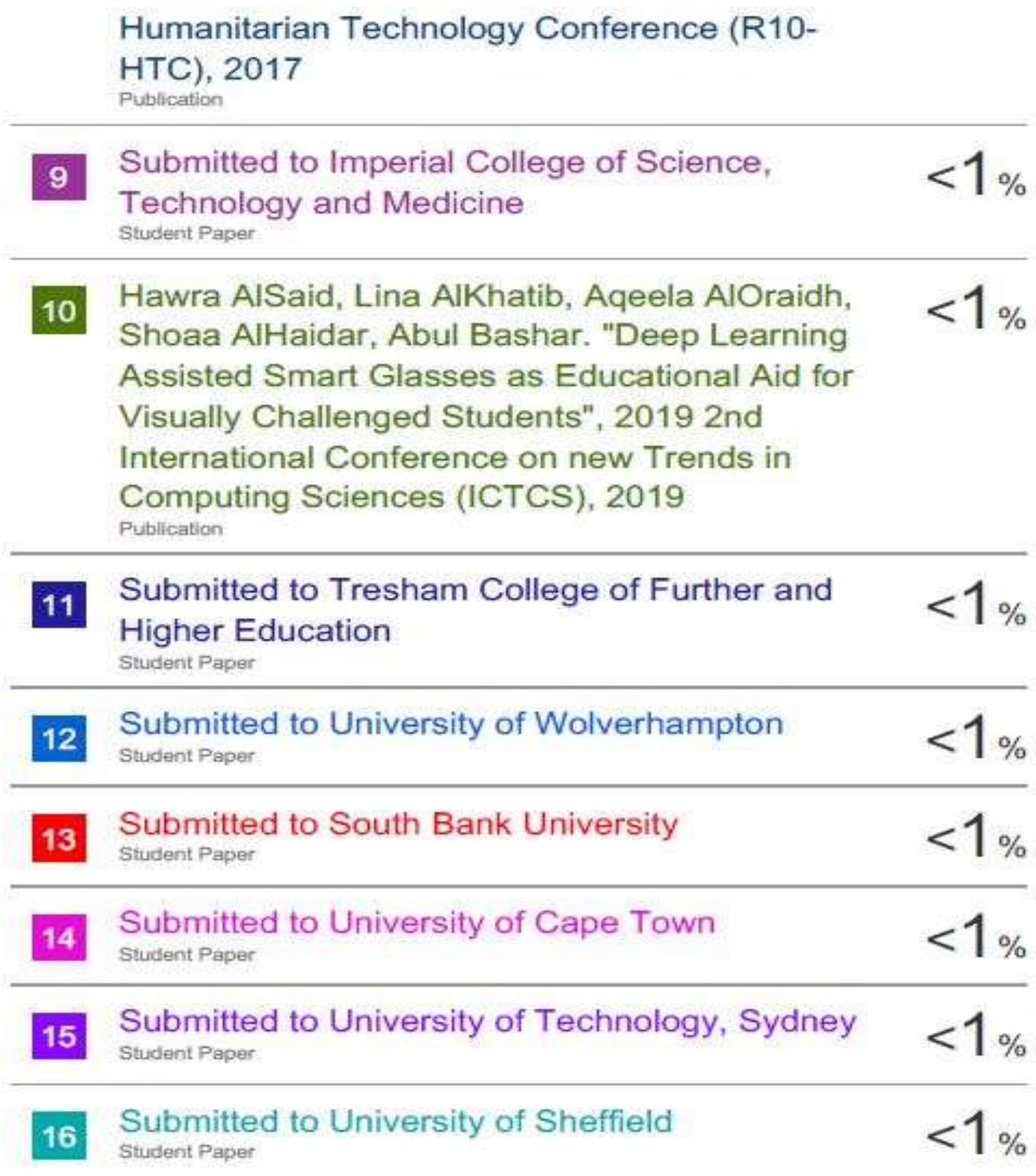
Condensed Sem

ORIGINALITY REPORT

5%	2%	0%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	moziani.tripod.com Internet Source	1%
2	Submitted to Strathfield College Pty Ltd Student Paper	<1%
3	Submitted to The University of Manchester Student Paper	<1%
4	Submitted to Liverpool Community College Student Paper	<1%
5	Submitted to Arts, Sciences & Technology University In Lebanon Student Paper	<1%
6	Submitted to University of Glamorgan Student Paper	<1%
7	w3school.in Internet Source	<1%
8	Md. Mahade Sarkar, Shuvasis Datta, Md. Mahedi Hassan. "Implementation of a reading device for bengali speaking visually handicapped people", 2017 IEEE Region 10	<1%



	Humanitarian Technology Conference (R10-HTC), 2017 Publication	
9	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1%
10	Hawra AlSaid, Lina AlKhatib, Aqeela AlOraidh, Shoaa AlHaidar, Abul Bashar. "Deep Learning Assisted Smart Glasses as Educational Aid for Visually Challenged Students", 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019 Publication	<1%
11	Submitted to Tresham College of Further and Higher Education Student Paper	<1%
12	Submitted to University of Wolverhampton Student Paper	<1%
13	Submitted to South Bank University Student Paper	<1%
14	Submitted to University of Cape Town Student Paper	<1%
15	Submitted to University of Technology, Sydney Student Paper	<1%
16	Submitted to University of Sheffield Student Paper	<1%

17 Submitted to University of Greenwich
Student Paper
<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16th JULY 2020

Type of Document: **B.Tech Project Report**

Name:__Navneet Thakur__ Department:__CSE__ EnrolmentNo:__161215

ContactNo.__9418669639__

E-mail.__thakurnavneet31@gmail.com__

Name oftheSupervisor:__Dr. Hemraj Saini__

Title of the Thesis/Dissertation/Project Report/Paper (In Capitalletters):

HOSTEL HERO: AN AIMLCHATBOT

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages= 39
- Total No. of Preliminary pages= 5
- Total No. of pages accommodate bibliography/references= 1

[illegible]

(Signature of Student)

FOR DEPARTMENT USE

Wehavecheckedthethesis/reportaspernormsandfound**SimilarityIndex**at .... 5(%). Therefore,we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible]

(SignatureofGuide/Supervisor) **Signature ofHOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• AllPreliminary Pages • Bibliography/Images/Quotes • 14 WordsString		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	