

# **Artificial Assistant for Department of Information Technology**

**A**

## **Synopsis/Project Report**

*Submitted in partial fulfillment for the award of the degree  
of*  
**Bachelor of Engineering in Information Technology**

*Submitted to*



**DEPARTMENT OF INFORMATION TECHNOLOGY  
UNIVERSITY INSTITUTE OF TECHNOLOGY  
RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P)**

*Submitted by*

**Anshuman Prajapati (0101IT181010), Ashutosh Gupta (0101IT181013),  
Samarpit Asnani (0101IT181047), Saniya Zuberi (0101IT181049)**

*Under the Guidance of*

**Dr. Asmita A. Moghe**  
(Head of IT department, UIT RGPV Bhopal)

**July, 2021**

**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**UNIVERSITY INSTITUTE OF TECHNOLOGY**  
**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P)**



**CERTIFICATE**

This is to certify that the project entitled “**AADIT (Artificial Assistant for Department of Information Technology)**” being submitted by “**Ashutosh Gupta (0101IT181013.)**” students of “**6th**”, Department of Information Technology towards partial fulfillment of Bachelor of Engineering in Information Technology from University Institute of Technology, RGPV, Bhopal (M.P) is a record of bonafide work carried out by them under my supervision.

***Guided by***

**Dr. Asmita A. Moghe**

**HOD**

**Department of Information Technology,  
UIT, RGPV, Bhopal.**

***Forwarded by***

**Dr. Asmita A. Moghe**

**HOD**

**Department of Information Technology,  
UIT, RGPV, Bhopal.**

**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**UNIVERSITY INSTITUTE OF TECHNOLOGY**  
**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P)**



**DECLARATION**

We declare that project entitled “**AADIT (Artificial Assistant for Department of Information Technology)**” is our own work conducted under the supervision of “**Dr. Asmita A. Moghe.**”, Department of Information Technology, University Institute of Technology, RGPV, Bhopal (M.P).

We further declare that, to the best of our knowledge the project does not contain any work which has been submitted for the award of the degree either in the University or in any other University/Deemed University without proper citations.

Place: Bhopal.  
Date: 16/7/2021

***Submitted by***  
**Anshuman Prajapati (101IT181010)**  
**Ashutosh Gupta (0101IT181013)**  
**Samarpit Asnani (0101IT181047)**  
**Saniya Zuberi (0101IT181049)**

**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**UNIVERSITY INSTITUTE OF TECHNOLOGY**  
**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA,**  
**BHOPAL (M.P)**

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Dr. Asmita A. Moghe (Head of Information Technology, UIT RGPV Bhopal) for her guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude to my colleague in developing the project and people who have willingly helped me out with their abilities.

My thanks and appreciations also towards my parents & members of Information Technology, UIT RGPV for their kind cooperation and encouragement which help me in completion of this project.

# ABSTRACT

Chatbots, or conversational interfaces as they are also known, present a new way for individuals to interact with computer systems. Traditionally, to get a question answered by a software program involved using a search engine, or filling out a form. A chatbot allows a user to simply ask questions in the same manner that they would address a human. The most well-known chatbots currently are voice chatbots: Alexa and Siri. However, chatbots are currently being adopted at a high rate on computer chat platforms. The technology at the core of the rise of the chatbot is natural language processing (“NLP”). Recent advances in machine learning have greatly improved the accuracy and effectiveness of natural language processing, making chatbots a viable option for many organizations. This improvement in NLP is firing a great deal of additional research which should lead to continued improvement in the effectiveness of chatbots in the years to come.

## **Table of content**

1. Introduction
  - 1.1. Background of topic
2. Background study (Include previous work)
  - 2.1. Introduction
  - 2.2. Problem definition
  - 2.3. Objective and scope of work
  - 2.4. Proposed methodology
  - 2.5. Hardware and software requirements
  - 2.6. Conclusion
3. Proposed system (System Design)
  - 3.1. Introduction
  - 3.2. Technology used
  - 3.3. Flow diagram or chart
  - 3.4. ER diagram
  - 3.5. Request response cycle
  - 3.6. Data flow diagram
  - 3.7. Object Oriented Analysis and Design: UML diagrams
  - 3.8. Conclusion
4. Implementation and result analysis (Design Specification)
  - 4.1. Introduction
  - 4.2. Simulation or implementation tool details (Front end and back end)
  - 4.3. High level design (Include some important code, front end snapshots)
  - 4.4. Low level design (Data base design or back-end tables)
  - 4.5. Result analysis and unit testing (test plan)
  - 4.6. Conclusion
5. Conclusion, Work outcomes or major finding and future directions
6. References

# Introduction

A CHATBOT is an artificial person, animal or other creature which holds conversations with humans. This could be a text based (typed) conversation, a spoken conversation or even a non-verbal conversation. Chat bot can run on local computers and phones, though most of the time it is accessed through the internet. Chat bot is typically perceived as engaging software entity which humans can talk to. It can be interesting, inspiring and intriguing. It appears everywhere, from old ancient HTML pages to modern advanced social networking websites, and from standard computers to fashionable smart mobile devices. Chat bots talk in almost every major language. Their language (Natural Language Processing, NLP) skills vary from extremely poor to very clever intelligent, helpful and funny. The same counts for their graphic design, sometimes it feels like a cartoonish character drawn by a child, and on the other hand there are photo-realistic 3D animated characters available, which are hard to distinguish from humans. And they are all referred to as “chat bots”.

Chatbot AADIT is a chatbot capable of giving answers on the basis of user queries. It is developed for the purpose of resolving user queries like frequently asked questions. Introduce “AADIT” as an alternative to existing FAQs section and help reduce workload of teams currently engaged in redundant task of responding to same kind of queries from end users

So, the chatbot being developed should be able to do the following activities:

- 1) Answering questions from users which are based on FAQs of the application using NLP techniques
- 2) Displaying departmental information based on user queries.

Thus, introducing a chatbot like AADIT can help us to provide assistance and to digitalize notice board and other information.

**Project Link – <https://github.com/anshumanprajapati/ADDIT>**

# Background study

## 2.1 Introduction

We begin by providing a dissected view of a chatbot to show its major elements. We present five different elements of a chatbot which are part of a typical chatbot. The Intent Classifier classifies any query that the chatbot receives in a class. These classes called the Intents, are predefined during the chatbot building phase. The Parameter Extractor attempts to locate named entities in the query. Similar to Intents, these parameters, called Entities, are defined before the chatbot is placed in operation. The Flow Manager element manages the discourse with the user. Its job is to make sure that the conversation sounds coherent. The Response Generator performs the tasks required to process the query and prepare a response for the user. It may include replying with static messages or invoking a complex processing pipeline that requires interacting with elements of the Containing system. Voice Utils are used when a chatbot supports an audio interface in addition to text messages. They perform the Speech-to-Text and Text-to-Speech conversions. Next, we discuss how the Containing system of the chatbot may affect its design. The System Interface refers to the collection of modes through which the Containing system interacts with its uses. Common interfaces maybe a website, an app or via messaging platforms like Messenger, Telegram, WhatsApp or Slack. The chatbot may be required to face the users at some or all of these interfaces, which can shape its requirements. Actions are business operations that the chatbot needs to invoke as part of processing a query, e.g., create new order function of an e-commerce enterprise. Fulfilments are one or more intermediaries between Actions and the chatbot, and help in shielding the enterprise's business operations from the chatbot. We also suggest a Reference Architecture for any application that contains a conversational interface. The Reference Architecture puts the different pieces of chatbots and the Containing system that we discussed before, in a coherent perspective. As Concrete Architectures for the Reference Architecture, we highlight the variations when different chatbot-building platforms are used for the process.



## 2.2 Problem definition

Changing of requirements as an industrial project to build a product, we must follow the requirement from the user. However, because the project's goal is to be used by the business team, but it is responsible by the technical team, the requirement changed a lot in the middle after a meeting with the business team. The business team want a simple bot that can give recommendation immediately. We had to archive what we had done before and build a new one.

- Lacking of training data. The quantity and quality of training data is critical to the performance to a machine learning model. However, because some confidential and privacy reasons, the business team cannot provide enough data for us, and we had to make up data by our own. For the machine learning model, we generate some fake data based on our daily life experience, which is really biased, although with a good accuracy on the fake dat.

- Unstable Voice to text API. Because API service we are using are still under development, and we cannot fix to a version for the API, the API may change overtime. Voice to text only works on the browsers based on Chromium.

## **2.3 Objective and scope of work**

Chatbots are mostly used to assist users. It has a high level of intelligence. You just need to train them once, and they'll be able to communicate with your target audience in their native tongue. A lot of money can be saved by using multilingual chatbots instead of paying various language resources. Furthermore, chatbots minimize human effort because there is no constructive work involved in answering the same queries of users.

With the increase in adoption of messaging and video-based technologies, the potential of conversational chatbots is also realized. Chatbots are powered by a set of rules and instructions that help learners to communicate via an easy to access chat interface. Users can ask questions and the chatbots carry out specific actions in their response. They are mainly considered as automated systems that run within messaging apps such as SMS, Messenger, or any other mobile app. As per Gartner, around 85% of customer interactions will be managed without humans by 2020.

## **2.4 Proposed methodology**

ChatterBot is a Python library that is designed to deliver automated responses to user inputs. It makes use of a combination of ML algorithms to generate many different types of responses. This feature allows developers to build chatbots using python that can converse with humans and deliver appropriate and relevant responses. Not just that, the ML algorithms help the bot to improve its performance with experience.

Another excellent feature of ChatterBot is its language independence. The library is designed in a way that makes it possible to train your bot in multiple programming languages.

Frontend was designed in HTML, CSS and Javascript. HTML and CSS being used for markup and styling of user interface, while Javascript and AJAX was used for making the interface dynamic, user friendly and for building API which binds the front end, backend and ML together.

## **2.5 Hardware and software requirements**

### **Minimum Software Requirements:**

- Any operating system capable of running a web browser.
- Google Chrome Firefox / Internet Explorer Any other web browser.

### **Minimum Hardware Requirements:**

- Processor – i3/Any processor capable of running a web browser.
- Hard Disk – 5 GB
- Memory – 1GB RAM

## **2.6 Conclusion**

What we've illustrated here is just one among the many ways of how to make a chatbot in Python. You can also use NLTK, another resourceful Python library to create a Python chatbot. And although what you learned here is a very basic chatbot in Python having hardly any cognitive skills, it should be enough to help you understand the anatomy of chatbots.

Once you understand the design of a chatbot using python fully well, you can experiment with it using different tools and commands to make it even smarter.

# System Design

## 3.1 Introduction

ChatterBot is a Python library that is designed to deliver automated responses to user inputs. It makes use of a combination of ML algorithms to generate many different types of responses. This feature allows developers to build chatbots using python that can converse with humans and deliver appropriate and relevant responses. Not just that, the ML algorithms help the bot to improve its performance with experience.

Another excellent feature of ChatterBot is its language independence. The library is designed in a way that makes it possible to train your bot in multiple programming languages.

To build a chatbot in Python, you have to import all the necessary packages and initialize the variables you want to use in your chatbot project. Also, remember that when working with text data, you need to perform data preprocessing on your dataset before designing an ML model.

This is where tokenizing helps with text data – it helps fragment the large text dataset into smaller, readable chunks (like words). Once that is done, you can also go for lemmatization that transforms a word into its lemma form. Then it creates a pickle file to store the python objects that are used for predicting the responses of the bot.

Another vital part of the chatbot development process is creating the training and testing datasets.

## 3.2 Technology used

In this Project we have used Python Programming Language (Version 3.7) along with HTML, CSS, Machine Learning and Artificial Intelligence. We have also utilized a number of python modules to implement various features and functionalities. These tools, modules and libraries include the following:

### 3.2.1 Front End

**HTML:** Hypertext Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages.

**Cascading Style Sheet:** CSS is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

**JavaScript:** Javascript is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g., functional programming) styles. We have used anime.js for few animations and some JQuery in AJAX while building API.

### 3.2.2 Backend

**NLTK:** (Natural Language Toolkit): NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

**TensorFlow:** TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.] It is used for both research and production at Google.

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

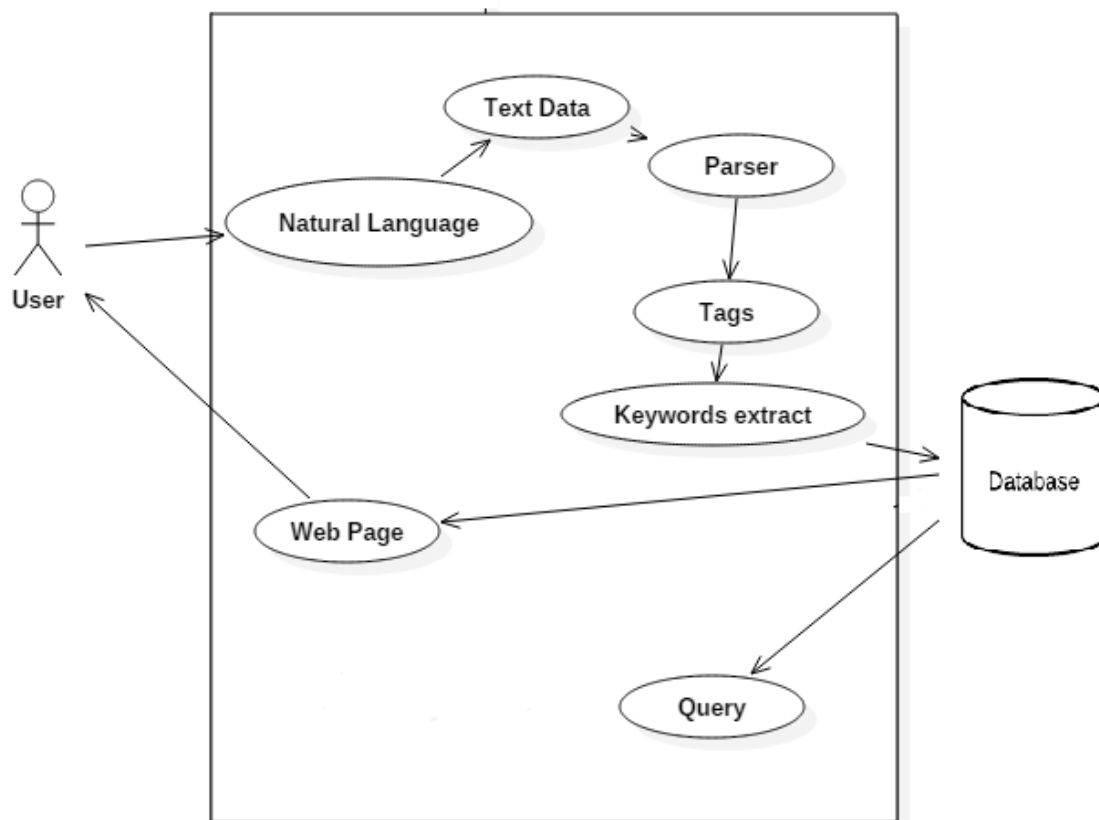
**ChatterBot:** ChatterBot is a machine-learning based conversational dialog engine build in Python which makes it possible to generate responses based on collections of known conversations. The language independent design of ChatterBot allows it to be trained to speak any language.

**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.

**PostgreSQL:** PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

### 3.3 Flow diagram

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

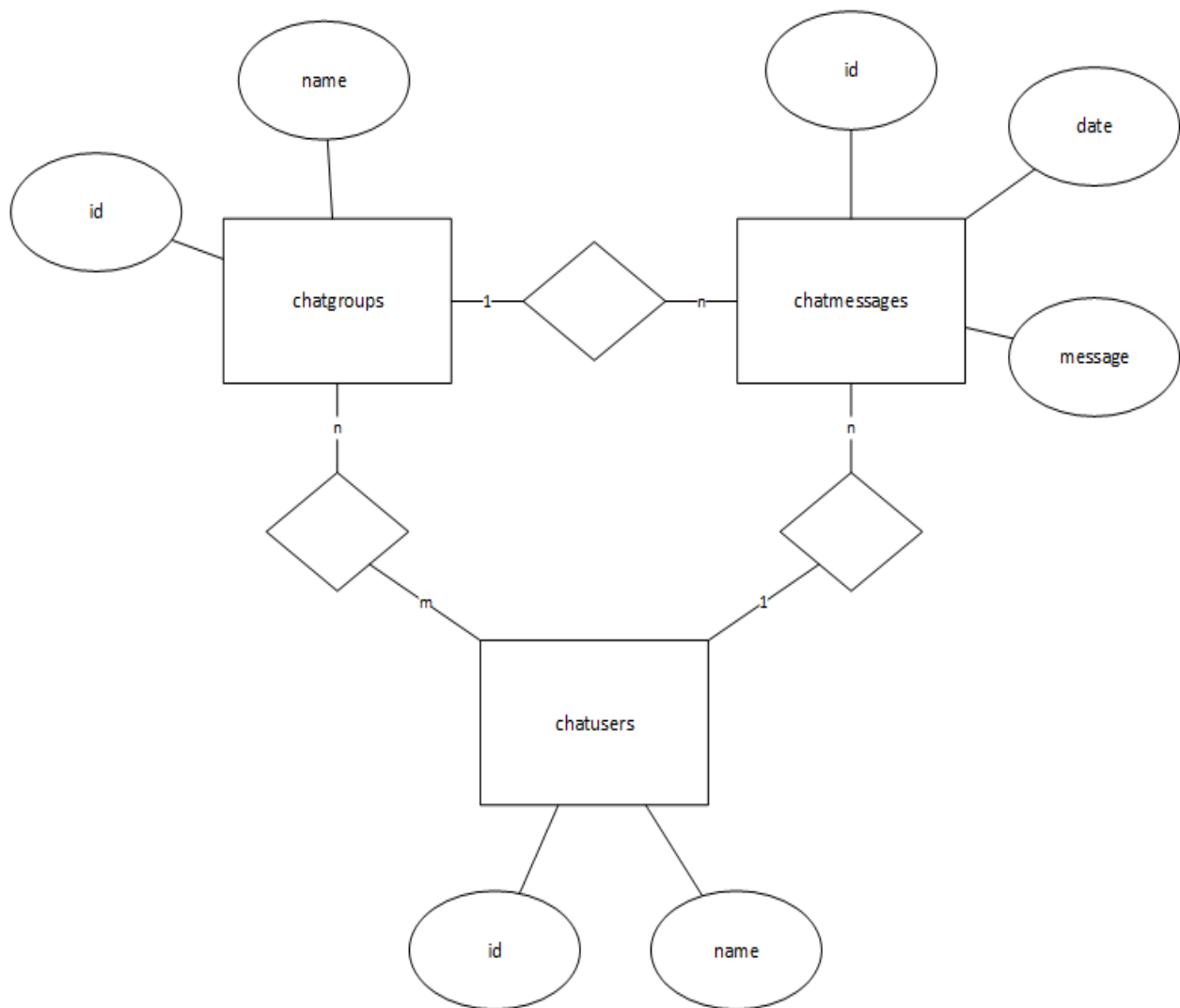


### 3.4 ER diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

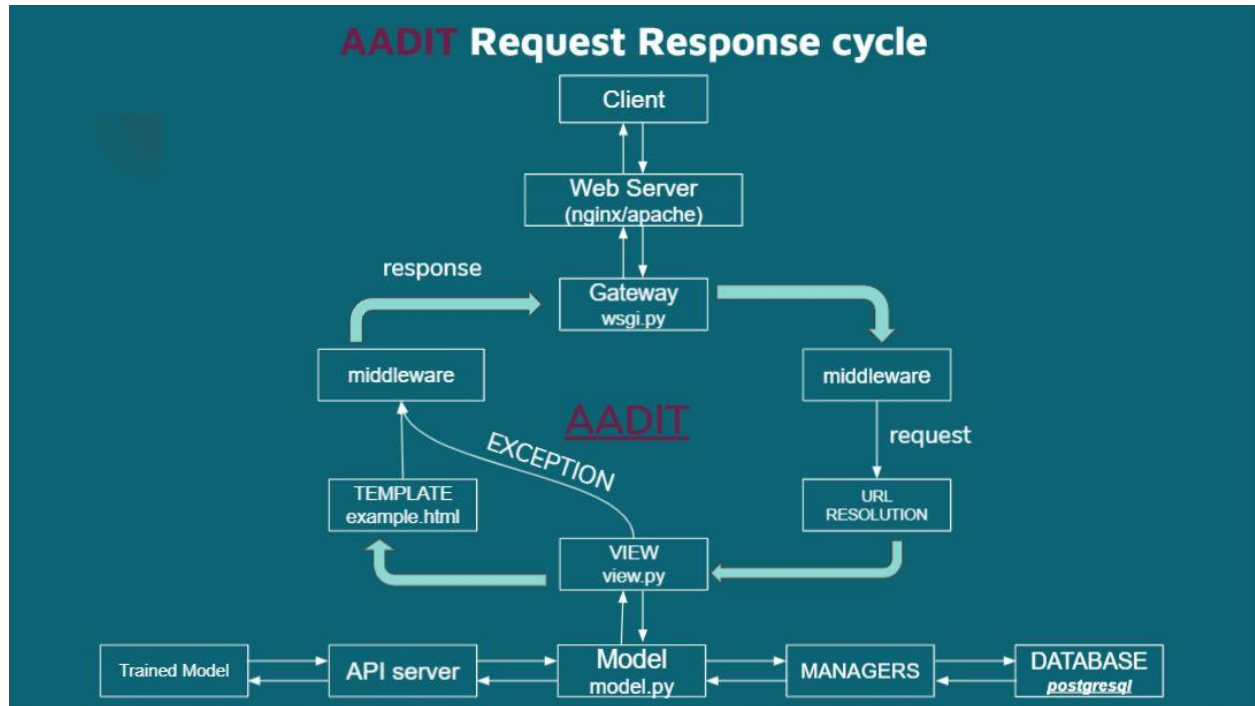
By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

ER diagrams are used to sketch out the design of a database.

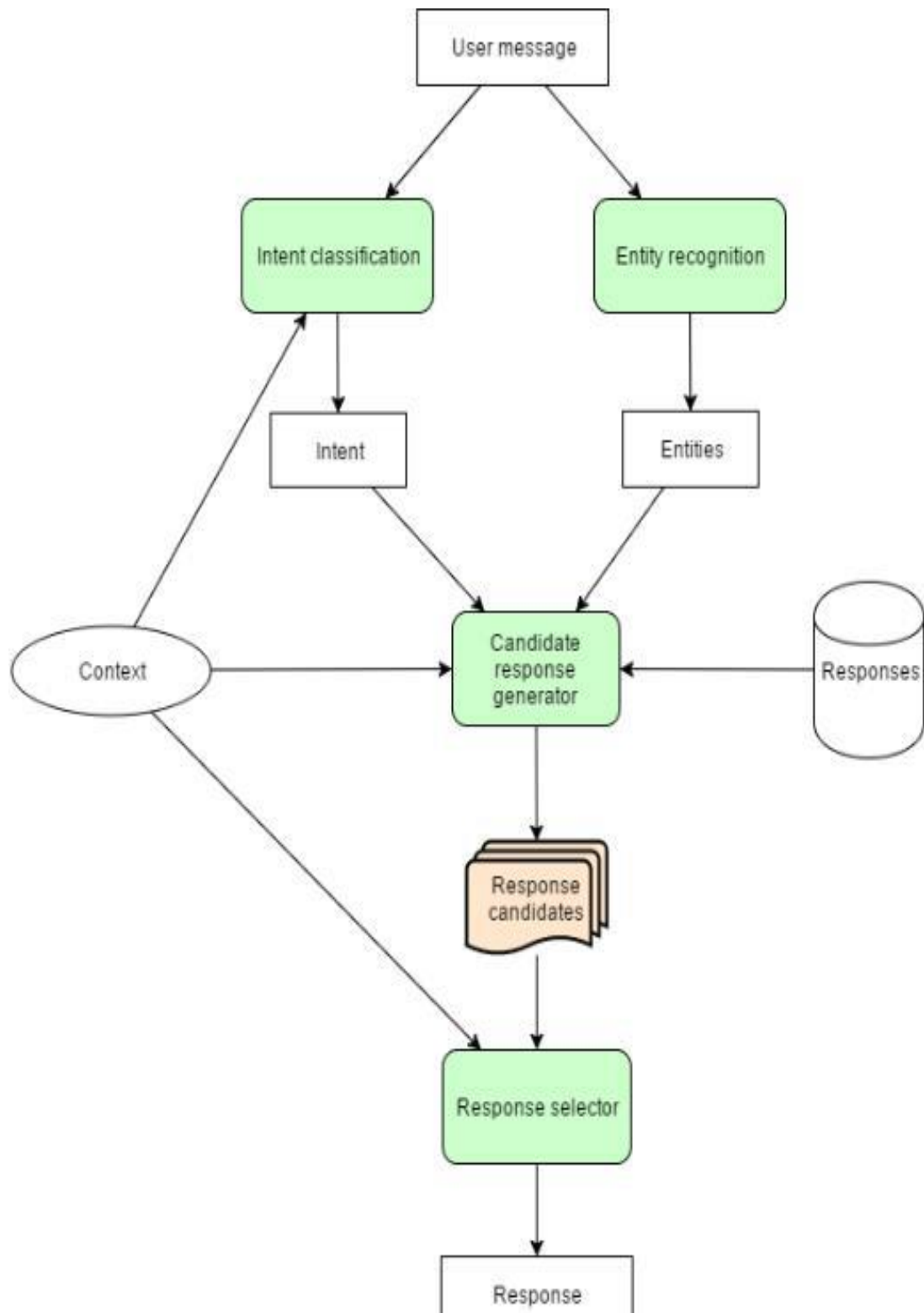




### 3.5 Request response cycle

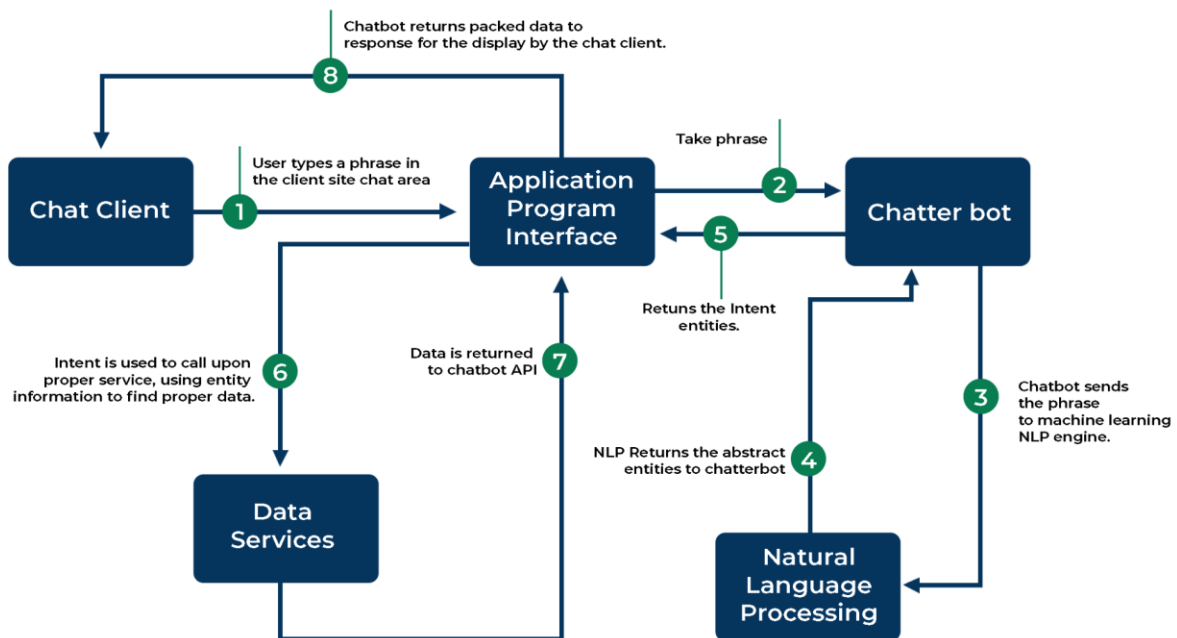


### 3.6 Data flow diagram



### 3.7 Object Oriented Analysis and Design: UML diagrams

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Website. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.



## 3.8 Conclusion

What we've illustrated here is just one among the many ways of how to make a chatbot in Python. You can also use NLTK, another resourceful Python library to create a Python chatbot. And although what you learned here is a very basic chatbot in Python having hardly any cognitive skills, it should be enough to help you understand the anatomy of chatbots.

Once you understand the design of a chatbot using python fully well, you can experiment with it using different tools and commands to make it even smarter.

# **Implementation and result analysis (Design Specification)**

## **4.1 Introduction**

The user interface of the page was designed as a single webpage divided in three vertical sections, each section serving a different purpose. First section being the Chatbot section, second being department information section, while the third section reserved for notices board, academic information, and other updates.

To build a chatbot in Python, you have to import all the necessary packages and initialize the variables you want to use in your chatbot project. Also, remember that when working with text data, you need to perform data preprocessing on your dataset before designing an ML model.

This is where tokenizing helps with text data – it helps fragment the large text dataset into smaller, readable chunks (like words). Once that is done, you can also go for lemmatization that transforms a word into its lemma form.

## 4.2 Simulation or implementation tool details

**VsCode:** Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

**GitHub:** GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features.

**Browser DevTools:** Web development tools (often called devtools) allow web developers to test and debug their code. ... Web development tools allow developers to work with a variety of web technologies, including HTML, CSS, the DOM, JavaScript, and other components that are handled by the web browser.

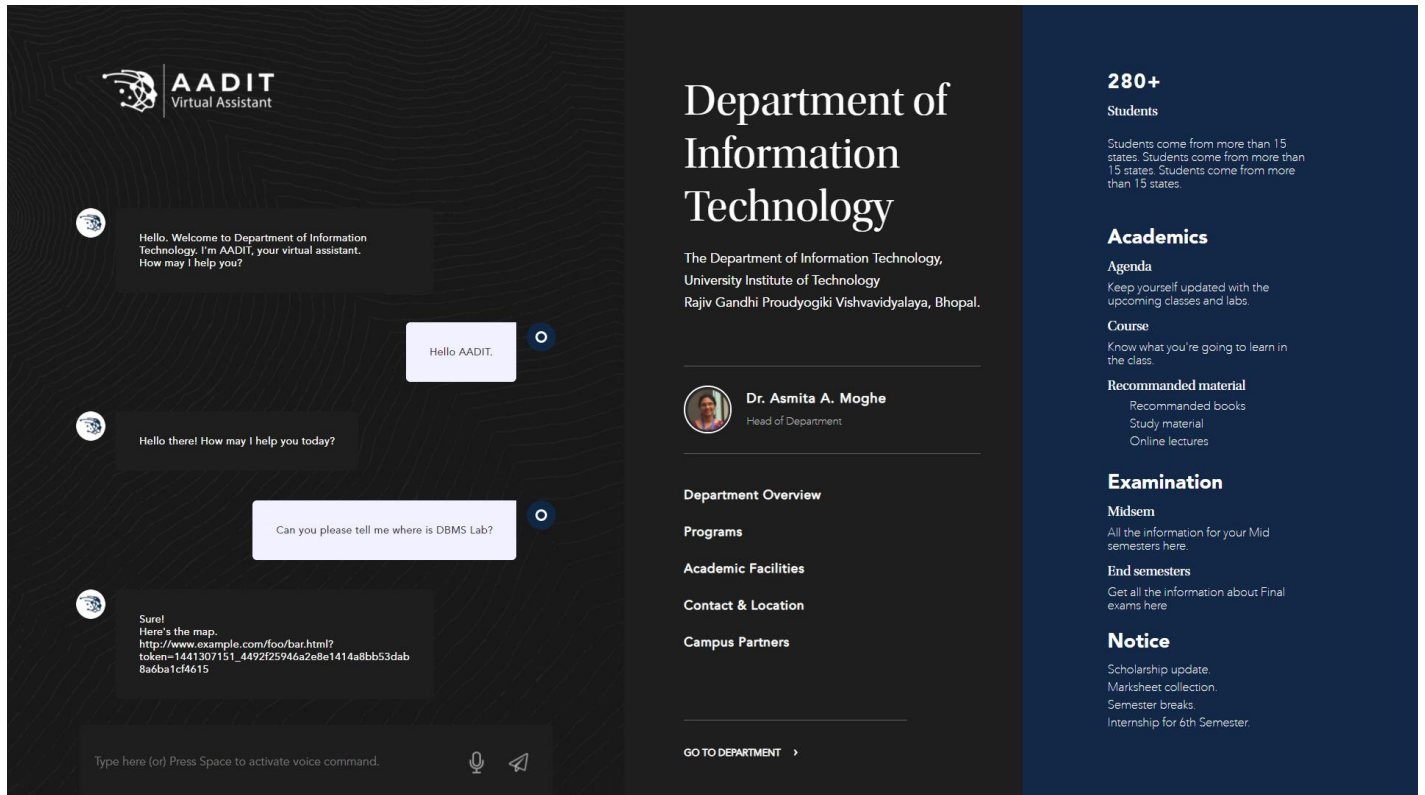
**Django:** Django is a Python-based free and open-source web framework that follows the model–template–views architectural pattern. It is maintained by the Django Software Foundation, an American independent organization established as a 501 non-profit.

**PgAdmin:** PgAdmin is the most popular and feature rich Open-Source administration and development platform for PostgreSQL, the most advanced Open-Source database in the world.

**PostgreSQL:** PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley.,

## 4.3 High level design (Include some important code, front end snapshots)

The final design's front end screenshot is shown below.



## HTML code snippet.

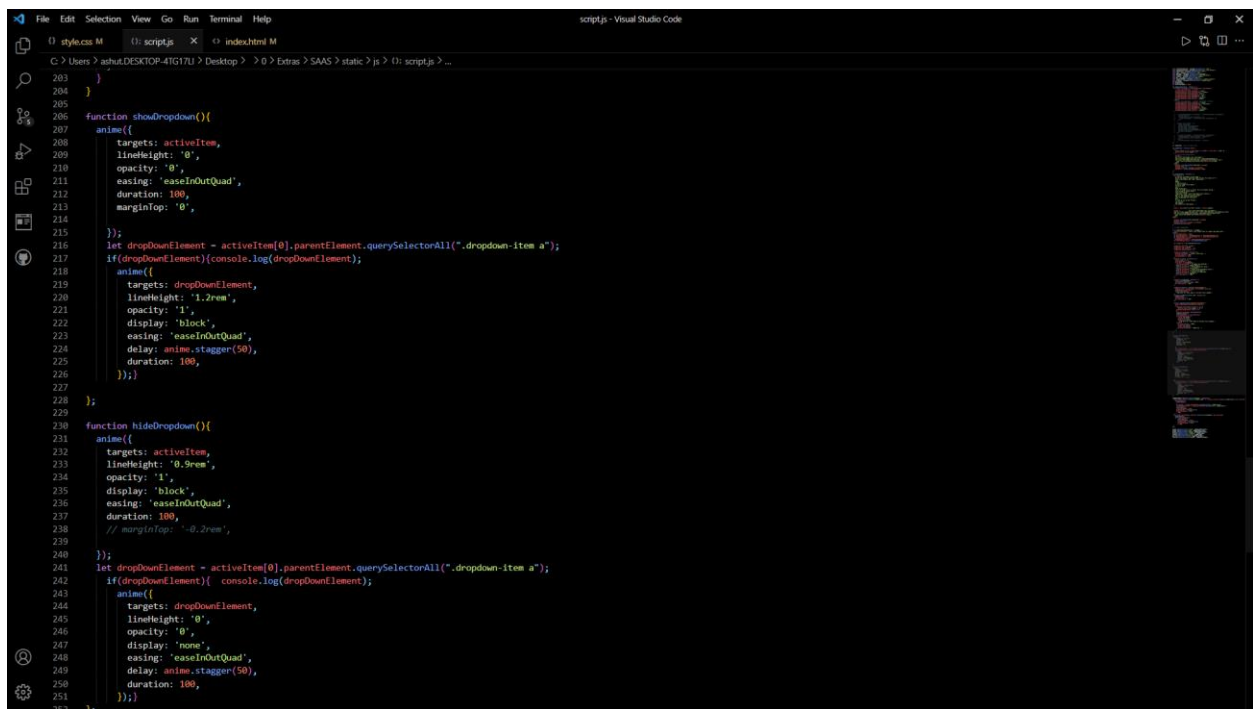
```
index.html - Visual Studio Code
C:\Users> cd Desktop > index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8" />
5 <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7 <link rel="stylesheet" href="static/css/style.css" />
8 <link rel="stylesheet" href="static/css/menuicon.css" />
9 <link rel="preconnect" href="https://fonts.gstatic.com" />
10 <link href="https://fonts.googleapis.com/css2?family=Cormorant&family=Crimson+Text&family=Source+Serif+Pro:ital@1&display=swap" rel="stylesheet">
11 <title>Document</title>
12 </head>
13
14 <body>
15 <div class="main-container">
16
17 <div style="position: fixed; top: 0; right: 0; z-index: 999;">
18 <div class="wrapper-menu">
19 <div class="line-menu half start"></div>
20 <div class="line-menu"></div>
21 <div class="line-menu half end"></div>
22 </div>
23 </div>
24
25
26
27 <!-- Col-2 -->
28 <div class="info">
29 <div class="info-content">
30 <div class="dept-name">Department of Information Technology</div>
31 <p>The Department of Information Technology, University Institute of Technology <span>Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal.</span></p>
32 <div class="hod">
33 
34 <p class="hod-info">Dr. Asmita A. Moghe <span class="designation">Head of Department</span></p>
35 <div class="overview-list">
36 <ul>
37 <li><a href="#">Department Overview</a></li>
38 <li><a href="#">Programs</a></li>
39 <li><a href="#">Academic Facilities</a></li>
40 <li><a href="#">Contact & Location</a></li>
41 <li><a href="#">Campus Partners</a></li>
42 </ul>
43 <div class="go-to-dept"><a href="#">GO TO DEPARTMENT</a></div>
44 <div class="chat">
45 <div class="chat-content">
46 
47 </div>
48 </div>
49 </div>
50 </div>
```

## CSS code snippet.

```
style.css - Visual Studio Code
C:\Users> cd Desktop > style.css > ...
1 :root {
2 --primary-color: #303030;
3 /* --secondary-color: #193970; */
4 --secondary-color: #193970;
5 --secondary-color-transparent: rgba(19, 39, 70, 0.5);
6 --text-color: #193970;
7 --text-color-transparent: rgba(19, 39, 70, 0.7);
8 --division-color: #193970;
9 --chat-bubble-color: #193970;
10 }
11
12 @import url("https://fonts.googleapis.com/css2?family=Cormorant&family=Crimson+Text&family=Source+Serif+Pro:ital@1&display=swap");
13
14 @font-face {
15 font-family: font-head;
16 src: url(../fonts/UtopiaStd-Disp.otf);
17 }
18 @font-face {
19 font-family: font-content;
20 src: url(../fonts/Avenir-LT-Std-45-Book-5171.ttf);
21 }
22
23 @font-face {
24 font-family: font-content-light;
25 src: url(../fonts/AvenirNextLTPro-Ultlt.ttf);
26 }
27 @font-face {
28 font-family: font-content-bold;
29 src: url(../fonts/AvenirLTStd-Black.otf);
30 }
31
32
33 html,
34 body {
35 overflow-x: hidden;
36 }
37
38 color: #fff;
39 text-decoration: none;
40
41
42 @media only screen and (min-width: 1024px) {
43 html,
44 body {
45 height: 100%;
46 overflow: hidden;
47 }
48 }
49
50 body {
```

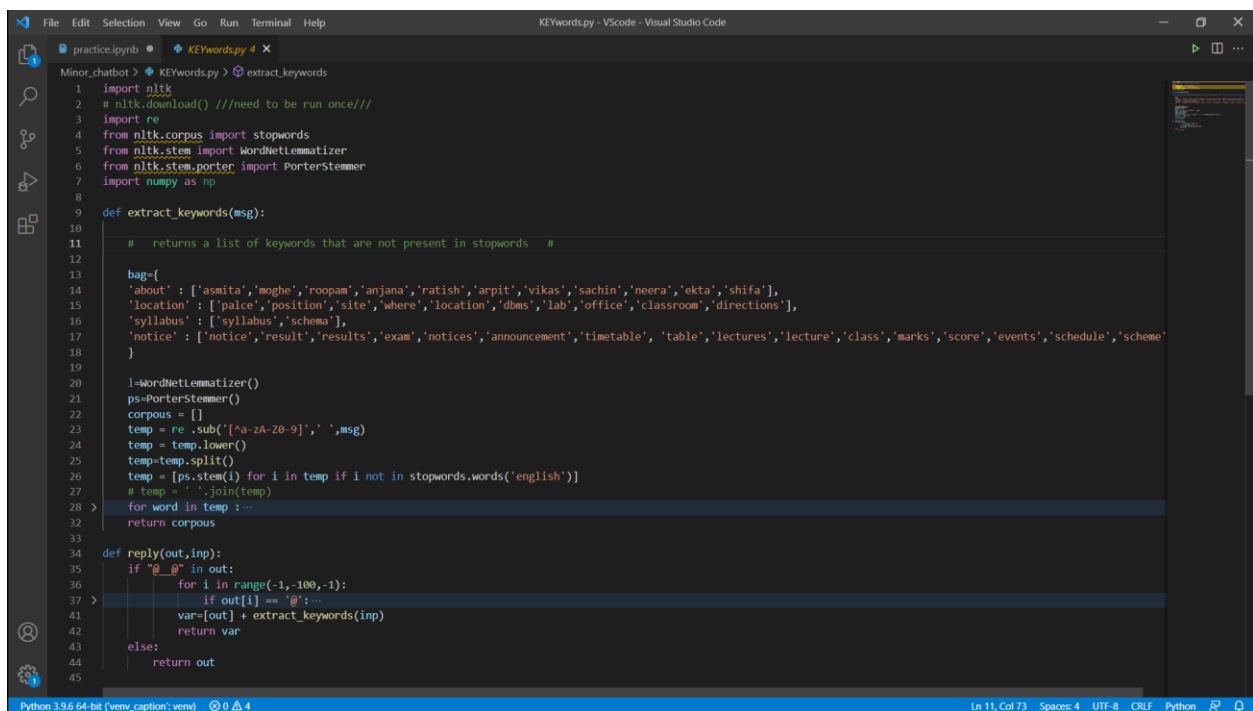


## Javascript code snippet



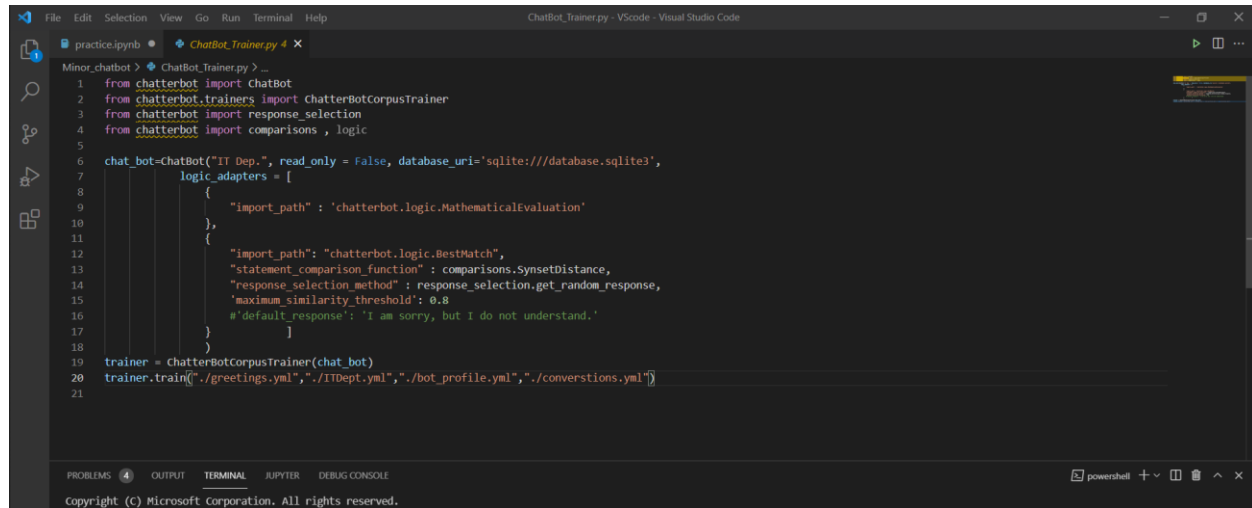
```
203 }
204 }
205
206 function showDropdown(){
207   anime({
208     targets: activeItem,
209     lineHeight: '0',
210     opacity: '0',
211     easing: 'easeInOutQuad',
212     duration: 100,
213     marginTop: '0',
214   });
215
216   let dropdownElement = activeItem[0].parentElement.querySelector(".dropdown-item a");
217   if(dropdownElement){console.log(dropdownElement);
218     anime({
219       targets: dropdownElement,
220       lineHeight: '1.2rem',
221       opacity: '1',
222       display: 'block',
223       easing: 'easeInOutQuad',
224       delay: anime.stagger(50),
225       duration: 100,
226     });
227   }
228 }
229
230 function hideDropdown(){
231   anime({
232     targets: activeItem,
233     lineHeight: '0.9rem',
234     opacity: '1',
235     display: 'block',
236     easing: 'easeInOutQuad',
237     duration: 100,
238     // marginTop: '-0.2rem',
239   });
240
241   let dropdownElement = activeItem[0].parentElement.querySelector(".dropdown-item a");
242   if(dropdownElement){ console.log(dropdownElement);
243     anime({
244       targets: dropdownElement,
245       lineHeight: '0',
246       opacity: '0',
247       display: 'none',
248       easing: 'easeInOutQuad',
249       delay: anime.stagger(50),
250       duration: 100,
251     });
252   }
253 }
```

The following is a snippet of the keyword extraction program.



```
1 import nltk
2 # nltk.download() ///need to be run once///
3 import re
4 from nltk.corpus import stopwords
5 from nltk.stem import WordNetLemmatizer
6 from nltk.stem.porter import PorterStemmer
7 import numpy as np
8
9 def extract_keywords(msg):
10
11     # returns a list of keywords that are not present in stopwords #
12
13     bag={
14         'about' : ['asmita','moghe','roopam','anjana','ratish','arpit','vikas','sachin','neera','ekta','shifa'],
15         'location' : ['palce','position','site','where','location','dbms','lab','office','classroom','directions'],
16         'syllabus' : ['syllabus','schema'],
17         'notice' : ['notice','result','results','exam','notices','announcement','timetable', 'table','lectures','lecture','class','marks','score','events','schedule','scheme']
18     }
19
20     l=WordNetLemmatizer()
21     ps=PorterStemmer()
22     corpus = []
23     temp = re.sub('[a-zA-Z0-9]', ' ',msg)
24     temp = temp.lower()
25     temp=temp.split()
26     temp = [ps.stem(i) for i in temp if i not in stopwords.words('english')]
27     # temp = ' '.join(temp)
28     for word in temp :
29         return corpus
30
31
32
33
34 def reply(out,inp):
35     if "@_@" in out:
36         for i in range(-1,-100,-1):
37             if out[i] == '@':
38                 var=[out] + extract_keywords(inp)
39                 return var
40     else:
41         return out
42
43
44
45
```

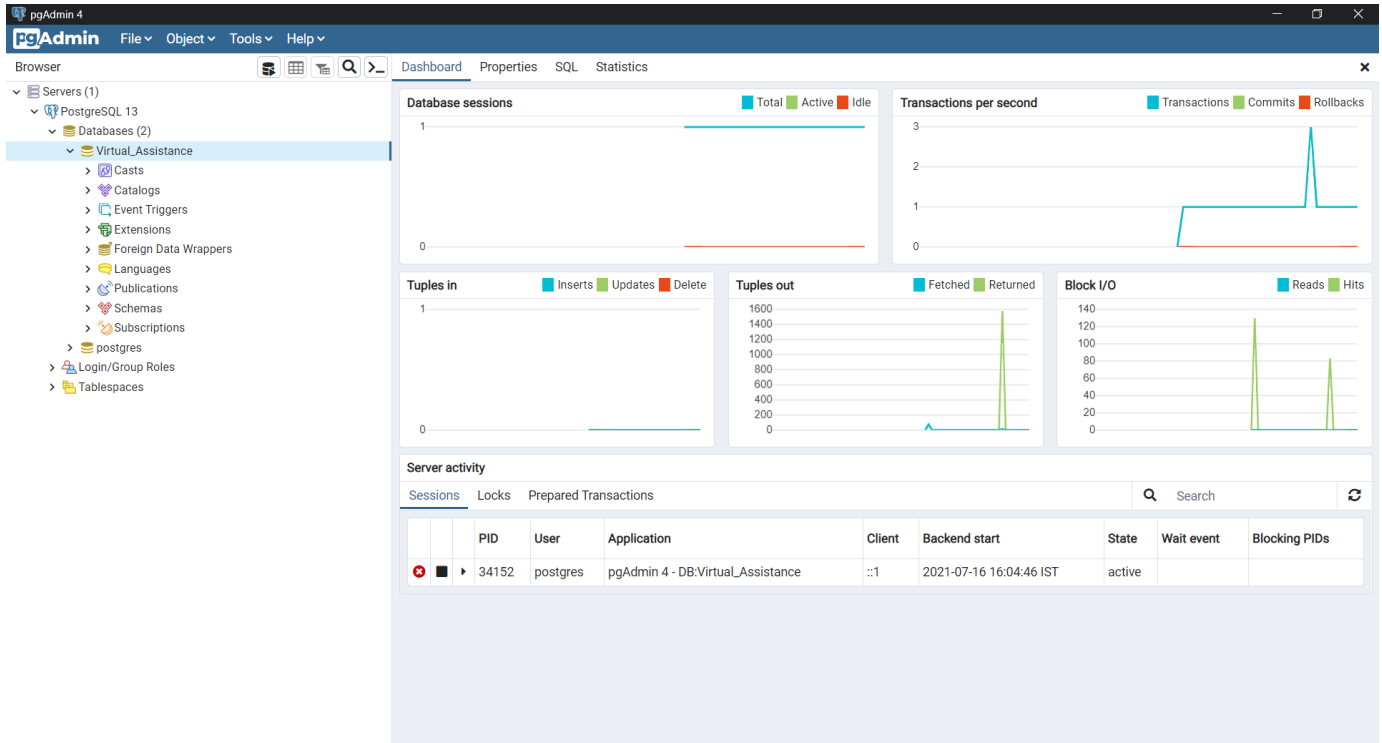
The snippet of the chatterbot training program is shown below.



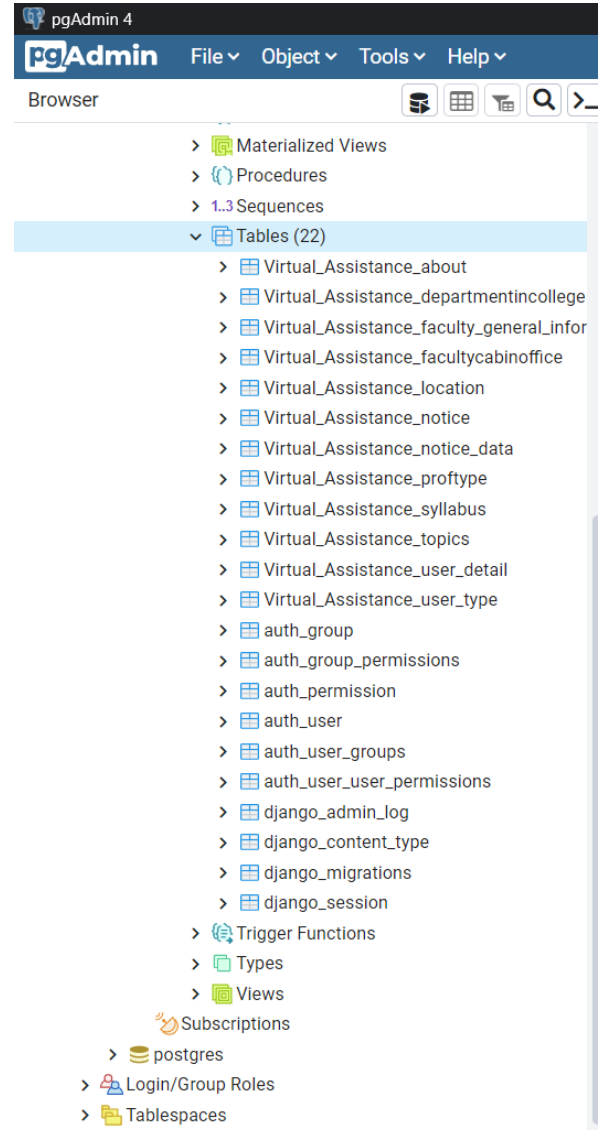
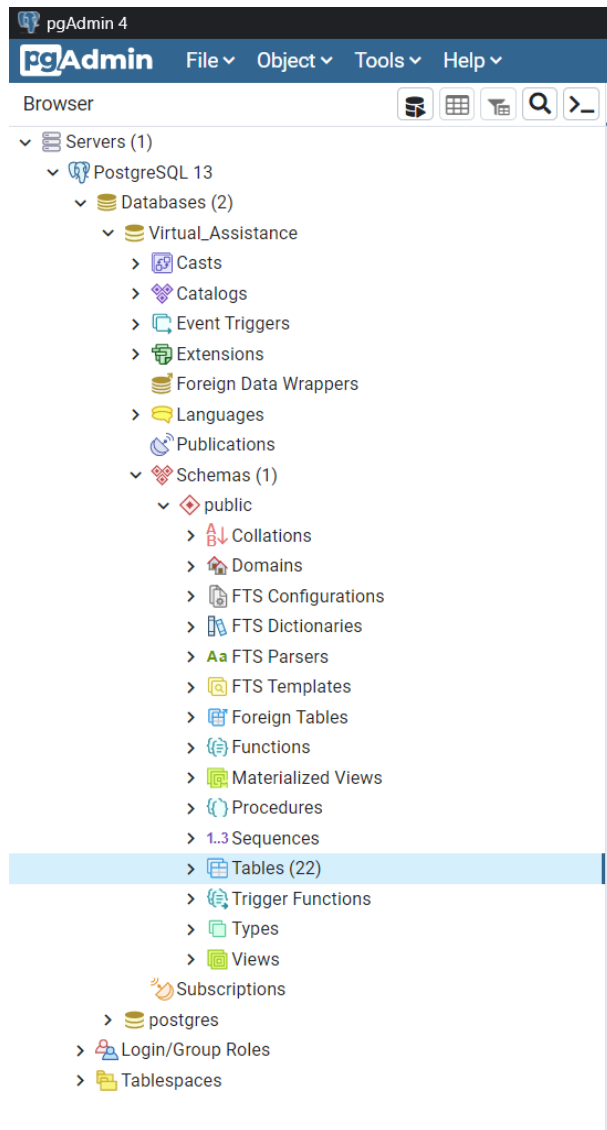
```
1 from chatterbot import ChatBot
2 from chatterbot.trainers import ChatterBotCorpusTrainer
3 from chatterbot import response_selection
4 from chatterbot import comparisons, logic
5
6 chat_bot=ChatBot("IT Dep.", read_only = False, database_uri='sqlite:///database.sqlite3',
7                 logic_adapters = [
8                     {
9                         "import_path": 'chatterbot.logic.MathematicalEvaluation'
10                    },
11                    {
12                        "import_path": "chatterbot.logic.BestMatch",
13                        "statement_comparison_function": comparisons.SynsetDistance,
14                        "response_selection_method": response_selection.get_random_response,
15                        "maximum_similarity_threshold": 0.8
16                    },
17                    {
18                        "import_path": "chatterbot.logic.DefaultResponse",
19                        "default_response": "I am sorry, but I do not understand."
20                    }
21                ])
22
23 trainer = ChatterBotCorpusTrainer(chat_bot)
24 trainer.train(["greetings.yml", "ITDept.yml", "bot_profile.yml", "conversations.yml"])
```

## 4.4 Low level design (Data base design or back-end tables)

Dashboard of PgAdmin for Virtual Assistant database



## Defination table and Entity





## **4.5 Result analysis and unit testing (test plan)**

A Chatterbot trained model was tested, and datasets were enhanced as a result.

During testing, the bot met our expectations, and minor problem fixes were implemented.

Tested user interface for different browsers and screen sizes to ensure that the user interface remains functional. Web speech API is supported in Chromium based browsers only.

During database testing, the chatbot's extracted keywords were utilised to construct a query, and data from the database was successfully obtained and delivered to the front end API.

## **Major finding and future directions**

- Self-training of chatbot
- Improving training dataset
- Enhancement in response time
- Proper Database connectivity
- More Interactive interface
- AI improvement and bugfixes
- Integration Attendance system with facial recognition
- Extending the chat bot from a departmental to a campus-wide level.

# REFERENCES

**Ajax:** <https://jquery.com/>

**Anime.js:** <https://animejs.com/>

**Python** documentation: <https://docs.python.org/3/>

**NLTK:** <https://www.nltk.org/>

**NumPy:** <https://numpy.org/>

**Chatterbot:** <https://chatterbot.readthedocs.io/en/stable/>

**Django:** <https://www.djangoproject.com/>

**PostgreSQL:** <https://www.postgresql.org/>, <https://www.pgadmin.org/>

**General:** <https://stackoverflow.com/>