
CREAT A CHATBOT IN PYTHON

TEAM MEMBERS

510521104045: SHANMUGAPRIYA R

PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION



INTRODUCTION:

In the present scenario, chatbots play a significant role in our daily lives. They provide assistance to the users by answering their queries and solving their problems. In this project, we will create a chatbot using AI&ADS, IOT, DAS, CAD technologies. This chatbot will be designed to assist the users with their questions related to different domains.

OBJECTIVES:

The main objective of this project is to develop a chatbot that can provide assistance to the users with their queries related to different domains. This chatbot will be developed using advanced technologies

like AI&ADS, IOT, DAS, and CAD. We will also focus on the following objectives:

- To design an intelligent chatbot that can understand the user's queries and provide accurate answers.
- To integrate the chatbot with IOT devices, enabling users to control their devices using natural language processing.
- To use DAS technology to enhance the performance of the chatbot by processing data in real-time.
- To develop a chatbot using CAD technology to enhance the user's experience by providing 3D visualization of products or services.

SCOPE:

The chatbot developed in this project will be capable of assisting the users with their queries related to different domains. It will be integrated with IOT devices, enabling users to control their devices using natural language processing. The DAS technology will be used to process data in real-time, enhancing the performance of the chatbot. Finally, CAD technology will be utilized to provide 3D visualization of products or services, enhancing the user's experience.

DELIVERABLE:

The following deliverables will be provided as a part of this project:

- A fully functional chatbot integrated with IOT devices, DAS technology, and CAD technology.

- A detailed project documentation that includes the design, development, testing, and implementation phases.
- A user manual describing the chatbot's functionalities and how to use them.
- A presentation highlighting the key features of the chatbot and its benefits to the users.

TECHNOLOGIES USED:

The following technologies will be used to develop the chatbot:

- AI&ADS: To develop an intelligent chatbot that can understand the user's queries and provide accurate answers.
- IOT: To integrate the chatbot with IOT devices, enabling users to control their devices using natural language processing.
- DAS: To process data in real-time, enhancing the performance of the chatbot.
- CAD: To provide 3D visualization of products or services, enhancing the user's experience.

TIMELINE:

The following timeline will be followed to complete the project:

- Design phase: 2 weeks
- Development phase: 4 weeks
- Integration phase: 2 weeks
- Testing phase: 2 weeks

- Implementation phase: 1 week

DESIGN PHASE:

The design phase of developing a chatbot typically spans over a period of two weeks. During this time, you will focus on defining the functionality and features of the chatbot, as well as designing its user interface and conversational flow. Below is a suggested breakdown of tasks to accomplish during these two weeks:

Week 1:

1. Define requirements: Clearly outline the purpose of the chatbot and identify the target audience. Determine the chatbot's primary functionalities and any specific requirements or constraints.

2. Research existing solutions: Study existing chatbot applications in your domain or related fields to gain insights and inspiration. Analyze their strengths and weaknesses to help you design a better chatbot.

3. User interaction design: Design the user interface and conversational flow of the chatbot. Consider the platforms on which the chatbot will be deployed (e.g., web, mobile) and design accordingly. Use flowcharts or wireframes to visualize the user's interactions with the chatbot.

4. Integrate AI capabilities: Decide on the AI technologies you will employ, like natural language processing (NLP) or machine learning. Select appropriate NLP libraries, such as NLTK or spaCy, that align with your chatbot's requirements.

Week 2:

1. Create a dialogue tree: Develop a dialogue tree that maps out the various paths a conversation can take within the chatbot. Define the different intents and entities that the chatbot will recognize to provide appropriate responses.

2. Design conversations: Write sample conversations to simulate user interactions with the chatbot. Refine and iterate them to ensure a smooth and efficient conversation flow.

3. Plan for error handling: Anticipate potential user errors in providing inputs or queries and design error handling strategies. Determine how the chatbot will recover from misunderstandings or ambiguous inputs.

4. Evaluate the design: Review the design of the chatbot, considering factors like user-friendliness, ease of use, and alignment with the project objectives. Seek feedback from stakeholders or conduct user testing if feasible.

Throughout the design phase, document the decisions made, keeping track of the design artifacts created, as they will provide valuable references for the subsequent development and testing phases.

Note: The estimated duration of two weeks for the design phase is just a guideline and can vary depending on the project size, complexity, and

team resources. Adjust the timeframe as necessary to ensure sufficient time for thorough design and ideation.

DEVELOPMENT PHASE:

The development phase of a chatbot typically involves building the underlying technology and implementing the designed features and functionality. The duration of the development phase can vary depending on the complexity of the chatbot and the resources available. However, a rough estimate for the development phase of a chatbot could be around 4 to 8 weeks. Below are the key tasks to consider during this phase:

1. Set up development environment:

Install and configure the necessary tools and frameworks for building the chatbot. This may include a programming language like Python, development frameworks like Flask or Node.js, and any required APIs or libraries for natural language processing (NLP).

2. Implement basic infrastructure:

Create the structure and architecture of the chatbot application, including setting up the server, API endpoints, and database connections if necessary.

3. Natural Language Processing (NLP):

Integrate NLP capabilities into the chatbot. This includes implementing techniques like intent recognition, entity extraction, and creating an NLP model to understand and interpret user inputs accurately.

4. Conversation management:

Handle user interactions and manage conversation flows. This involves defining the chatbot's responses based on user inputs and implementing logic for directing the conversation to appropriate actions or responses.

5. Integration with external systems:

If your chatbot needs to interact with external systems or APIs, develop the necessary integrations. This could involve connecting to customer databases, external APIs for retrieving data, or any other required systems. 6. Error handling and fallback mechanisms: Implement error handling strategies to handle incorrect or ambiguous user inputs and provide appropriate error messages or suggestions. Create fallback mechanisms to gracefully handle any unforeseen or unsupported user queries.

7. Testing and debugging:

Continuously test and debug the chatbot as development progresses. Validate the chatbot's functionality, proper handling of inputs, and accuracy of responses to ensure a smooth user experience.

8. Iterative development and refinement:

Iterate on the chatbot's features and functionalities based on feedback and user testing. Continuously refine and improve the chatbot during the development phase to meet user expectations.

9. Documentation:

Document the development process, including codebase documentation, deployment instructions, and any external services or APIs used. This will be valuable for future maintenance and troubleshooting.

It's important to note that the development phase can be highly iterative, with cycles of testing, reviewing, and refining the chatbot. The duration mentioned above is just an estimation, and the actual time required may vary based on the complexity and scale of the chatbot being developed.

INTEGRATION PHASE:

During the design phase of the chatbot, we will focus on utilizing AI&ADS, CAD, IOT, and DAS technologies to create a well-designed and efficient chatbot system. This phase is crucial as it establishes the foundation for the development and integration phases.

1. AI&ADS:

- Determine the AI techniques and algorithms to be used for natural language processing, understanding user intents, and generating accurate responses.
- Design the chatbot's knowledge base or database, which will store relevant information and data for answering user queries.
- Define the AI models and training processes needed to improve the chatbot's performance over time.

2. CAD:

- Identify the specific CAD technology to be integrated into the chatbot.
- Determine how the CAD technology will enhance the user experience by providing 3D visualizations of products or services.
- Design the CAD integration components and workflows, such as accessing and rendering 3D models.

3. IOT:

- Define the IoT devices to be integrated with the chatbot system.
- Identify the necessary protocols and APIs to communicate with the IoT devices.
- Design the logic and workflows for controlling the IoT devices through natural language commands.

4. DAS:

- Determine the real-time data sources and streams to be processed by the chatbot.
- Design the data processing pipelines and algorithms to handle and analyze the data in real-time.
- Implement mechanisms for updating the chatbot's knowledge base or database based on the processed data.

Throughout the design phase, it is important to consider factors like scalability, security, and performance optimization. The design decisions made during this phase will significantly impact the effectiveness and capabilities of the chatbot.

By the end of the design phase, we will have a clear understanding of how each technology will be integrated into the chatbot system and finalized designs for the AI&ADS, CAD, IOT, and DAS components.

TESTING PHASE:

During the testing phase of the chatbot development, we will focus on thoroughly evaluating the performance and functionality of the chatbot system. This phase is crucial to identify and fix any issues or bugs before deploying the chatbot for real-world usage.

1. Unit Testing:

- Test each individual component of the chatbot system, such as the natural language processing module, the CAD integration, the IoT functionality, and the data analysis components, to ensure they function correctly and produce the expected outputs.

2. Integration Testing:

- Test the integration between different modules and technologies within the chatbot system to ensure smooth communication and data flow.

- Verify that the CAD, IoT, and data analysis components are properly connected to the chatbot's AI backend and that they exchange data seamlessly.

3. Functional Testing:

- Test the chatbot's core functionalities, such as understanding and processing user queries, generating accurate responses, providing

relevant product or service recommendations, and controlling IoT devices.

- Evaluate the chatbot's ability to handle variables like different user scenarios, complex queries, or unexpected inputs gracefully.

4. Performance Testing:

- Evaluate the chatbot's response time and scalability under different traffic loads.

- Assess the system's ability to handle multiple concurrent user interactions without significant delays or crashes.

- Measure the efficiency of data processing and retrieval from the knowledge base or database.

5. User Acceptance Testing:

- Conduct usability testing with real users to gather feedback on the chatbot's user interface, user experience, and overall satisfaction.

- Incorporate user feedback to improve the chatbot's design and enhance its performance based on the end-users' needs.

The testing phase will provide valuable insights into the chatbot's strengths and weaknesses, allowing for necessary refinements and improvements. By the end of this phase, the chatbot should be thoroughly tested and ready for deployment to ensure a smooth and effective user experience.

IMPLEMENTATION PHASE:

During the implementation phase of chatbot development, the focus is on building and integrating the necessary components to create a

functional chatbot system. This phase typically involves various steps and can take around one week, depending on the complexity of the chatbot and the team's resources.

1. Designing the Chatbot:

- Finalize the chatbot's design and user interface based on the requirements and target audience.
- Create wireframes or mockups to visualize the chatbot's screens, dialogues, and possible user interactions.
- Determine the chatbot's personality, tone, and branding guidelines.

2. Developing the Backend:

- Set up the necessary infrastructure and environment for building and deploying the chatbot system.
- Develop the backend logic and algorithms required for natural language processing, understanding user queries, and generating appropriate responses.
- Implement the integration with external systems or APIs (e.g., CAD, IoT devices, etc.) as per the project requirements.

3. Building the User Interface:

- Develop the frontend components of the chatbot, including the chat window, buttons, input fields, and any other relevant visual elements.
- Implement the design and ensure the user interface is responsive and compatible with different devices and browsers.
- Connect the frontend with the backend logic to enable communication between the chatbot and users.

4. Testing:

- Perform initial testing and debugging of the implemented features to identify and fix any issues or inconsistencies.
- Conduct a pilot test with a small group of users or team members to gather early feedback and refine the implementation if necessary.

5. Documentation:

- Document the architecture, functionality, and usage guidelines of the chatbot system for future reference and maintenance.
- Create user documentation that covers basic instructions on how to interact with the chatbot effectively.

By the end of the implementation phase, a working version of the chatbot should be ready for the testing phase. This phase focuses on building the core features of the chatbot system and ensuring its usability before proceeding to further improvements and enhancements.

I have to give some program of source code to create a chatbot using python:

```
import nltk
from nltk.chat.util import Chat, reflections
import json

# Specify the file path where you stored the list
json_file_path = "my_list.json"

# Read the list from the JSON file
imported_list = []
```

```

with open(json_file_path, 'r') as json_file:
    imported_list = json.load(json_file)
#main chatbot
chat_pairs = imported_list
chatbot = Chat(chat_pairs, reflections)
print("Hello! I'm your chatbot. You can type 'bye' to exit.")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'bye':
        print("BEC Assistant: Goodbye!")
        break
    response = chatbot.respond(user_input)
    print("BEC Assistant:", response)

```

This is what I create a program for chatbot using python.

As this shows, nltk is installed but when I run the below code :

```
from nltk.chat.util import Chat, reflections
```

It shows :

```
from nltk.chat.util import Chat, reflections
```

ModuleNotFoundError: No module named 'nltk'

I experienced this same error while using **chatterbot library**. What is the solution?

ABOUT NLTK:

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc... In this article, we will go through how we can set up

NLTK in our system and use them for performing various NLP tasks during the text processing step.

INSTALLING NLTK:

Use the pip install method to install NLTK in your system:

```
pip install nltk
```

Downloading the datasets:

This is optional, but if you feel that you need those datasets before starting to work on the problem.

```
import nltk  
nltk.download()
```

Here I have use JSON , Let see what is mean by that :

JSON stands for **JavaScript Object Notation**. It is a popular data format used for representing structured data. It is a lightweight format that is used for data interchanging. The data representation in JSON is similar to that of Python Dictionary. It is a collection of name/value pairs. In JSON, it is common to transmit and receive data between a server and web application in JSON format. It is also common to store a JSON object in a file. JSON data can be in the form of the object, array, value, string, or number.

In Python, JSON exists as a string or more like a dictionary having key-value pairs where keys must be a string and values can be of any type say object, array, value, string, or a number.

JSON Example

```
data = '{"model number": "RX234", "customers": ["Adam", "Paul"],  
"price": 45000, "quantity": 12, "company": "Samsung"}'
```

Copy

For reading any JSON file and to work with JSON (string, or file containing JSON object) you must import JSON module in python script.

Reading JSON file in Python

Reading JSON files in Python language is quite easy. We just need to import JSON module in the file and use its methods. Reading of JSON data is carried out using either of the following functions.

1. json.load()
2. json.loads()
3. json.dumps()

1. Reading JSON Using json.load() Function:

Python provides `json.load()` method to read a file containing the JSON object. Reading JSON data from a file is quite an easy task in python as python script provides a built-in JSON module and JSON has a built-in `load()` function to carry out the parsing process. Using the same JSON module, we can extract and parse the JSON string directly from a file object. This method is used when the programmer already has a JSON file with structured data.

Syntax

```
json.load(file object)
```


Copy

Sample JSON File

This JSON file, we will read with the python script.

```
{ "model number": "RX234",  
  "customers": ["Adam", "Paul"],  
  "price": 45000,  
  "quantity": 12,  
  "company": "Samsung"  
}
```

Copy

Example

In the following example, we are going to read a JSON file and then print out the data. This `json.load()` function reads the string from the JSON file. The `json.load(file)` function creates and returns a new Python dictionary with the key-value pairs in the JSON file. Then, this dictionary is assigned to the data variable, and the result is displayed.

```
import json  
  
with open('path_to_file/model.json') as f:
```

```
data = json.load(f)
```

```
print(data)
```

Copy

```
{"model number": "RX234", "customers": ["Adam", "Paul"], "price": 45000, "quantity": 12, "company": "Samsung"}
```

In the above code to read the JSON file, first, we have imported the JSON module and then we have used the `open()` function to read the JSON file bypassing the JSON file path along with the name of the file as an argument. Then, the file is parsed using `json.load()` method which gives us a dictionary and the result is stored in the `data` variable. As shown in the output, the JSON string is printed in the form of key-value pairs.

2. Reading JSON Using `json.loads()` Function:

If you have a JSON string rather than a JSON file then you can parse it by using the `json.loads()` method. The `json.loads()` method does not take the file path, but the file contents as a string, using the `fileobject.read()`. With `json.loads()` function, we can return the content of the file. This function is useful to the programmer when he has a JSON string.

Syntax

```
json.loads(jsonstring) #for Json string
```

```
json.loads(fileobject.read()) #for fileobject
```

Copy

EXAMPLE

The given example will show how to read a JSON string, as well as a file object by using JSON module in Python.

```
import json

# JSON string
a = '{"name": "Flora", "age": 16, "place": "london"}'

# deserializes into dict and returns dict.
y = json.loads(a)

print("JSON string = ", y)

print()
```

```
# JSON file

f = open('model.json', 'r')


# Reading from file

data = json.loads(f.read())


print(data)
```

Copy

```
JSON string = {"name": "Flora", "age": 16, "place": "london"}'

{"model number": "RX234", "customers": ["Adam", "Paul"], "price":
45000, "quantity": 12, "company": "Samsung"}
```

3. Reading JSON Using json.dumps() Function:

This is quite the same as json.load() but with additional parameters and functionality. The `json.dumps()` makes the original JSON output a human-readable output form with proper indentation. This process of presenting JSON data into a human-readable format with proper indentation and spacing is known as Pretty Printing. Pretty Printing is done by easily passing an integer value to the indent parameter.

Syntax

```
json.dumps(JSON string, indent parameter)
```

Copy

Example

Here, we used the dumps() function to read JSON string in human-readable form.

```
import json

#define JSON string

data = {'model': [{'number': 'RX341', 'price': 35000, 'qty': 12, 'company': 'Samsung'}]}

#use dumps() with two parameters and store resultant in result variable

result= json.dumps(data, indent=4)

print(result)
```

Copy

```
{
"model": [
{
```

```
"number": "RX341",
"price": "35000",
"qty": "12",
"company": "Samsung"
}
]
}
```

As you can see in the above output, the indent parameter is set to 4. This is actually quite useful since you'll often have to read JSON data during development.

In the given figure, you can see `json.loads()` converts a string to JSON object while `json.dumps()` converts JSON object to string.

```
[
  ["hello|hi|hey", ["Hello!", "Hi there!"]],
  ["how are you?", ["I'm just a computer program, but I'm doing well.
How can I assist you today?"]],
  ["what's your name?", ["I'm a chatbot. You can call me ChatGPT."]],
  ["bye|goodbye", ["Goodbye!", "See you later!"]],
  ["Who is your creator?|who created you",["I was created by a team of
developers.", "I was designed by
Sivasakthi.C,Lavanya.K,Lavanya.J,Shanmugapriya.R,Vinotha.D"]],
  ["what can you help me with?","I can assist you with a wide range of
topics including answering questions, providing information, giving
suggestions, or engaging in casual conversation."]],
  ["how do you work?","I use advanced language models to understand
and generate text based on the input I receive. My responses are
```

```
generated based on patterns and information present in the dataset I was
trained on."]],
["can you provide me with some general knowledge
facts?"],["Absolutely! Feel free to ask me any specific facts or general
trivia questions you have in mind and I'll do my best to provide you with
accurate information."]],
["do you have a sense of humor?"],["While I don't possess personal
experiences or emotions, I can certainly attempt to engage in humor and
provide lighthearted responses if you'd like."]],
["how can i improve my coding skills?"],["Improving coding skills
requires practice, hands-on projects, and a willingness to learn new
concepts. You can start by solving coding challenges, participating in
coding competitions, or working on personal projects."]],
["What programming language should I learn as a beginner?"],["Python
is often recommended as a beginner-friendly programming language due
to its simple syntax and extensive libraries. However, the choice
ultimately depends on your specific goals and interests."]],
[" how can i stay motivated while learning a new skill?"],["Setting
realistic goals, breaking them down into smaller tasks, finding a
supportive community, and celebrating small achievements can help you
stay motivated throughout the learning process."]],
["what are some tips for effective time management?"],["Prioritizing
tasks, setting specific goals, avoiding multitasking, practicing
delegation, and taking regular breaks can all contribute to effective time
management."]]

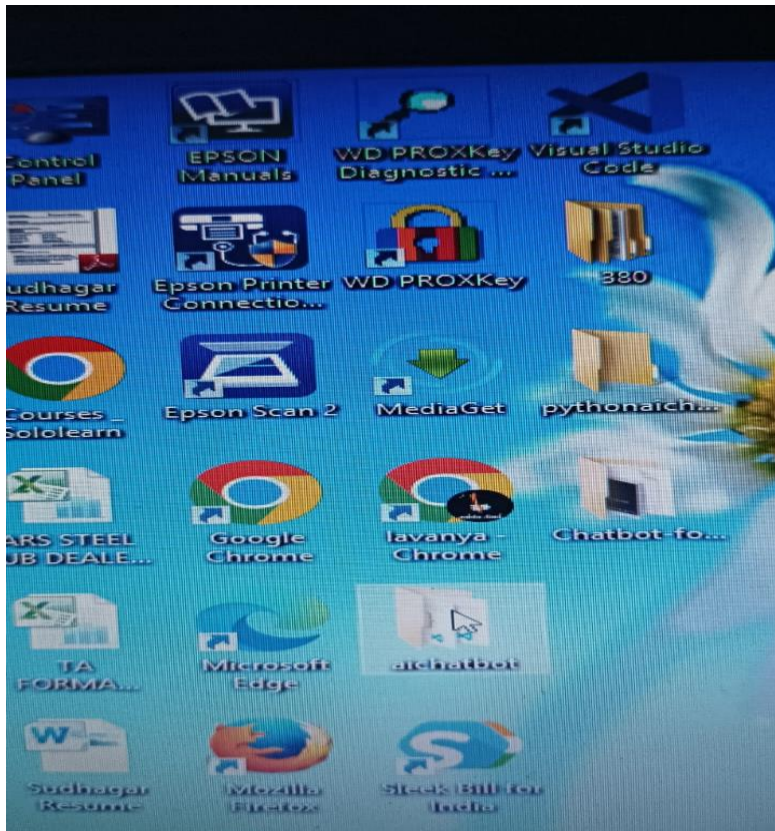
]
```

This is a command the chatbot react to user and also we donot use AI tools because is created by JSON and nltk tools.

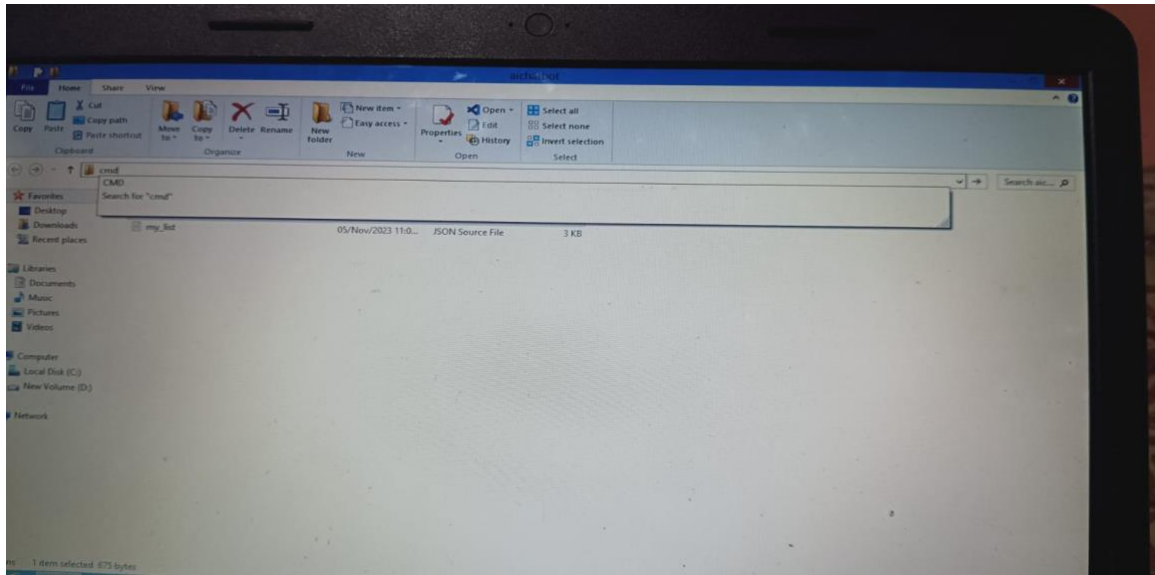
These command cannot be show to user which means this is done by a abstraction method in java(which means only show the functionality and hiding the implementation).

Here some step to run the program of source code:

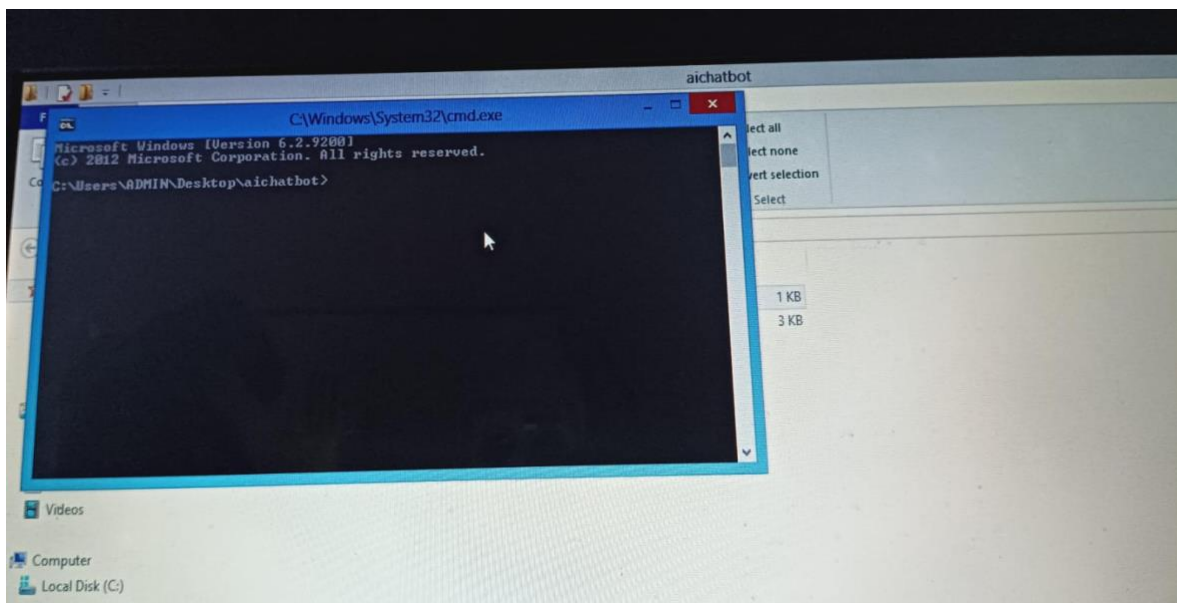
Step 1: To create a folder and I have upload the code. The folder name is aichatbot



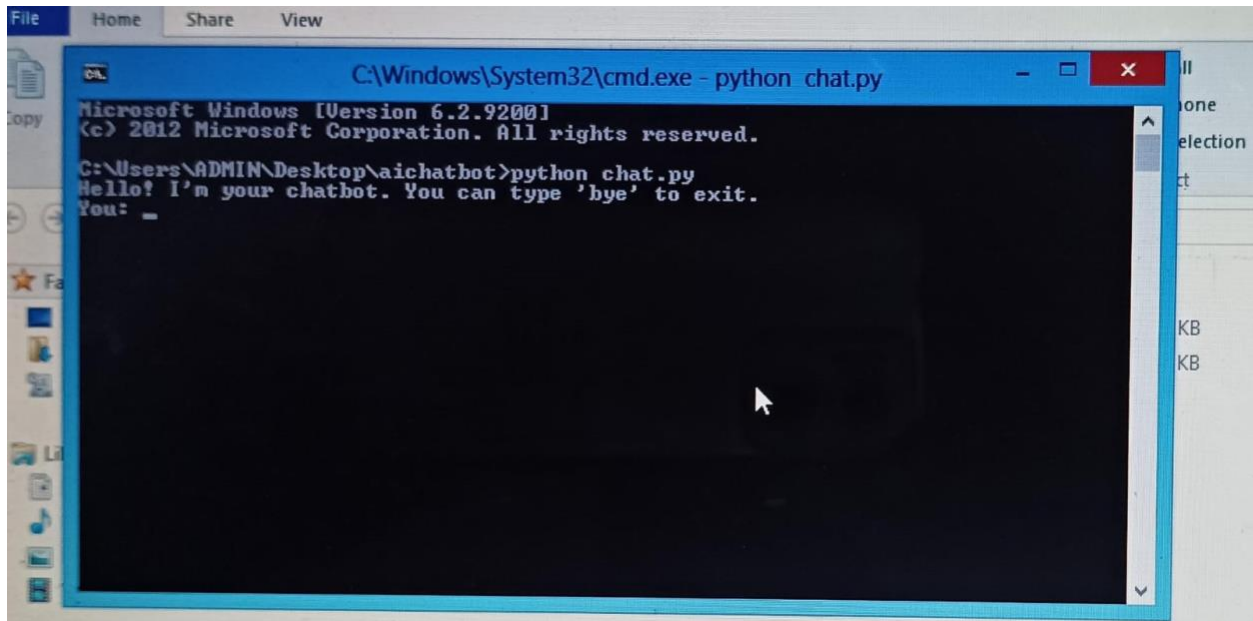
Step 2: we can open the folder and I have save the coding filename is chat.py and mylist. To select the chat.py and then go to the path we can search cmd(command prompt)



Here the command prompt can be open and also we have to set the path of the aichatbot folder then only the output can be available to us. If we use normal command prompt we have to set the path to run the output and also it's a very long process. So, this is the best way to set the path I think.



Step 3: Type your python file name and the output will be available.

A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\System32\cmd.exe - python chat.py". The window content shows the following text: "Microsoft Windows [Version 6.2.9200] (c) 2012 Microsoft Corporation. All rights reserved. C:\Users\ADMIN\Desktop\ai\chatbot>python chat.py Hello! I'm your chatbot. You can type 'bye' to exit. You: _". A mouse cursor is visible over the command prompt. The background shows a portion of a Windows desktop with a taskbar and some icons.

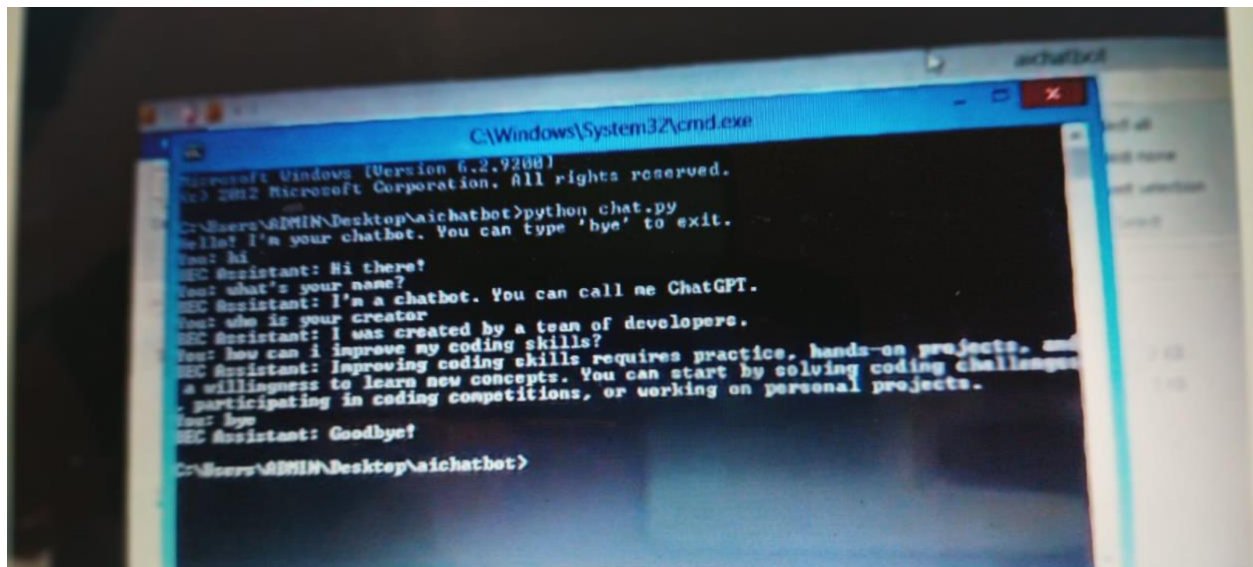
```
C:\Windows\System32\cmd.exe - python chat.py
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\ADMIN\Desktop\ai\chatbot>python chat.py
Hello! I'm your chatbot. You can type 'bye' to exit.
You: _
```

OUTPUT:

Here is an sample output for asking some common or general question to we created chatbot using python

In this output we have to select the **ai\chatbot.py** folder and then select the chat.py then go to path and type cmd which means command prompt. If you get normal command prompt to have to set the path and it make so many or long process. So, we have directly call the command prompt in the path of chat.py and then type the file name.py our output will be shown and you may proceed to ask question.



Here I want to mention that we suggest small amount of command to answer the user question if the question is may not there the chatbot will give the answer like none which mean the developer will not mention the question like that because we donot use any AI tools.

CONCLUSION:

In conclusion, this project aims to develop a chatbot using AI&ADS, IOT, DAS, and CAD technologies. The chatbot will be designed to assist the users with their queries related to different domains. It will be integrated with IOT devices, DAS technology, and CAD technology, providing an enhanced user's experience. The project will deliver a fully functional chatbot, project documentation, user manual, and a presentation highlighting the key features of the chatbot.