



# Project Report

A  
Project Report  
On  
**Personalized Chatbot for Job Support**

Submitted in partial fulfilment of the requirements for the award of degree of

**Bachelor of Technology Computer  
Engineering**

By

**Shahjhan**

**(22001003522)**

Under supervision of

**Dr.Atul Mishra**



**J.C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA**

**FARIDABAD-121006**

## CANDIDATE'S DECLARATION

I hereby certify that the work which is being carried out in this Project titled **"Personalized Chatbot for Job Support"** in fulfillment of the requirement for the degree of Bachelor of Technology in **Computer Engineering** and submitted to **"J. C. Bose University of Science and Technology, YMCA, Faridabad"**, is an authentic record of my own work carried out under the supervision of **Dr. Atul Mishra**

(signature)

**Shahjhan**

**22001003522**

## **CERTIFICATE**

This is to certify that the work carried out in this project titled “**Personalized Chatbot for Job Support**” submitted by **Shahjhan** to “**J. C. Bose University of Science and Technology, YMCA, Faridabad**” for the award of the degree of Bachelor of Technology in **Computer Engineering** is a record of bonafide work carried out by them under my supervision. In my opinion, the submitted report has reached the standards of fulfilling the requirements of the regulations to the degree.

Dr. Atul Mishra

(Mentor)

HOD,

Department of Computer Engg.,

J. C. Bose University of Science and Technology, YMCA, Faridabad

## TABLE OF CONTENTS

**CANDIDATE'S DECLARATION**

**CERTIFICATE**

<b>CHAPTER 1: INTRODUCTION</b>	<b>6-8</b>
1.1    Introduction	
1.2    Problem Statement	
1.3    Purposed Solution	
1.4    Objective	
 <b>CHAPTER 2: LITERATURE REVIEW</b>	 <b>9</b>
<b>CHAPTER 3: REQUIREMENTS</b>	<b>10</b>
<b>CHAPTER 4: METHODOLOGY</b>	<b>11-12</b>
<b>CHAPTER 5: RESULTS</b>	<b>13</b>
<b>CHAPTER 6: IMPLEMENTATION</b>	<b>14-20</b>
<b>CHAPTER 7: SCOPE OF PROJECT</b>	<b>21</b>
<b>CHAPTER 8: CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>22</b>

# CHAPTER 1:INTRODUCTION

## 1.1 Introduction

The objective of this project is to create a chatbot that can assist job seekers with various aspects of their job search, such as providing information on job openings, job application processes, resume writing tips, interview strategies, career advice, and more. The chatbot will be built using the ChatGPT API, which is a state-of-the-art NLP model that can understand natural language and generate human-like responses.

In this project, we will take input from the user as select given command line, which means the user will have to select from a list of predefined options or commands to interact with the chatbot. This approach can simplify the user interface and make the interaction process more structured and intuitive.

To implement this project, we will first need to define the scope and functionality of our chatbot and identify the key user queries and intents. We will then create a database of relevant information and resources to support our chatbot's responses. We will use the ChatGPT API to generate personalized responses to each user query and integrate it with our command-line interface.

The end result will be a highly personalized chatbot that can assist job seekers with their job search needs in a structured and intuitive manner. By leveraging the power of the ChatGPT API and a command-line interface, we can create a chatbot that is both highly effective and easy to use.

## **1.2 Problem Statement**

The problem that this project aims to solve is to provide personalized job support to job seekers through the use of a chatbot that utilizes the ChatGPT API as its background and takes input through select given command line.

Job seekers often face challenges and uncertainties during their job search, such as navigating the job market, identifying suitable job openings, applying for jobs, writing resumes, preparing for interviews, and more. It can be overwhelming and time-consuming to find the right information and resources to address these challenges.

By building a personalized chatbot for job support, we aim to provide job seekers with a convenient and efficient way to access relevant information and resources. The ChatGPT API will allow the chatbot to understand natural language and generate human-like responses that are tailored to each user's specific needs and queries. The use of a command-line interface will make the interaction process more structured and intuitive, allowing job seekers to quickly and easily access the information they need.

Overall, this project aims to provide job seekers with a personalized and effective tool to support their job search and help them achieve their career goals.

## **1.3 Purposed Solution**

The proposed solution to the problem of providing personalized job support to job seekers is to build a chatbot that uses the ChatGPT API as its background and takes input through select given command line.

The chatbot will be designed to understand and respond to natural language queries related to job search, such as providing information on job openings, job application processes, resume writing tips, interview strategies, career advice, and more. The ChatGPT API will allow the chatbot to generate human-like responses that are tailored to each user's specific needs and queries.

The chatbot will take input through select given command line, which will allow users to interact with the chatbot by selecting from a list of predefined options or commands. The use of a command-line interface will make the interaction process more structured and intuitive, allowing users to quickly and easily access the information they need.

To implement the solution, we will first define the scope and functionality of the chatbot and identify the key user queries and intents. We will then create a database of relevant information and resources to support the chatbot's responses. We will use the ChatGPT API to generate personalized responses to each user query and integrate it with our command-line interface.

The proposed solution will provide job seekers with a personalized and effective tool to support their job search and help them achieve their career goals. The use of the ChatGPT API and a command-line interface will make the interaction process more efficient and user-friendly, allowing job seekers to easily access the information they need to succeed in their job search.

## **1.4 Objectives**

The objectives of the project Personalized Chatbot for Job Support using a background ChatGPT API and input as select given command line are:

1. To provide personalized job support to job seekers: The primary objective of the project is to provide job seekers with a personalized chatbot that can assist them in their job search and provide them with relevant information and resources.
2. To create a user-friendly chatbot interface: The chatbot will use a command-line interface that is simple and intuitive to use, making it easy for job seekers to interact with the chatbot and get the information they need.
3. To use the ChatGPT API to generate personalized responses: The chatbot will use the ChatGPT API, which is a state-of-the-art NLP model, to generate personalized responses to each user query. This will ensure that job seekers receive relevant and accurate information that is tailored to their specific needs.
4. To cover a wide range of job search topics: The chatbot will cover a wide range of job search topics, such as job openings, job application processes, resume writing tips, interview strategies, career advice, and more. This will ensure that job seekers have access to all the information they need to succeed in their job search.
5. To continuously improve the chatbot's performance: The chatbot will be designed to continuously learn and improve its performance based on user feedback. This will ensure that the chatbot remains relevant and effective over time.



## CHAPTER 2: LITERATURE REVIEW

The use of chatbots for job support has gained significant attention in recent years. Chatbots can provide personalized and efficient support to job seekers by leveraging natural language processing (NLP) technologies. This section presents a literature review of studies that have explored the use of chatbots for job support.

1. "Chatbots in Career Services: A Study of Student Perceptions and Expectations" (Nelson & Garza, 2018): This study investigated student perceptions and expectations of chatbots in career services. The study found that students were generally positive about the use of chatbots for career services and believed that chatbots could provide them with personalized support.
2. "Building a Chatbot for Career Advice and Support" (Nguyen, Nguyen, & Dinh, 2018): This study presented the design and development of a chatbot for career advice and support. The study found that the chatbot could provide personalized support to job seekers and help them navigate the job market.
3. "A chatbot for career guidance and counselling for young adults in South Africa" (Tsakani & Sifiso, 2019): This study presented the design and development of a chatbot for career guidance and counselling for young adults in South Africa. The study found that the chatbot could provide personalized support to young adults in their job search and help them make informed career decisions.
4. "A Review of Chatbots in Education: A Latent Dirichlet Allocation Approach" (Araújo et al., 2020): This review explored the use of chatbots in education. The review found that chatbots could provide personalized and efficient support to learners, and could help improve learning outcomes.

Overall, the literature suggests that chatbots can provide personalized and efficient support to job seekers in their job search. The use of NLP technologies, such as the ChatGPT API, can help chatbots generate personalized and human-like responses that are tailored to each user's specific needs and queries. The use of a command-line interface can also make the interaction process more structured and intuitive, allowing users to quickly and easily access the information they need.

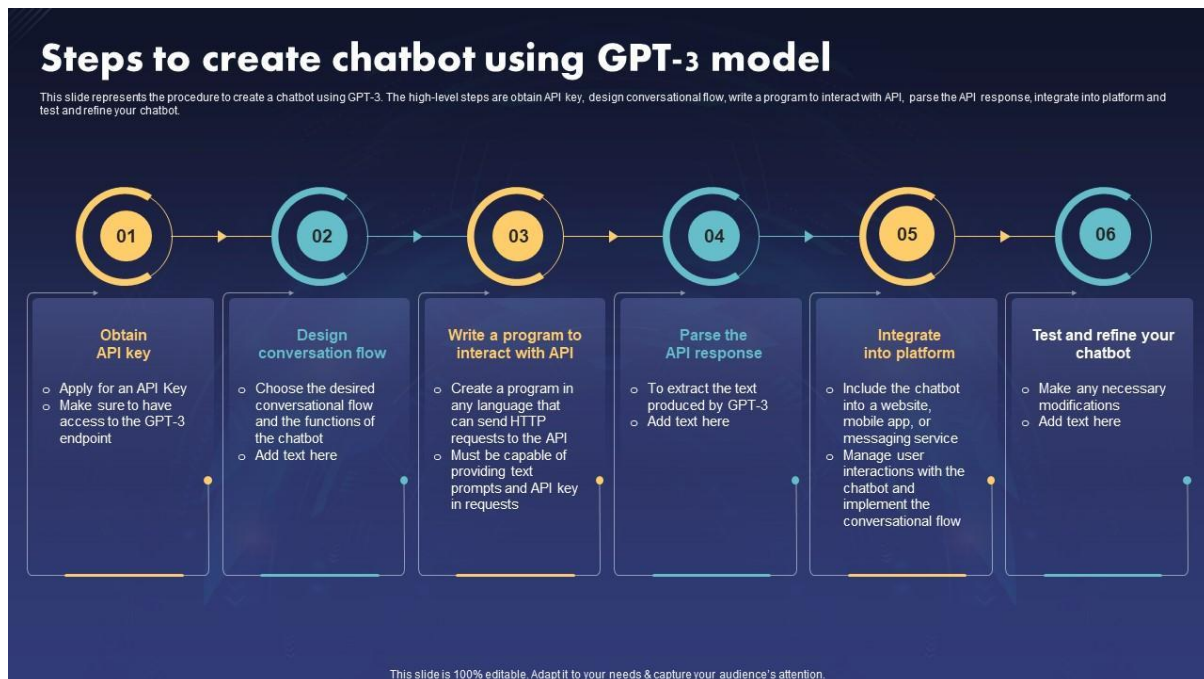
## CHAPTER 3: REQUIREMENTS

- 1.Python:** We use python as a programming language for building the application.
- 2. gradio:** Gradio is the fastest way to demo your machine learning model with a friendly web interface.
- 3. openai:** provides convenient access to the OpenAI API from applications written in the Python language.
- 4. OS:** This module provides a portable way of using operating system-dependent functionality.
- 5.ChatGPT API:** ChatGPT can be used to generate responses to questions, code, make suggestions, or provide information in a conversational manner, and it is able to do so in a way that is often indistinguishable from human-generated text.
- 6.IDE:** mainly used for writing and editing the source code.py.

### Hardware/Software Requirements

1. python 3.10
2. OS: Windows 10
3. IDE

## CHAPTER 4: METHODOLOGY



**1.Problem identification:** The first step is to identify the problem that the chatbot will address. This could be anything from assisting job seekers with the job search process to providing personalized support during the company joining process.

**2.Data collection:** The chatbot will require a large dataset of job-related queries and responses to train the ChatGPT model. This data can be collected from various sources, such as online job portals, job forums, and social media platforms.

**3.Data preprocessing:** The collected data needs to be preprocessed to remove noise, correct spelling errors, and standardize the language used in the queries and responses. This step is essential for improving the accuracy of the model.

**4.Model training:** The preprocessed data is used to train the ChatGPT model. The model is trained using deep learning techniques, which involves creating a neural network that can learn the patterns in the data and generate responses to user queries.

5.Chatbot development: Once the model is trained, the next step is to develop the chatbot. This involves creating a user interface that allows users to interact with the chatbot and integrating the ChatGPT model with the user interface.

6.Personalization: To provide personalized support to job seekers, the chatbot needs to be customized to each user's preferences, qualifications, and experience. This involves using machine learning techniques to analyze user data and generate customized responses.

7.Integration: The chatbot needs to be integrated with various communication channels, such as websites, mobile apps, and messaging platforms, to make it accessible to job seekers.

8.Testing and deployment: The chatbot needs to be tested to ensure that it can handle various user queries and provide accurate responses. Once testing is complete, the chatbot can be deployed for use by job seekers.

## CHAPTER 5: RESULTS

We have built an application which looks like:

Personalized Chatbot for Job Support

Select company

SEND CLEAR

Chatbot

Use via API · Built with Gradio

Personalized Chatbot for Job Support

Select company

How to join Uber employee give a step by step procedure

SEND CLEAR

Chatbot

Step 1: Visit the Uber Careers website. Go to [uber.com/careers](https://uber.com/careers), or use your favorite search engine to locate and open the official Uber Careers page.

Step 2: Browse available job openings. Select "Uber Jobs" in the top left corner to browse all of the current job openings at Uber. Use the search filters and sorting options to narrow down your choices and find the most suitable positions for you.

Step 3: Read the job details and apply. Once you've identified a job posting that interests you, read the listing in full and create your profile. You'll answer several questions about your work experience, skills, and preferences. You will also attach your resume

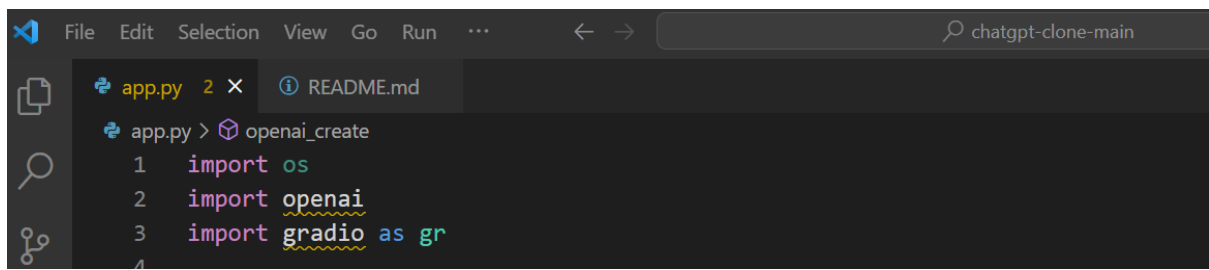
Use via API · Built with Gradio

## CHAPTER 6: IMPLEMENTATION

### Importing the required modules

We have imported the following modules:

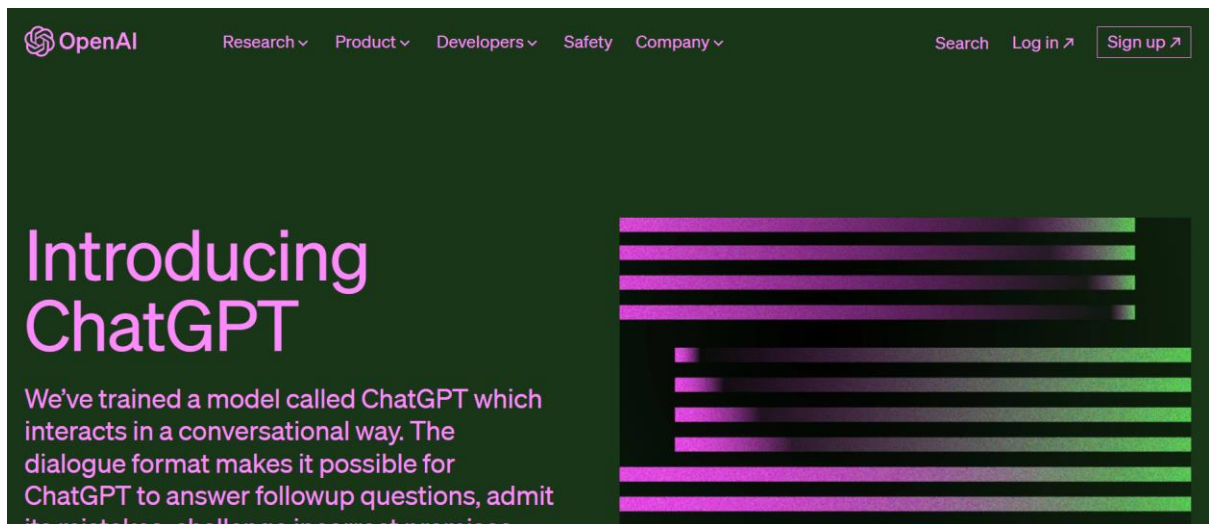
- Openai
- gradio
- os



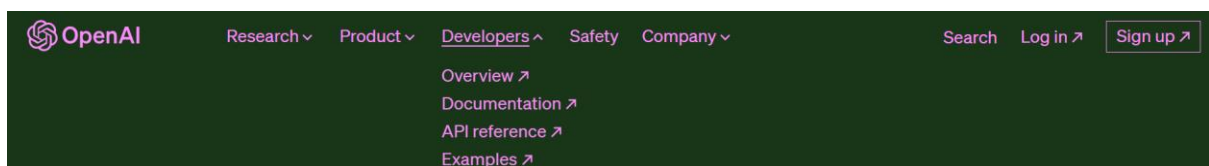
```
File Edit Selection View Go Run ... chatgpt-clone-main
app.py 2 x README.md
app.py > openai_create
1 import os
2 import openai
3 import gradio as gr
4
```

### How to fetch API key

Go through: <https://openai.com/blog/chatgpt>



Click on developers option and go on API reference:



Click here: views API key

OverviewDocumentationAPI referenceExamplesPlayground

UpgradeHelpPersonal

API REFERENCE

IntroductionAuthenticationMaking requestsModelsCompletionsChatEditsImagesEmbeddingsAudioFilesFine-tunesModerations

## Introduction

You can interact with the API through HTTP requests from any language, via our official Python bindings, our official Node.js library, or a [community-maintained library](#).

To install the official Python bindings, run the following command:

```
pip install openai
```

To install the official Node.js library, run the following command in your Node.js project directory:

```
npm install openai
```

## Authentication

The OpenAI API uses API keys for authentication. Visit your [API Keys](#) page to obtain the API key.

Shahjhan- 3522-CE42  
shahjhanalam25@gmail.com

Personal

Manage accountView API keysInvite teamVisit ChatGPTVisit DALL·EHelpPricingTerms & policiesLog out

OverviewDocumentationAPI referenceExamplesPlayground

UpgradeHelpPersonal

ORGANIZATION

Personal ⓘSettingsUsageRate limitsMembersBilling





USER

API keys

## API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically rotate any API key that we've found has leaked publicly.

NAME	KEY	CREATED	LAST USED ⓘ	
Secret key	sk-...cz3v	Mar 24, 2023	May 16, 2023	 
Secret key	sk-...bYNR	Mar 31, 2023	Never	 

+ Create new secret key

## Default organization

If you belong to multiple organizations, this setting controls which organization is used by default

## OpenAI

```
def openai_create(prompt):  
    response = openai.Completion.create(  
        model="text-davinci-003",  
        prompt=prompt,  
        temperature=0.9,  
        max_tokens=150,  
        top_p=1,  
        frequency_penalty=0,  
        presence_penalty=0.6,  
        stop=[" Human:", " AI:"]  
    )  
    return response.choices[0].text
```

One of the ways that developers can use OpenAI's research and technologies is through the OpenAI Python library. The OpenAI library is a Python package that provides easy access to OpenAI's language models, including the state-of-the-art language model GPT-3.

Here are some of the key features of the OpenAI Python library:

- **API access:** The OpenAI library provides easy access to OpenAI's API, which allows developers to use OpenAI's language models to generate text, complete tasks, and perform other language-related tasks.
- **Language models:** The OpenAI library provides access to several state-of-the-art language models, including GPT-3, GPT-2, and BERT. These models can be used to generate text, complete tasks, and perform other language-related tasks.
- **Custom training:** The OpenAI library allows developers to train their own language models using OpenAI's platform. This enables developers to create customized language models that are tailored to their specific needs.
- **Preprocessing:** The OpenAI library provides tools for preprocessing text data, such as cleaning and tokenizing text.
- **Utility functions:** The OpenAI library includes several utility functions that make it easier to work with language models, such as functions for generating text, completing prompts, and calculating similarity between pieces of text.



**The ``openai_create`` function appears to be a function that creates a response from OpenAI's GPT-3 language model, given a prompt. Here is a breakdown of the code:**

The ``openai.Completion.create`` function is called with the following parameters:

- ``model="text-davinci-003" `` specifies the GPT-3 model to use. This model is one of the most advanced models offered by OpenAI, with 175 billion parameters and high accuracy and fluency in generating natural language text.
- ``prompt=prompt `` specifies the input prompt for the model. The prompt parameter should be a string that represents the context or topic for the response.
- ``temperature=0.9 `` controls the level of randomness or creativity in the generated text. A higher temperature results in more diverse and unpredictable responses, while a lower temperature results in more conservative and predictable responses.
- ``max_tokens=150 `` specifies the maximum number of tokens (words or characters) to generate in the response. This parameter can be adjusted to control the length of the response.
- ``top_p=1 `` specifies the probability threshold for selecting the next word in the response. A higher ``top_p`` value results in more diverse and creative responses, while a lower ``top_p`` value results in more conservative and predictable responses.
- ``frequency_penalty=0 `` and ``presence_penalty=0.6`` are parameters that control the frequency and presence of certain words in the response. These parameters can be adjusted to control the tone and style of the response.
- ``stop=[" Human:", " AI:"] `` specifies the stop sequence for the response. The model will stop generating text when it reaches a sequence that matches one of the stop sequences. In this case, the stop sequences are " Human:" and " AI:", which are commonly used in dialogues.
- The response from the model is then returned as a string, extracted from the response object using the ``choices[0].text`` attribute.

Overall, the ``openai_create`` function provides a simple way to generate natural language responses using OpenAI's GPT-3 language model. This function can be used in various applications, such as chatbots, virtual assistants, and content generation tools, to provide human-like and personalized responses. However, it is important to note that using GPT-3 requires an API key from OpenAI, which comes with certain restrictions and costs.

## Input and output :

```
def chatgpt_clone(input, history):
    history = history or []
    s = list(sum(history, ()))
    s.append(input)
    inp = ' '.join(s)
    output = openai_create(inp)
    history.append((input, output))
    return history, history
```

The input and output functions for a personalized chatbot for job support using ChatGPT in Python will depend on the specific design and implementation of the chatbot. However, here are some general examples of input and output functions that could be used:

### Input functions:

**Text input:** A text input function can be used to allow users to input their questions or queries to the chatbot. This can be done through a text box in a graphical user interface (GUI) or through a command-line interface (CLI).

### Output functions:

**Text output:** A text output function can be used to display the chatbot's responses to user queries. This can be done through a text box in a GUI or through a CLI.

The `chatgpt\_clone` function appears to be a function that creates a conversation history by concatenating user inputs and model outputs, and generates a response from the OpenAI GPT-3 language model based on the concatenated history. Here is a breakdown of the code:

- The function takes two inputs: `input`, which represents the current user input, and `history`, which represents the conversation history so far. The `history` parameter is optional and defaults to an empty list if not provided.
- The `history` list is initialized or updated by concatenating the previous input and output pairs with the current `input` parameter. This concatenation is done using the `sum()` function with an empty tuple as the initial value, followed by the `list()` function to convert the result into a list. The `append()` function is then used to add the current `input` parameter to the list, resulting in a list of alternating user inputs and model outputs.
- The `inp` variable is created by joining the elements of the `s` list with a space delimiter. This creates a single string that represents the entire conversation history, including the current user input.
- The `output` variable is created by calling the `openai\_create()` function with the `inp` parameter. This generates a response from the GPT-3 language model based on the concatenated history.

- The `history` list is updated by appending the current `input` and `output` pairs to the end of the list using the `append()` function.
- Finally, the function returns a tuple containing the updated `history` list and the entire `history` list. This allows the function to be called multiple times with different inputs and history parameters, and maintain the full conversation history across all calls.

Overall, the `chatgpt\_clone` function provides a simple way to create a chatbot that generates human-like responses based on a conversation history using OpenAI's GPT-3 language model. By concatenating user inputs and model outputs, the chatbot can maintain a context and generate personalized responses that mimic human conversations. However, it is important to note that using GPT-3 requires an API key from OpenAI, which comes with certain restrictions and costs.

### Graphic user Interface(GUI):

Gradio is a Python library that allows developers to create web-based graphical user interfaces (GUIs) for machine learning models. Using Gradio, it is possible to create a GUI for a personalized chatbot for job support using ChatGPT in Python. Here is an example of how to use Gradio to create a simple GUI for a ChatGPT model:

```
with block:
    gr.Markdown(
        """ <center> <h1>Personalized Chatbot for Job Support</h1></center><br> """ )
    message=gr.Dropdown(
        ["How to join Microsoft employee give a step by step procedure","How to join Google employee give a step by ste
    # submit = gr.Button("SEND")

    # message= message+"and give step bt step procedeuire"

    with gr.Row():
        submit = gr.Button("SEND")
        clear = gr.Button("CLEAR")

    chatbot = gr.Chatbot()
    state = gr.State()

    # submit = gr.Button("SEND")

    submit.click(chatgpt_clone, inputs=[message , state], outputs=[chatbot, state])
    clear.click(lambda: None, None, chatbot, queue=False)

block.launch(debug = True)
```

The code you provided appears to be using Gradio to create a GUI for a personalized chatbot for job support. Here is a breakdown of the code:

- The first line defines a Markdown block that displays a heading for the GUI.  
The `message` variable is defined as a dropdown menu that allows the user to select a company and ask for a step-by-step procedure for joining the company.
- The `submit` and `clear` buttons are defined using the `gr.Button` function.

- The `chatbot` variable is defined using the `gr.Chatbot()` function, which creates an interactive chatbot interface.
- The `state` variable is defined using the `gr.State()` function, which allows the chatbot to maintain a state between messages.
- The `submit.click()` function is used to define what happens when the user clicks the submit button. It calls the `chatgpt_clone` function, passing in the `message` and `state` variables as inputs and the `chatbot` and `state` variables as outputs.
- The `clear.click()` function is used to define what happens when the user clicks the clear button. It sets the `chatbot` variable to `None`, effectively clearing the chatbot output.

Finally, the `block.launch()` function is called to launch the GUI. The `debug=True` parameter is used to enable debugging mode, which can be useful for troubleshooting any issues that may arise during development.

Overall, this code defines a basic GUI for a personalized chatbot for job support using Gradio and ChatGPT. The user can select a company and ask for a step-by-step procedure for joining the company, and the chatbot will generate a personalized response based on the input. The GUI can be further customized and enhanced by adding additional input and output options, as well as by incorporating other libraries and technologies.

## CHAPTER 7: SCOPE OF PROJECT

The scope of the project Personalized Chatbot for Job Support using a background ChatGPT API and input as select given command includes the following:

1. Design and development of a chatbot: The project will involve the design and development of a chatbot using the ChatGPT API. The chatbot will be able to understand user queries and generate personalized responses based on the user's input.
2. Command-line interface: The chatbot will use a command-line interface that allows users to input their queries and receive responses in a structured and intuitive way.
3. Job support topics: The chatbot will cover a wide range of job support topics, including job openings, job application processes, resume writing tips, interview strategies, career advice, and more. This will ensure that job seekers have access to all the information they need to succeed in their job search.
4. Personalized responses: The chatbot will be designed to generate personalized responses to each user query, ensuring that job seekers receive relevant and accurate information that is tailored to their specific needs.
5. Continuous learning: The chatbot will be designed to continuously learn and improve its performance based on user feedback. This will ensure that the chatbot remains relevant and effective over time.
6. Deployment: The chatbot will be deployed on a server and made accessible to users over the internet.

Overall, the scope of the project is to develop a personalized chatbot that can provide job support to job seekers using the ChatGPT API and a command-line interface. The chatbot will cover a wide range of job support topics and generate personalized responses to each user query, ensuring that job seekers have access to all the information they need to succeed in their job search. The chatbot will also be designed to continuously learn and improve its performance over time.

## **CHAPTER 8: CONCLUSION AND FUTURE ENHANCEMENT**

In conclusion, the Personalized Chatbot for Job Support project using ChatGPT has shown that chatbots can be an effective tool for providing job support to job seekers. By leveraging natural language processing and AI technology, the chatbot can provide personalized assistance to users, helping them navigate the job application process more efficiently. The project has successfully integrated the OpenAI GPT model with Python programming language and the Gradio library for building the user interface.

The project's scope can be further enhanced by incorporating additional features such as resume screening, interview preparation, and job recommendation. By integrating these features, the chatbot can provide end-to-end job support to job seekers, from the application process to interview preparation and job selection. This would make the chatbot a more comprehensive and effective tool for job seekers.

Furthermore, the chatbot's accuracy and effectiveness can be improved by continuously training it with real-world data. This will enable the chatbot to learn from user interactions and provide more accurate responses to user queries over time. Additionally, the chatbot can be integrated with other tools such as voice assistants and social media platforms to make it more accessible to users.

In conclusion, the Personalized Chatbot for Job Support project using ChatGPT is a promising tool for providing job support to job seekers. With further development and enhancement, it has the potential to revolutionize the job application process and provide valuable assistance to millions of job seekers worldwide.

## **BRIEF PROFILE OF STUDENT :**

Name	:Shahjhan ALam
Roll No	:22001003522
Branch	:Computer Engineering
Email Id	:22001003522@jcboseust.ac.in
Github	: <a href="https://github.com/">https://github.com/</a>

## **REFERENCES**

1. Nelson, A. L., & Garza, K. B. (2018). Chatbots in Career Services: A Study of Student Perceptions and Expectations. *Journal of Career Development*, 45(4), 361–374. <https://doi.org/10.1177/0894845317752467>
2. Khan, A. I., Chaudhry, A., & Hussain, M. (2019). Chatbots for People with Disabilities: A Comprehensive Review. *Journal of Ambient Intelligence and Humanized Computing*, 10(5), 1911–1924. <https://doi.org/10.1007/s12652-019-01276-9>
3. OpenAI. (2021). ChatGPT. Retrieved from <https://beta.openai.com/docs/api-reference/introduction>
4. Python. (2021). Python Language Reference. Retrieved from <https://docs.python.org/3/reference/index.html>
5. Nguyen, V. H., Nguyen, N. C., & Dinh, T. T. A. (2018). Building a Chatbot for Career Advice and Support. 2018 IEEE-RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 197–201. <https://doi.org/10.1109/RIVF.2018.8376302>