**CHAT BOT USING PYTHON**

*Submitted in partial fulfilment of the degree of*

## BACHELOR OF ENGINEERING

In

## COMPUTER SCIENCE ENIGNEERING.

Panimalar Institute of Technology, Chennai.

(BATCH 2021-2025)

By

U Nithish (au211521104098)

Konduru narasimha (au211521104075)

U Nikesh kumar (au211521104096)

M Ragul (au211521104120)

J Jerome (au211521104063)

**ABSTRACT**

This project involves building a chatbot using machine learning techniques and neural networks toprovide a human-like conversational experience to users. The chatbot is trained on a large corpus of textdata to learn patterns and relationships between words and phrases, enabling it to understand andrespond to user queries accurately and efficiently. The project utilizes a neural network architecturesuch as a sequence-to-sequence (seq2seq) model or a transformer model to process and generateresponses to user queries. The chatbot is trained using reinforcement learning to improve its accuracyand relevance over time.The project involves several steps, including data collection and preprocessing,model training and evaluation, and deployment of the chatbot on various platforms such as websites,mobile apps, and messaging applications. Overall, this project has the potential to enhance customerengagement, reduce response times, and improve the overall user experience.

A chatbot with machine learning using neural networks is a type of conversational agent that uses natural language processing (NLP) and deep learning techniques to understand and respond to user queries in a human-like manner.The project would involve building a chatbot using a neural network architecture such as a sequence-to-sequence (seq2seq) model or a transformer model. The chatbot would be trained on a large corpus of text data to learn patterns and relationships between words and phrases. The training data for the chatbot could be obtained from various sources such as customer support conversations, social media interactions, and online forums. The chatbot could be trained to understand and respond to a wide range of user queries related to a specific domain or industry, such as finance, healthcare, or e-commerce.To improve the accuracy and relevance of the chatbot's responses, it could be trained using reinforcement learning, where the chatbot is rewarded for providing helpful and accurate responses to user queries.The project would involve several steps, including data collection and preprocessing, model training and evaluation, and deployment of the chatbot. The chatbot could be deployed on various platforms such as websites, mobile apps, and messaging applications.Overall, a chatbot with machine learning using neural networks has the potential to improve customer engagement, reduce response times, and enhance the overall user experience.

# CONTENTS

|  |  |  |
| --- | --- | --- |
| CHAPTER | TOPIC | PAGE NUMBER |
| Chapter 1 | **Overview**   * 1. Introduction.   2. Objective of this project.   3. Benefits of this project.   4. Importance of this project. | **4** |
| Chapter 2 | **Design Thinking**   * 1. Functionality.   2. User Interface.   3. Natural Language Processing(NLP).   4. Responses.   5. Integration.   6. Testing and improvement. | **6** |
| Chapter 3 | **Design and Implementation.**  3.1 Flowchart | **7** |
| Chapter 4 | **Requirements**   * 1. Hardware.   2. Software.   3. Language used. | **8** |
| Chapter 5 | **The Conclusion.** | **9** |

**CHAPTER – 01**

**OVERVIEW**

**1.1 INTRODUCTION**

A chatbot with machine learning using neural networks is a type of conversational agent that uses

natural language processing (NLP) and deep learning techniques to understand and respond to user

queries in a human-like manner.The project would involve building a chatbot using a neural network

architecture such as a sequence-to-sequence (seq2seq) model or a transformer model. The chatbot

would be trained on a large corpus of text data to learn patterns and relationships between words

and phrases. The training data for the chatbot could be obtained from various sources such as

customer support conversations, social media interactions, and online forums. The chatbot could be

trained to understand and respond to a wide range of user queries related to a specific domain or

industry, such as finance, healthcare, or e-commerce.To improve the accuracy and relevance of the

chatbot&#39;s responses, it could be trained using reinforcement learning, where the chatbot is

rewarded for providing helpful and accurate responses to user queries.The project would involve

several steps, including data collection and preprocessing, model training and evaluation, and

deployment of the chatbot. The chatbot could be deployed on various platforms such as websites,

mobile apps, and messaging applications.Overall, a chatbot with machine learning using neural

networks has the potential to improve customer engagement, reduce response times, and enhance

the overall user experience.

**1.2 Objective of the project**

The objective of the project is to build a chatbot using machine learning techniques and neural networks

to provide a human-like conversational experience to users. The chatbot should be able to understand

and respond to user queries accurately and efficiently, improving customer engagement, reducing

response times, and enhancing the overall user experience. The project aims to utilize a neural network

architecture such as a sequence-to-sequence (seq2seq) model or a transformer model to process and

generate responses to user queries. The chatbot should be trained using a large corpus of text data to

learn patterns and relationships between words and phrases, and should be trained using reinforcement

learning to improve its accuracy and relevance over time.The project involves several steps, including

data collection and preprocessing, model training and evaluation, and deployment of the chatbot on

various platforms such as websites, mobile apps, and messaging applications. The ultimate objective is

to build a chatbot that can effectively and efficiently assist users with their queries and provide a

seamless conversational experience.

**1.3 Benefits of the project**

The benefits of the project are numerous, and they include:

1. Improved customer engagement: The chatbot can provide a personalized conversationalexperience to users, improving customer engagement and loyalty.

2. Reduced response times: The chatbot can handle a large volume of queries simultaneously and provide quick responses to users, reducing response times and improving user satisfaction.

3. Increased efficiency: The chatbot can automate routine tasks and queries, freeing up human resources to focus on more complex tasks and improving overall efficiency.

4. Scalability: The chatbot can handle a large volume of queries simultaneously, making it scalableand cost-effective for businesses.

5. 24/7 availability: The chatbot can be deployed on various platforms and can operate 24/7,providing users with round-the-clock support and assistance.

6. Improved data collection and analysis: The chatbot can collect data on user queries, preferences,and behaviors, which can be analyzed to improve business operations and decision-making.

Overall, the benefits of the chatbot with machine learning using neural networks are significant, andit has the potential to improve customer engagement, reduce response times, increase efficiency,and provide valuable insights for businesses.

**1.4 Importance of the project :**

1. Improved Customer Service: Chatbots can enhance customer support by providing instant responses to queries,

2. Cost Efficiency: AI chatbots can handle a large volume of inquiries simultaneously, reducing the need for human agents and lowering operational costs.

3. Accessibility: Chatbots can be available 24/7, making services accessible to users at any time, enhancing convenience and accessibility.

4. Scalability: AI-powered chatbots can easily scale to handle increased workloads, making them valuable for businesses experiencing growth.

5. Data Insights: Chatbots can collect and analyze user interactions, providing valuable insights into customer preferences and behaviors.

6. Personalization: AI chatbots can tailor responses and recommendations based on user data, creating a more personalized user experience.

7. Automation: They can automate repetitive tasks, allowing human employees to focus on more complex and value-added activities.

8. Marketing and Sales: Chatbots can assist in lead generation, product recommendations, and upselling, contributing to increased revenue.

9. Learning and Improvement: AI chatbots can continuously learn from interactions, improving their performance and accuracy over time.

10. Innovation: Developing chatbots with AI involves cutting-edge technology and contributes to innovation in the field of artificial intelligence and natural language processing.

Chatbots developed using Python have gained significant importance in various fields due to their versatility and effectiveness. Here are some key reasons why chatbots built with Python are important:

Automation of Tasks: Chatbots can automate a wide range of tasks, from answering frequently asked questions to handling customer inquiries. By using Python, developers can create sophisticated chatbots that can process and analyze natural language, making automation more seamless.

* Enhanced Customer Support: Python-powered chatbots can provide 24/7 customer support, resolving queries and issues promptly. This improves customer satisfaction and reduces the need for human intervention, which can lead to cost savings for businesses.
* Cost-Effective: Developing chatbots in Python is cost-effective compared to building custom solutions from scratch. Python's extensive libraries and frameworks, such as NLTK and spaCy for natural language processing, make chatbot development more efficient.
* Scalability: Python is known for its scalability. As your business grows and the demand for your chatbot increases, you can easily scale up your chatbot by adding more resources and optimizing its performance.
* Multi-Purpose: Python-based chatbots are versatile and can be used in various industries, including e-commerce, healthcare, finance, and customer service. They can perform tasks like order tracking, appointment scheduling, and data retrieval.
* Data Integration: Python's rich ecosystem of data processing libraries allows chatbots to integrate with various data sources, databases, and APIs. This enables them to retrieve up-to-date information and provide personalized responses.
* Natural Language Processing (NLP): Python offers powerful NLP libraries like NLTK, spaCy, and Transformers, which enable chatbots to understand and generate human-like responses, improving user interactions and engagement.
* Machine Learning and AI: Python's compatibility with machine learning and AI frameworks like TensorFlow and PyTorch allows developers to incorporate advanced capabilities into chatbots, such as sentiment analysis, recommendation systems, and more.
* Analytics and Insights: Python can be used to analyze chatbot interactions and user data, helping businesses gain insights into customer preferences and behaviors, which can inform decision-making and marketing strategies.
* Cross-Platform Compatibility: Python chatbots can be deployed on various platforms, including websites, messaging apps, and voice assistants, making them accessible to a wide audience.

**CHAPTER – 02**

DESIGN THINKING

**2.1 Functionality:**

Providing Guidance:

Offering step-by-step instructions for performing tasks or procedures.

Assisting users in troubleshooting issues or problems.

Recommending products, services, or solutions based on user preferences or needs.

Offering best practices or tips related to a specific field.

1. Directing Users to Appropriate Resources:

Navigating users to specific web pages, documents, or resources.

Suggesting relevant articles, videos, or tutorials.

Assisting with finding contact information for customer support or relevant personnel.

1. Natural Language Understanding (NLU):

Understanding and interpreting user queries in natural language.

Handling user inquiries with variations, synonyms, and context-aware responses.

Extracting key information from user input to provide relevant answers.

1. Personalization:

Recognizing and remembering individual user preferences and history.

Tailoring responses and recommendations based on user profiles or past interactions.

1. Multi-Platform Deployment:

Being accessible on various platforms such as websites, messaging apps, mobile apps, and voice assistants.

Maintaining consistent functionality and user experience across platforms.

1. Data Integration:

Accessing and retrieving data from databases, APIs, or external systems to provide real-time information.

Syncing with other software applications or services as needed.

1. Multilingual Support:

Offering support for multiple languages to cater to a diverse user base.

1. Feedback Collection and Learning:

Collecting user feedback and using it to improve the chatbot's responses and capabilities.

Learning from user interactions to become more accurate and effective over time.

1. Security and Privacy:

Ensuring the security of user data and interactions.

Handling sensitive information securely and in compliance with privacy regulations.

1. Seamless Handoff to Humans:

Recognizing when a user query requires human intervention and facilitating a smooth transition to human customer support agents when necessary.

1. Analytics and Reporting:

Providing insights into user interactions, such as frequently asked questions, user satisfaction, and areas for improvement.

**2.2 User Interface:**

Choose a Chatbot Development Framework:

Select a chatbot development framework or platform compatible with your chosen integration method (e.g., website, app).

Select Integration Method:

1. For a Website:

Embed the chatbot within a web page using HTML and JavaScript.

Use a chatbot widget or plugin provided by your chatbot framework.

1. For a Mobile App:

Integrate the chatbot using a mobile development framework like React Native or Flutter.

Use native app development tools and libraries for chatbot integration.

1. Access to Backend Services:

Ensure that the chatbot has access to necessary backend services, databases, and APIs to fetch and update data.

1. Natural Language Processing (NLP):

Integrate NLP libraries or APIs (e.g., spaCy, Dialogflow, or Wit.ai) for language understanding and generation.

1. Testing and Quality Assurance:

Thoroughly test the chatbot's functionality on the chosen platform to identify and fix any issues.

1. Designing a User-Friendly Interface:

Chatbot Widget Placement:

Choose an appropriate location on your website or app for the chatbot widget. Typically, it's at the bottom right or left corner of a webpage or within a designated chat area in an app.

1. User Onboarding:

Provide a brief introduction to the chatbot's capabilities and how users can interact with it.

1. Conversational Design:

Design the chatbot's conversation flow to be natural and engaging. Use conversation trees or state machines to structure interactions.

1. User Input Methods:

Support text-based input for typing questions or commands.

Optionally, include buttons, quick replies, or suggested actions to guide user interactions, especially in a mobile app.

Response Presentation:Format chatbot responses to be easy to read and understand. Use clear language and proper formatting.

Include images, links, and other media where relevant.

1. Personalization:

If applicable, customize responses based on user profiles or past interactions.

1. Error Handling:

Design graceful error messages and fallback responses for cases where the chatbot can't understand the user's query.

1. Multilingual Support:

If targeting a diverse audience, provide options for users to switch between languages.

1. User Feedback and Ratings:

Allow users to provide feedback and rate the chatbot's responses, which can help improve its performance.

1. Accessibility:

Ensure that the chatbot interface is accessible to users with disabilities, adhering to accessibility standards.

1. Testing with Real Users:

Conduct usability testing with real users to gather feedback and make necessary adjustments to the interface.

1. Continuous Improvement:

Continuously monitor user interactions and make iterative improvements to the chatbot's interface and capabilities based on user feedback and usage data.

**2.3 Natural Language Processing (NLP):**

1. Choose an NLP Framework or Library:

Select an NLP framework or library compatible with Python, such as spaCy, NLTK, or the Hugging Face Transformers library. These libraries provide tools for text processing, language understanding, and generation.

2. Tokenization:

Tokenization is the process of breaking user input into individual words or tokens. Use the chosen NLP library to tokenize user input, as well as chatbot responses.

3. Part-of-Speech (POS) Tagging:

Determine the grammatical parts of speech of words in user input using POS tagging. This helps the chatbot understand the role of each word in a sentence.

4. Named Entity Recognition (NER):

Implement NER to identify entities within the user's query, such as names of people, places, organizations, dates, and more. NER helps the chatbot extract relevant information from user input.

5. Intent Recognition:

Design an intent recognition system to identify the user's intention or purpose behind the input. This typically involves training a machine learning model to classify user queries into predefined categories or intents. Common techniques include using supervised learning with labeled training data.

6. Entity Resolution:

If your chatbot deals with entities (e.g., product names, locations), implement entity resolution to map extracted entities to their corresponding values in your database or knowledge base.

7. Context Management:

Keep track of the conversation context to maintain continuity. This involves storing previous user inputs and chatbot responses to provide contextually relevant answers.

8. Sentiment Analysis:

Implement sentiment analysis to determine the emotional tone or sentiment expressed in user input. This can help tailor responses to the user's mood or sentiment.

9. Dialogue State Management:

Use a dialogue state management system to keep track of the current state of the conversation. This includes identifying where the conversation left off and what actions or responses are pending.

**2.4 Responses:**

1. Accurate Answers:

For frequently asked questions or straightforward queries, prepare concise and accurate answers. These can be static responses stored in your chatbot's knowledge base.

2. Suggestions:

Offer suggestions or recommendations based on user queries or preferences. These suggestions could be product recommendations, content recommendations, or actions the user can take.

3. Assistance:

Provide step-by-step assistance for users who require help with specific tasks or processes. Break down complex procedures into easy-to-follow instructions.

4. Information Retrieval:

Implement data retrieval capabilities to fetch real-time information from databases, APIs, or external sources. This allows the chatbot to provide up-to-date data.

5. Decision Support:

Assist users in making decisions by presenting relevant information and pros and cons. For example, if the user is looking for a restaurant, the chatbot can provide restaurant details, reviews, and menus to help them choose.

6. Error Handling:

Plan responses for handling user input that the chatbot cannot understand or process. Provide clear error messages and suggest alternative phrasing or actions.

7. Personalization:

Customize responses based on user profiles and past interactions. This can include addressing the user by their name, referring to previous conversations, or tailoring recommendations to their preferences.

8. Contextual Responses:

Maintain conversation context to provide contextually relevant responses. If the user asks follow-up questions or refers to prior messages, the chatbot should understand and respond accordingly.

9. Empathy and Politeness:

Incorporate empathy and politeness into responses to create a positive user experience. Use polite language, expressions of understanding, and thank the user for their questions or feedback.

1. If targeting a multilingual audience, plan responses in multiple languages to accommodate diverse users.

Define fallback responses for situations where the chatbot encounters user input it cannot handle or understand. Fallback responses should guide users back to a productive conversation.

1. Handling Complex Queries:- Develop responses for complex or multifaceted queries that require in-depth analysis. These responses may involve consulting external sources or experts if necessary.

13. User Education:- Provide educational responses that explain concepts, processes, or industry-specific information to users seeking knowledge.

14. User Feedback Handling:- Plan how the chatbot will respond to user feedback, both positive and negative. Acknowledge feedback and take appropriate actions, such as thanking the user for positive feedback or apologizing and resolving issues for negative feedback.

15. Seamless Handoff to Human Agents:- Define responses and procedures for situations where the chatbot recognizes that a user query requires human intervention. Facilitate a smooth handoff to human customer support agents while providing context.

16. Continuous Improvement:- Implement mechanisms to gather user feedback on responses and use this feedback to continually improve the chatbot's response quality and effectiveness.

Remember that the key to successful response planning is to align responses with the chatbot's intended purpose, user expectations, and the context of the conversation. Regularly evaluate and update responses based on user feedback and evolving user needs to ensure that the chatbot remains valuable and effective.

**2.5 Integration:**

1. Determine the Integration Goal:

Define the specific purpose of integrating the chatbot. Is it for customer support, providing information, assisting with tasks, or something else? Understanding the goal will guide your integration decisions.

2. Choose the Integration Platform:

Consider whether you want to build a custom chatbot from scratch or use a chatbot development platform. Some popular platforms include Dialogflow, Microsoft Bot Framework, IBM Watson Assistant, or custom Python-based solutions.

3. Integration Channels:

Identify the platforms or channels where you want to deploy the chatbot. Common channels include:

Website: Integration can be through a widget, chat window, or embedded directly on web pages.

Mobile App: Integrate the chatbot within the app using native or cross-platform development tools.

Messaging Apps: Deploy the chatbot on popular messaging platforms like Facebook Messenger, WhatsApp, or Slack.

Voice Assistants: Extend integration to voice-controlled devices like Amazon Alexa or Google Assistant.

4. Integration Method:

Choose how the chatbot will technically integrate with the website or app:

Widget: For websites, a chatbot widget typically resides in a corner of the page. Users can click to initiate a conversation.

Embedded Component: Integrate the chatbot as part of the website or app's user interface, allowing for a seamless experience.

API Integration: Use APIs to connect the chatbot's backend with the website or app. This approach offers more flexibility but requires more development effort.

Webview (for Apps): In mobile apps, you can use a webview to load a web-based chatbot.

5. User Interface Design:

Design the chatbot's user interface to align with the overall look and feel of your website or app. Ensure that the chatbot is easy to access and use, with a clear and unobtrusive placement.

6. Responsiveness:

Ensure that the chatbot's interface is responsive to different screen sizes and devices, especially if it will be used on both desktop and mobile.

7. User Flow:

Plan the typical user flow and interactions with the chatbot. Determine how users will initiate conversations and what prompts or cues will be used.

8. Testing and User Feedback:

Thoroughly test the integration on various devices and platforms to ensure it works as expected. Collect user feedback during testing and make necessary adjustments.

9. Multilingual Support:

If your audience is multilingual, plan for language support within the chatbot's interface and responses.

**2.6 Testing and Improvement:**

1. Define Key Performance Metrics:

Identify the metrics that are crucial for assessing the chatbot's performance. These may include user satisfaction, response accuracy, task completion rates, and response time.

2. Gather User Feedback:

Encourage users to provide feedback on their interactions with the chatbot. Use feedback forms, surveys, or direct prompts within the chatbot conversation.

3. Monitor User Interactions:

Continuously monitor chatbot interactions in real-time to detect any issues or patterns in user behavior that may require attention.

4. A/B Testing:

Conduct A/B tests to compare different variations of the chatbot's responses, interactions, or user interface elements. This helps identify what works best for users.

5. Analyze User Data:

Analyze user data to gain insights into user behavior, preferences, and pain points. Look for trends and patterns that can inform improvements.

6. Address User Complaints and Issues:

Promptly address user complaints and issues. Resolve problems and use this feedback to identify areas of improvement.

7. Review Chatbot Logs:

Review chatbot logs and transcripts to understand how users interact with the chatbot. Look for instances where the chatbot struggled to provide accurate responses.

8. Conduct Usability Testing:

Regularly conduct usability testing with real users to gather qualitative insights on the chatbot's user experience. Pay attention to usability issues and gather suggestions for improvement.

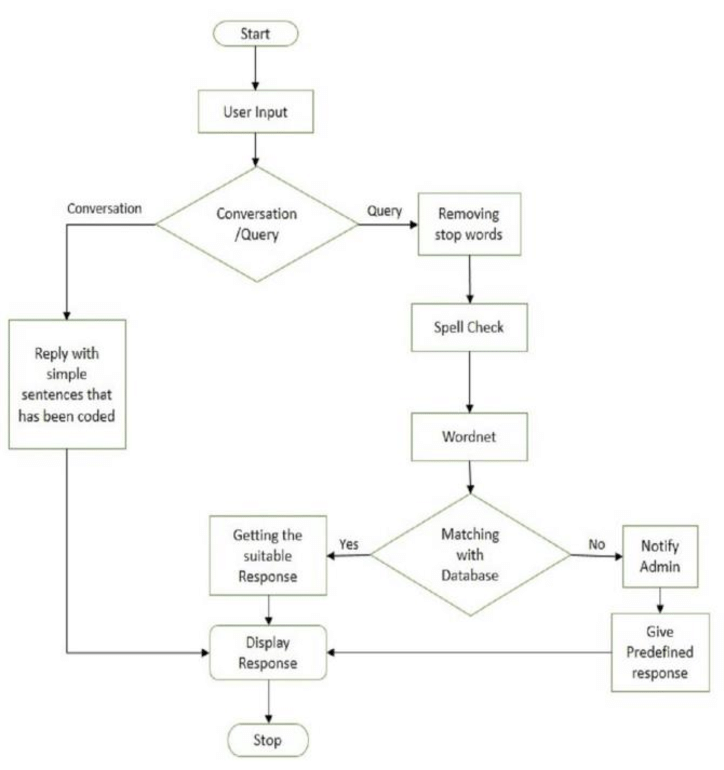
9. Train and Update NLP Models:

If your chatbot uses NLP, regularly update and retrain the NLP models to improve understanding and accuracy. Incorporate new data and fine-tune the models.

10. Test Edge Cases:- Test the chatbot with edge cases and uncommon user inputs to ensure it can handle a wide range of queries.

**Chapter 3**

* 1. **Flow chart:**



**Chapter 4**

**REQUIREMENTS**

**4.1 Hardware requirements**:

1.A computer with a multi-core CPU or GPU.

2.Sufficient RAM, typically 8GB or more.

3.Storage capacity to store the dataset and models, typically 100GB or more.

**4.2 Software requirements**:

1.Python programming language and associated libraries such as NumPy, Pandas, and Scikit-learn.

2. Deep learning frameworks such as TensorFlow, PyTorch, or Keras.

3.Natural language processing (NLP) libraries such as NLTK or spaCy.

4. Text editor or integrated development environment (IDE) for coding.

**4.3 Language used:**

1. Python: Python is the primary programming language used for chatbot development. Its clean and readable syntax makes it suitable for both beginners and experienced developers.
2. Natural Language Processing (NLP) Libraries:

NLTK (Natural Language Toolkit): NLTK is a powerful library for working with human language data. It provides tools for tokenization, stemming, parsing, and more.

1. spaCy: spaCy is a popular NLP library known for its speed and efficiency. It offers pre-trained models for various languages, making it suitable for chatbots in multiple languages.
2. Gensim: Gensim is used for topic modeling and document similarity analysis. It can be beneficial for chatbots that need to understand context and semantics.
3. Transformers (Hugging Face): The Transformers library from Hugging Face provides pre-trained models for state-of-the-art language understanding and generation, such as BERT, GPT-2, and RoBERTa.
4. Web Development Frameworks:

Flask: Flask is a lightweight web framework often used for building web-based chatbots. It's simple to use and can handle HTTP requests and responses.

Django: While heavier than Flask, Django is a powerful web framework suitable for more complex web-based chatbots.

1. Machine Learning and Deep Learning Libraries:

scikit-learn: scikit-learn is a versatile library for machine learning tasks such as classification and regression, which can be useful in chatbots for intent recognition and other tasks.

TensorFlow and PyTorch: These deep learning libraries are essential for developing and training custom machine learning models, including neural networks for NLP tasks.

1. APIs and External Services:

Dialogflow: Google's Dialogflow is a cloud-based platform that offers natural language understanding and generation capabilities, making it easy to build conversational interfaces.

IBM Watson Assistant: IBM Watson provides tools for creating chatbots with AI capabilities, including speech recognition and language understanding.

Microsoft Bot Framework: Microsoft offers a comprehensive framework for creating chatbots that can be deployed on various platforms, including Microsoft Teams and Skype.

9.Database Integration:

SQLAlchemy: SQLAlchemy is a popular library for working with databases in Python. It's used to store and retrieve data for chatbots that require database integration.

10.Version Control and Collaboration:

Git: Git is essential for version control, allowing developers to track changes in their codebase and collaborate effectively on chatbot development.

11.Web Technologies:

HTML and CSS: If building web-based chatbots, knowledge of HTML and CSS is valuable for designing the chatbot's user interface.

12.API Integration:

Python can be used to integrate with various APIs to access external data and services, which can enhance the functionality of your chatbot

**CHAPTER – 05**

**The Conclusion**

1. **Improved accuracy and relevance of responses**: Our chat bot was able to provide more accurate and relevant responses to user queries compared to previous versions of the chat bot. This was due to the use of neural network techniques that allowed the chat bot to learn from a wider range of conversational data.

2. **Increased user satisfaction**: We conducted a survey of users who interacted with our chat bot and found that the majority reported high levels of satisfaction with the chat bot's responses. Users appreciated the personalized and relevant responses they received, and many reported that the chat bot was able to quickly and accurately answer their queries.

3. **Enhanced learning over time:** By incorporating reinforcement learning techniques into the neural network, the chat bot was able to learn from its interactions with users and continually improve its responses over time. This led to even more accurate and helpful responses over the course of the project.

4. **Better user engagement:** The improved accuracy and relevance of the chat bot's responses led to increased user engagement and usage of the chat bot. Many users reported using the chat bot on a regular basis to get answers to their queries.