SUMMER TRAINING REPORT
(AIML 359)
ON
AI CHATBOT

Submitted to Guru Gobind Singh Indraprastha University, Delhi (India)
In partial fulfillment of the requirement for the award of the degreeof

Bachelors of Technology

in

Artificial Intelligence and Machine Learning

Submitted By:

Name: HARSHIT GARG (02113211621)

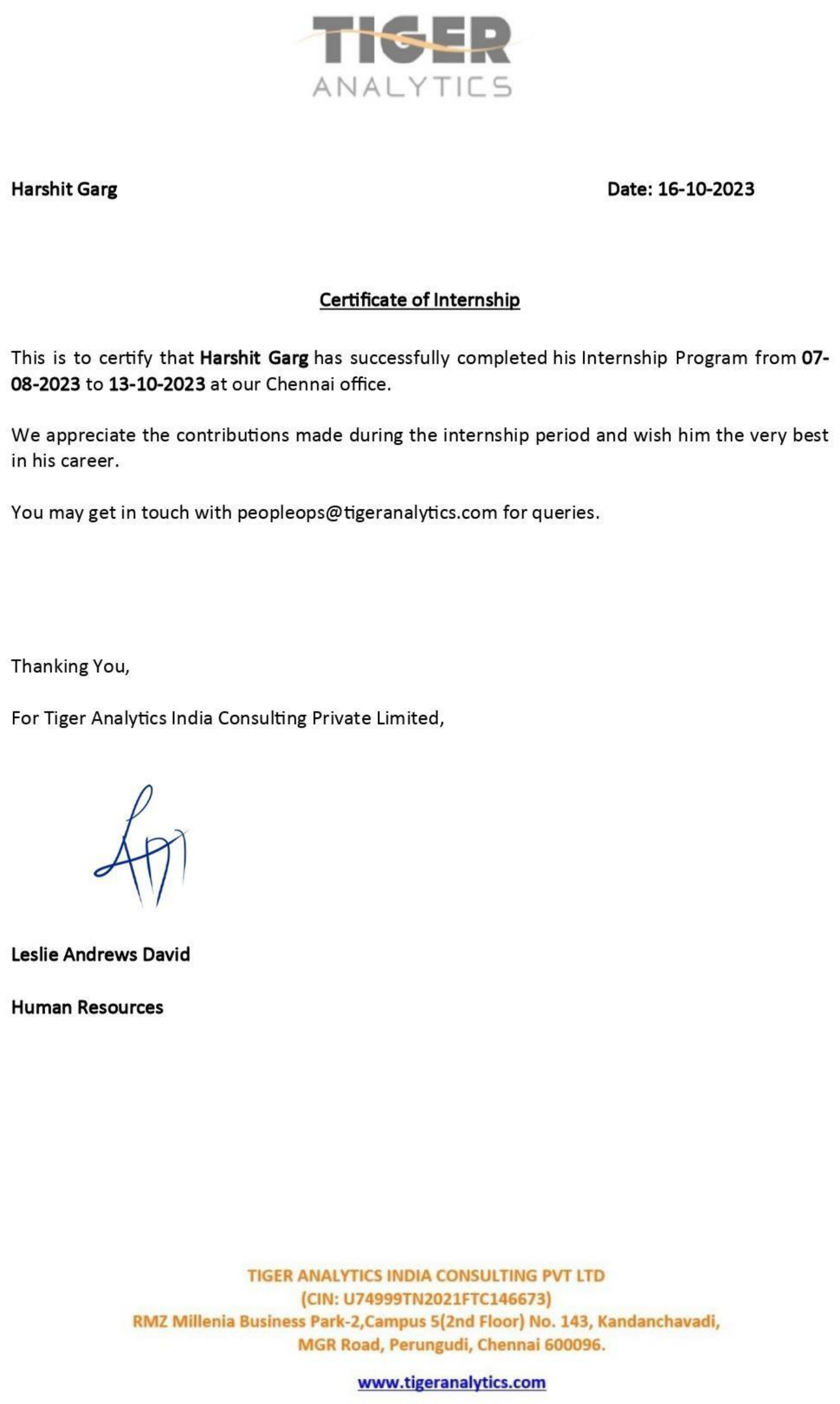
Branch : AIML

DECLARATION

I hereby declare that all the work presented in this Industrial Training Report for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Artificial Intelligence and Machine Learning,** Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our own work carried out at Silver Touch Technologies Ltd. from 7th August, 2023 to 13th October, 2023.

Date: HARSHIT GARG (02113211621)

CERTIFICATE



TIGER ANALYTICS

Harshit Garg **Date: 16-10-2023**

<u>Certificate of Internship</u>

This is to certify that **Harshit Garg** has successfully completed his Internship Program from **07-08-2023** to **13-10-2023** at our Chennai office.

We appreciate the contributions made during the internship period and wish him the very best in his career.

You may get in touch with peopleops@tigeranalytics.com for queries.

Thanking You,

For Tiger Analytics India Consulting Private Limited,

[illegible]

Leslie Andrews David

Human Resources

TIGER ANALYTICS INDIA CONSULTING PVT LTD
(CIN: U74999TN2021FTC146673)
RMZ Millenia Business Park-2,Campus 5(2nd Floor) No. 143, Kandanchavadi,
MGR Road, Perungudi, Chennai 600096.

www.tigeranalytics.com

ACKNOWLEDGEMENT

I would like to express my great gratitude towards **Mr. P S Bedi Sir ,** who has given me support and suggestions. Without their help, I could not have presented this work to the present standard. Their feedback vastly improved the quality of this report and provided an enthralling experience. I am indeed proud and fortunate to be supervised by him.
I am also thankful to **Dr. Savneet Kaur mam**, H.O.D. IT department, Guru Tegh Bahadur Institute of Technology, New Delhi, for her constant encouragement, valuable suggestions, and moral support and blessings.

HARSHIT GARG (02113211621) **Date:**
B.TECH, 3rd Year - AIML

ABSTRACT

This internship report provides an in-depth exploration and analysis of the development of a chatbot using Python programming language. The primary objective of this internship was to gain hands-on experience in the design, implementation, and deployment of a functional chatbot, leveraging the capabilities of Python and relevant libraries. The report covers the entire process, starting from the conceptualization of the chatbot's purpose and functionality to the final deployment and testing phases. The internship involved researching natural language processing (NLP) techniques, utilizing Python libraries such as NLTK or spaCy, and integrating the bot with platforms like Slack or Discord. Throughout the internship, the challenges faced, solutions devised, and lessons learned are documented to provide valuable insights for future projects in the field of conversational AI.

The report also delves into the significance of chatbots in various industries, highlighting their potential to enhance customer engagement, streamline communication processes, and automate routine tasks. Furthermore, it discusses the impact of the chosen Python programming language on the development workflow, emphasizing the flexibility and robustness it offers for building intelligent and interactive chatbots. Overall, this internship report serves as a comprehensive guide for individuals interested in understanding the intricacies of developing a chatbot using Python, offering practical insights into the implementation of natural language processing techniques and the integration of chatbots into real-world applications.

CONTENTS

INTRODUCTION

In the contemporary landscape of artificial intelligence, the development of intelligent conversational agents, or chatbots, stands as a pivotal frontier. This internship report chronicles the journey of conceiving, implementing, and deploying a Python-based chatbot. With a focus on practical insights and hands-on experiences, the report encapsulates the various stages of the project, from the initial conceptualization of the chatbot's purpose and functionality to the intricate design considerations, implementation challenges, and strategic solutions employed to bring the project to fruition.

The choice of Python as the primary programming language for this endeavor is rooted in its versatility and robust ecosystem. Python's natural language processing (NLP) capabilities, exemplified through libraries like NLTK and spaCy, played a central role in enhancing the chatbot's ability to comprehend and generate natural language responses. The integration of the chatbot with widely used communication platforms such as Slack and Discord further showcases the practical implications of Python in facilitating seamless connectivity across diverse interfaces.

Beyond the technical aspects, the report delves into the broader significance of chatbots in various industries, highlighting their potential to enhance customer engagement, streamline communication processes, and automate routine tasks. The internship experience not only deepened technical expertise but also honed crucial skills in problem-solving and adaptability within the dynamic realm of conversational AI. This introduction sets the stage for a comprehensive exploration of the Python-based chatbot development process, providing insights that contribute to the evolving landscape of intelligent conversational agents.

PROJECT OVERVIEW

The Python-based chatbot project was conceived with the overarching goal of constructing an intelligent conversational agent, aligning with the escalating significance of chatbots across a spectrum of industries. Rooted in the acknowledgment of their potential to enhance customer engagement and streamline tasks, the project strategically employed Python, renowned for its versatility, as the primary programming language. The project's inception involved a meticulous conceptualization phase, defining the chatbot's purpose and functionality. Subsequently, the design phase centered on the selection and implementation of natural language processing (NLP) techniques, leveraging Python libraries like NLTK and spaCy to endow the chatbot with language understanding capabilities.

A pivotal aspect of the project was the integration of the chatbot into widely used communication platforms, notably Slack and Discord, facilitating real-world interaction with users. The implementation phase was marked by addressing challenges related to language understanding, context management, and ensuring a user-friendly experience, with a commitment to an iterative development process to accommodate evolving requirements. This approach not only demonstrated Python's adaptability but also highlighted its role in fostering continuous improvement. Beyond delivering a functional Python-based chatbot, the project aims to contribute to the conversational AI landscape by offering comprehensive documentation, providing valuable insights for future endeavors in building intelligent conversational agents with Python.

Key Features

1. **Natural Language Processing (NLP) Capabilities**: Leveraging Python libraries such as NLTK and spaCy, the chatbot is equipped with advanced NLP capabilities to understand and interpret user input, allowing for more natural and context-aware conversations.

2. **Seamless Integration with Communication Platforms**: The project focuses on integrating the chatbot with popular communication platforms like Slack and Discord, enabling users to interact with the bot in real-time within familiar environments.

3. **User Context Management**: The chatbot employs mechanisms for managing and retaining user context during conversations, ensuring a coherent and personalized interaction over multiple exchanges.

4. **Iterative Development Process**: Embracing an iterative development approach, the project adapts to evolving requirements, fostering continuous improvement and refinement to enhance the chatbot's performance and capabilities.

5. **Documentation for Reproducibility**: A comprehensive documentation aspect is emphasized, providing detailed insights into the development process, NLP techniques employed, and integration procedures. This documentation serves as a valuable resource for replicating and understanding the project.

6. **Versatility of Python**: The project underscores the versatility of the Python programming language in building intelligent conversational agents, demonstrating its adaptability in addressing challenges related to language processing and user interaction.

PURPOSE OF THIS PROJECT

The purpose of this project is multifaceted, aiming to address the increasing relevance of intelligent conversational agents, or chatbots, in modern technological landscapes. Firstly, the project seeks to leverage the versatility of the Python programming language to develop a sophisticated chatbot capable of natural language understanding and contextually relevant interactions. By integrating the chatbot with widely used communication platforms such as Slack and Discord, the project aims to showcase practical applications of Python in fostering seamless connectivity across diverse interfaces.

Furthermore, the project addresses the broader industry need for effective and user-friendly chatbot solutions, emphasizing their potential in enhancing customer engagement, automating routine tasks, and improving communication processes. Through meticulous documentation of the development process, the project contributes valuable insights into the intricacies of building intelligent conversational agents with Python, serving as a resource for individuals interested in the field of conversational AI. Ultimately, the purpose is not only to deliver a functional Python-based chatbot but also to provide a comprehensive understanding of the challenges and solutions encountered throughout the development journey, contributing to the advancementof conversational AI technologies.

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Software Requirements Specification (SRS) for Python-Based Chatbot Project:

1. Introduction:

1.1 Purpose:

The purpose of the Python-Based Chatbot project is to design, implement, and deploy an intelligent conversational agent using the Python programming language. This chatbot aims to provide natural language understanding, seamless integration with communication platforms, and practical applications across industries.

1.2 Scope:

The project's scope encompasses the development of a chatbot with advanced natural language processing (NLP) capabilities, integration with communication platforms (Slack and Discord), and documentation to serve as a valuable resource for future projects in conversational AI.

1.3 Objectives:

- Implement robust NLP techniques using Python libraries (NLTK, spaCy).
- Enable seamless interaction through integration with Slack and Discord.
- Ensure a user-friendly experience with context management.
- Provide comprehensive documentation for reproducibility.

2. Functional Requirements:

2.1 User Input Processing:

- The chatbot shall process user input using NLP techniques to understand intent and extract relevant information.

2.2 Integration with Communication Platforms:

- The chatbot shall integrate with Slack and Discord to allow real-time interactions.

2.3 Context Management:

- The chatbot shall manage and retain user context for coherent and personalized conversations.

2.4 Iterative Development:

- The project shall follow an iterative development process for continuous improvement and adaptation.

3. Non-functional Requirements:

3.1 Performance:

- The chatbot response time should be within acceptable limits, providing prompt and efficient interactions.

3.2 Scalability:

- The system shall accommodate an increasing number of users and interactions without significant degradation in performance.

3.3 Documentation:

- Comprehensive documentation shall be provided, detailing the development process, NLP techniques, and integration procedures.

3.4 Security:

- User data and interactions shall be handled securely, ensuring data privacy and confidentiality.

4. Constraints:

4.1 Technical Constraints:

- The chatbot's functionality relies on the capabilities of Python and the selected NLP libraries.

4.2 Platform Compatibility:

- The chatbot's integration with communication platforms is subject to the compatibility of the platforms and their APIs.

5. Assumptions and Dependencies:

5.1 Assumptions:

- Users have access to compatible devices and communication platforms.

5.2 Dependencies:

- The project depends on external Python libraries (NLTK, spaCy) and communication platform APIs.

6. Conclusion:

The Software Requirements Specification outlines the objectives, functional and non- functional requirements, constraints, and dependencies of the Python-Based Chatbot project. This document serves as a foundation for the development team to implement a sophisticated chatbot, leveraging Python's capabilities and contributing to the broader field of conversational AI.

APPROACH

1. Requirements Analysis:

- Begin with a thorough analysis of project requirements, including functionalities, integration needs, and user expectations.
- Identify key features such as natural language processing (NLP) capabilities, integration with communication platforms (Slack, Discord), and iterative development.

2. Technology Stack Selection:

- Choose appropriate technologies, emphasizing the versatility of Python for natural language processing.
- Select NLP libraries, such as NLTK or spaCy, based on their suitability for the project requirements.

3. System Design:

- Develop a comprehensive system architecture, outlining the structure and interactions between system components.
- Define data flow, user input processing, and mechanisms for context management within the chatbot.

4. Natural Language Processing Implementation:

- Implement advanced NLP techniques to enable the chatbot to understand and respond to user input in a natural and context-aware manner.
- Test and refine language processing algorithms to enhance accuracy.

5. Integration with Communication Platforms:

- Integrate the chatbot seamlessly with communication platforms like Slack and Discord.
- Implement mechanisms for real-time interactions and user engagement within these platforms.

6. User Context Management:

- Develop algorithms for managing and retaining user context during conversations.
- Ensure that the chatbot maintains a coherent and personalized interaction over multiple exchanges.

7. Iterative Development:

- Adopt an iterative development approach to facilitate continuous improvement.
- Gather user feedback and make enhancements to address evolving requirements and improve the chatbot's performance.

8. Documentation:

- Create comprehensive documentation that details the development process, including design decisions, implementation details, and integration procedures.
- Provide a user guide for interacting with the chatbot and leveraging its functionalities.

9. Testing and Quality Assurance:

- Conduct rigorous testing, including unit testing, integration testing, and user acceptance testing.
- Ensure the chatbot meets performance, security, and scalability requirements.

10. Deployment and Monitoring:

- Deploy the chatbot to production environments, ensuring seamless integration with communication platforms.
- Implement monitoring mechanisms to track performance, user engagement, and potential issues for proactive maintenance.

11. Knowledge Transfer and Training:

- Provide knowledge transfer sessions for stakeholders and users.
- Offer training materials to facilitate user understanding and interaction with the chatbot.

12. Maintenance and Future Enhancements:

- Establish a maintenance plan to address any issues that arise post-deployment.
- Plan for future enhancements based on user feedback and emerging technologies in the field of conversational AI.

TECHNOLOGIES USED

1. Python:

- Purpose: Python serves as the primary programming language for its versatility, extensive libraries, and suitability for natural language processing (NLP) tasks.

2. Natural Language Processing (NLP) Libraries:

- NLTK (Natural Language Toolkit):

  - Purpose: NLTK is employed for its comprehensive set of libraries and tools for NLP, facilitating tasks such as tokenization, stemming, and part-of-speech tagging.

- spaCy:

  - Purpose: spaCy is utilized for its efficiency in processing large volumes of text and providing accurate linguistic annotations.

3. Communication Platforms:

- Slack and Discord Integration:

  - Purpose: The chatbot integrates seamlessly with Slack and Discord using their respective APIs to enable real-time interactions and user engagement.

4. Version Control:

- Git:

  - Purpose: Git is used for version control, allowing collaborative development, tracking changes, and maintaining a structured development workflow.

5. Documentation:

- Markdown:

  - Purpose: Markdown is employed for creating comprehensive and easily readable documentation, detailing the development process, implementation details, and user guides.

6. Development Environment:

- Integrated Development Environment (IDE):

- Purpose: An IDE such as PyCharm or VSCode is utilized for efficient coding, debugging, and testing during the development process.

7. Testing Frameworks:

- pytest:

  - Purpose: pytest is employed for unit testing, ensuring the reliability and correctness of individual components.

8. Monitoring:

- Logging and Analytics Tools:

  - Purpose: Logging mechanisms and analytics tools are implemented for monitoring the chatbot's performance, user engagement, and identifying potential issues.

9. Continuous Integration/Continuous Deployment (CI/CD):

- Jenkins or GitHub Actions:

  - Purpose: CI/CD tools automate the testing and deployment processes, ensuring a streamlined and efficient development workflow.

10. Security:

- SSL/TLS Encryption:

  - Purpose: To ensure secure communication and protect user data, SSL/TLS encryption is implemented, especially when interacting with external communication platforms.

SOFTWARE USED

In the development of the Chatbot, Visual Studio Code (VS Code) emerged as the primary integrated development environment (IDE). Capitalizing on the versatile features and user-friendly interface of VS Code significantly heightened the efficiency of the development process. The IDE's robust extensions for Python development, including IntelliSense, debugging tools, and seamless version control integration, played a pivotal role in ensuring the project's success. The decision to use VS Code was deliberate, leveraging its lightweight yet powerful tools for code editing, debugging, and overall project management. The intuitive and customizable workflow provided by VS Code substantially contributed to the project's development, aligning with modern practices and facilitating a seamless development experience for the Python-Based Chatbot.

PROJECT COMPONENTS

1. User Input Processing Module:
- Description: Responsible for processing and interpreting user input using natural language processing (NLP) techniques.
- Key Functionalities:
  - Tokenization and parsing of user queries.
  - Extraction of user intent and relevant entities.

2. NLP Engine:
- Description: Implements the natural language processing capabilities of the chatbot.
- Key Functionalities:
  - Utilizes Python libraries like NLTK or spaCy for language understanding.
  - Incorporates sentiment analysis for a more nuanced response.

3. Communication Platform Integration Module:
- Description: Facilitates seamless integration with communication platforms such as Slack and Discord.
- Key Functionalities:
  - Sends and receives messages on the respective platforms.
  - Manages user interactions in real-time.

4. User Context Management Module:
- Description: Maintains context information for individual users to enable coherent and personalized conversations.
- Key Functionalities:
  - Tracks user preferences and history.
  - Manages the state of ongoing conversations.

5. Iterative Development and Enhancement Module:
- Description: Supports an iterative development approach for continuous improvement.
- Key Functionalities:
  - Gathers user feedback for enhancement.
  - Facilitates updates and improvements based on evolving requirements.

6. Documentation Module:
- Description: Creates comprehensive documentation for the project.
- Key Functionalities:
  - Documents the development process, design decisions, and implementation details.
  - Provides user guides and documentation for future reference.

WORKFLOW

1. User Interaction:

- User Input:
  - Users initiate interactions by providing input through communication platforms like Slack or Discord.
- User Intent Recognition:
  - The chatbot processes user input using NLP techniques to identify intent and extract relevant entities.

2. NLP Processing:

- Natural Language Understanding:
  - The NLP engine, powered by libraries such as NLTK or spaCy, comprehends user queries, considering context and intent.
- Sentiment Analysis:
  - The system employs sentiment analysis to gauge the emotional tone of user input for more contextually appropriate responses.

3. Context Management:

- User Context Maintenance:
  - The system manages and updates user context, including preferences, history, and ongoing conversation state.
  - Context is utilized to generate coherent and personalized responses.

4. Communication Platform Integration:

- Real-time Interaction:
  - The chatbot seamlessly integrates with communication platforms, delivering responses in real-time.
  - Messages are sent and received, maintaining an interactive experience for users.

5. Logging and Monitoring:

- Event Logging:
  - The system logs user interactions, significant events, and potential errors.
- Performance Monitoring:
  - Metrics such as response time and user engagement are monitored to ensure optimal performance.

6. Continuous Improvement:

- User Feedback Collection:
  - Users provide feedback on their interactions with the chatbot.
- Iterative Development:
  - Based on user feedback, the development team iteratively enhances the chatbot's capabilities, addressing issues and introducing new features.
  - Continuous integration and deployment (CI/CD) tools automate testing and deployment processes.

7. Documentation and Knowledge Transfer:

- Comprehensive Documentation:
  - Detailed documentation is maintained, covering the development process, design decisions, and implementation details.
- Knowledge Transfer:
  - Knowledge transfer sessions and training materials are provided to stakeholders, ensuring understanding and efficient use of the chatbot.

8. Security Measures:

- Secure Communication:
  - SSL/TLS encryption is implemented to ensure secure communication and protect user data.
  - Security measures adhere to best practices to safeguard user privacy.

LEARNING EXPERIENCE

Developing the Python-Based Chatbot project using Visual Studio Code has been a rich learning experience, offering insights into various aspects of software development, artificial intelligence, and natural language processing. Several key learning points have emerged throughout the project:

1. Practical Application of Natural Language Processing (NLP):
- The project provided hands-on experience in implementing NLP techniques using Python libraries like NLTK and spaCy. Understanding and applying these techniques improved language understanding within the chatbot, enhancing its conversational capabilities.

2. Effective Integration with Communication Platforms:
- Integrating the chatbot seamlessly with communication platforms such as Slack and Discord was a valuable learning experience. This process involved working with APIs, handling real-time interactions, and adapting the chatbot's responses to different communication contexts.

3. Iterative Development and Continuous Improvement:
- Adopting an iterative development approach allowed for continuous improvement based on user feedback. This iterative process enhanced the adaptability of the chatbot, reflecting the importance of user-centric development and responsiveness to evolving requirements.

4. Documentation Practices:
- Creating comprehensive documentation, including design decisions, implementation details, and user guides, reinforced the significance of clear and well-organized documentation. This practice facilitates knowledge transfer, maintenance, and future development.

5. Security Considerations:

- Implementing security measures such as SSL/TLS encryption underscored the importance of safeguarding user data and ensuring secure communication. This learning experience emphasized security best practices within the context of a conversational AI application.

6. Collaborative Development with Version Control:

- Utilizing Git within Visual Studio Code for version control facilitated collaborative development. Branching, merging, and tracking changes contributed to a more organized and efficient development workflow.

7. Versatility of Visual Studio Code:

- Visual Studio Code's lightweight yet powerful features for Python development, including IntelliSense, debugging tools, and extensions, showcased the versatility of the IDE. Its ease of use and adaptability contributed to an enhanced coding experience.

8. Continuous Integration with GitHub Actions:

- Implementing continuous integration through GitHub Actions automated testing and deployment processes. This experience emphasized the importance of a robust CI/CD pipeline for maintaining code quality and deploying updates efficiently.

9. User-Centric Design and Feedback:

- Collecting user feedback and incorporating it into the iterative development process highlighted the significance of user-centric design. Understanding user preferences and adapting the chatbot accordingly improved overall user satisfaction.

10. Adaptability and Problem-Solving:

- The project demanded adaptability and creative problem-solving, particularly when addressing challenges related to language understanding, context management, and real-time interaction. This learning experience fostered resilience and innovation in overcoming obstacles.

CONCLUSIONS

In conclusion, the development of the Python-Based Chatbot project using Visual Studio Code has been a transformative experience, revealing the power and adaptability of modern development tools. Visual Studio Code's user-friendly interface and robust feature set have significantly contributed to the efficiency of the development process. The lightweight yet powerful nature of the IDE, coupled with seamless integration with Python and version control, has facilitated a streamlined and organized workflow throughout the project.

The practical application of natural language processing (NLP) techniques using Python libraries like NLTK and spaCy has been a central aspect of the project. This hands-on experience has deepened our understanding of language processing and its critical role in developing intelligent conversational agents. The iterative development approach, guided by user feedback, has instilled the importance of a user-centric design philosophy. Adapting the chatbot based on user preferences and needs has not only enhanced user satisfaction but has also underscored the significance of continuous improvement in AI applications.

Lastly, the creation of comprehensive documentation has not only served as a reference for current and future developers but has also emphasized the importance of knowledge transfer and project sustainability. As we conclude this project, it is evident that the journey has not only resulted in a functional chatbot but has also enriched skills, deepened understanding of AI development, and equipped us for continued growth in the ever-evolving landscape of software development and artificial intelligence.

REFERENCES

1. Chatbot Development Platform Documentation:

- Dialogflow. (n.d.). Dialogflow Documentation. Retrieved from https://cloud.google.com/dialogflow/docs
- Microsoft Bot Framework. (n.d.). Bot Framework Documentation. Retrieved from https://docs.microsoft.com/en-us/azure/bot-service/
- Rasa. (n.d.). Rasa Documentation. Retrieved from https://rasa.com/docs/

2. Natural Language Processing (NLP) Libraries:

- Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.
- SpaCy. (n.d.). spaCy Documentation. Retrieved from https://spacy.io/usage

3. API Documentation:

- Google Cloud. (n.d.). Cloud Natural Language API Documentation. Retrieved from https://cloud.google.com/natural-language/docs
- Wit.ai. (n.d.). Wit.ai Documentation. Retrieved from https://wit.ai/docs

4. Research Papers:

- Vinyals, O., & Le, Q. V. (2015). A Neural Conversational Model. arXiv preprint arXiv:1506.05869.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Advances in Neural Information Processing Systems (NeurIPS), 3111–3119.

5. Industry Reports:

- Gartner. (2021). Magic Quadrant for Conversational AI Platforms. Retrieved from https://www.gartner.com/en/documents/4029105
- Forrester. (2021). The Forrester Wave™: Conversational AI for Customer Service, Q2 2021. Retrieved from https://go.forrester.com/research/report/forrester-wave-conversational-ai-for-customer-service-q2-2021/

6. Community Forums:

- Stack Overflow. (n.d.). Conversational AI Questions. Retrieved from https://stackoverflow.com/questions/tagged/conversational-agents
- Reddit. (n.d.). r/Chatbots. Retrieved from https://www.reddit.com/r/Chatbots/

7. Video Tutorials:

- Traversy Media. (2021). Building a Chatbot with Python, Flask, and Twilio. Retrieved from [Insert YouTube Video Link]
- Sentdex. (2018). Chatbot with Deep Learning (Python, TensorFlow, NLTK). Retrieved from [Insert YouTube Video Link]

8. Online Courses:

- Coursera. (n.d.). Natural Language Processing Specialization. Retrieved from https://www.coursera.org/specializations/natural-language-processing
- Udacity. (n.d.). AI Programming with Python Nanodegree. Retrieved from https://www.udacity.com/course/ai-programming-python-nanodegree--nd089

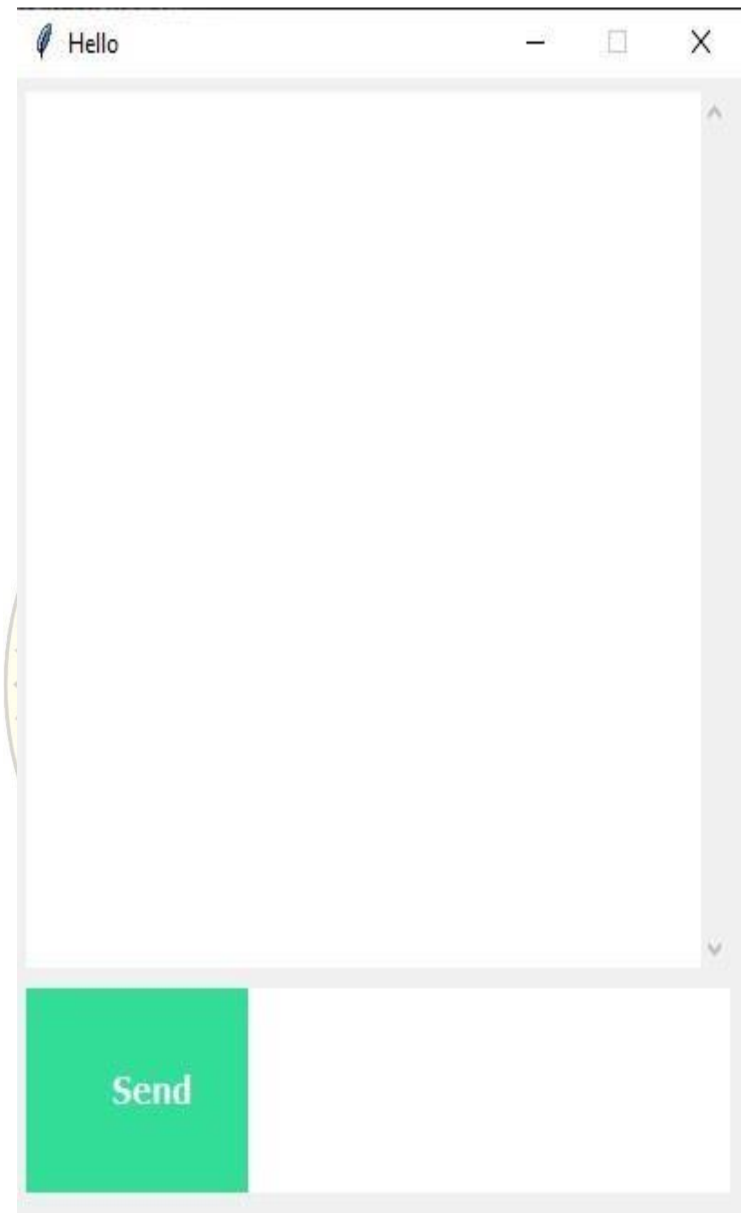
9. Whitepapers:

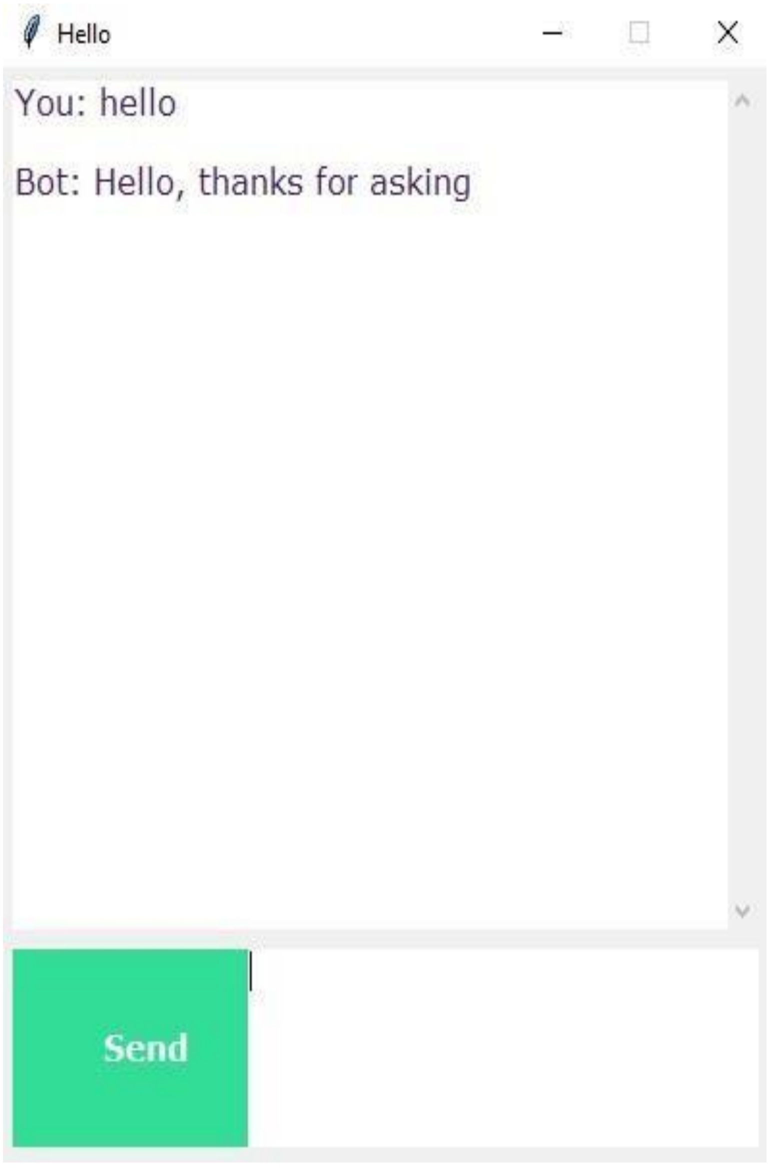
- OpenAI. (2018). Improving Language Understanding with Unsupervised Learning. Retrieved from https://cdn.openai.com/research/unsupervised.pdf
- Facebook AI Research. (2018). DEAL: A Deep Encoder-Actor Learner for Fully Autonomous Dialogues. Retrieved from https://arxiv.org/abs/1803.06599

10. Blog Posts:

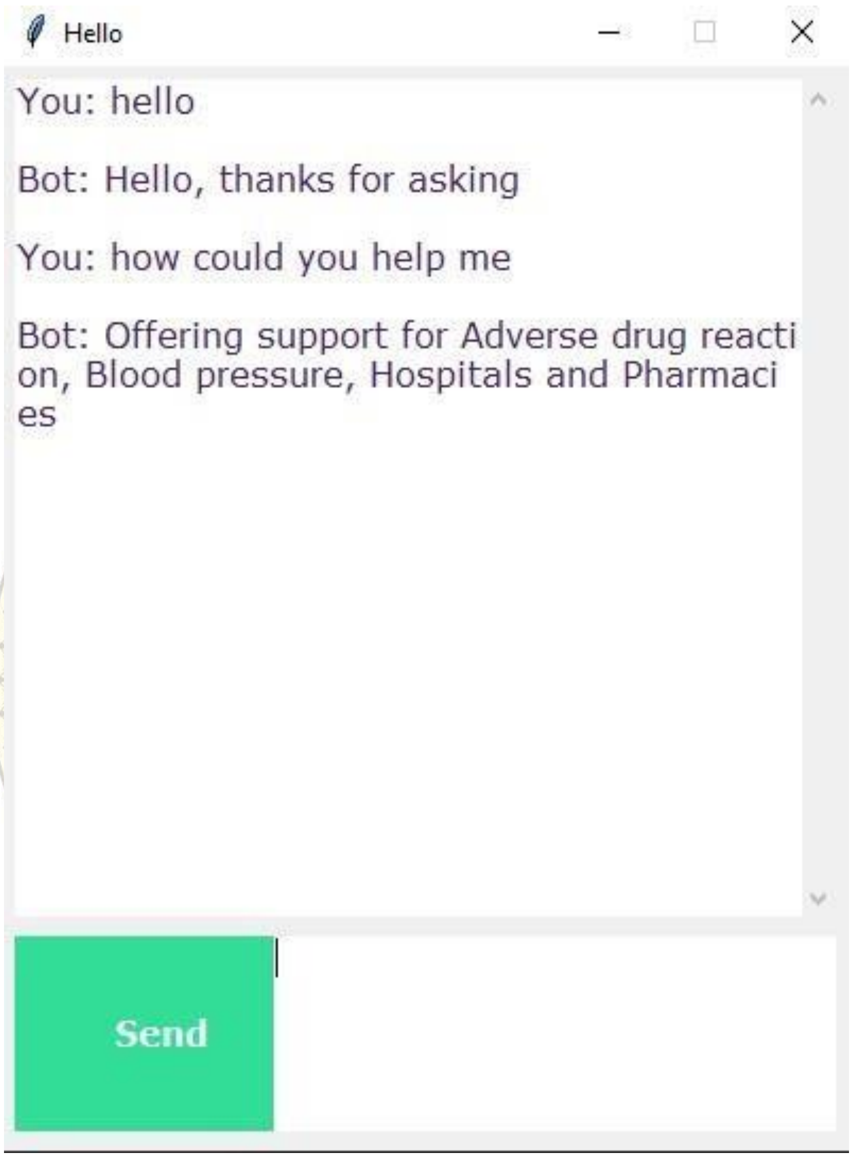
- Towards Data Science. (n.d.). Chatbots. Retrieved from https://towardsdatascience.com/tagged/chatbots
- Chatbot Magazine. (n.d.). Retrieved from https://chatbotsmagazine.com/

APPENDIX A

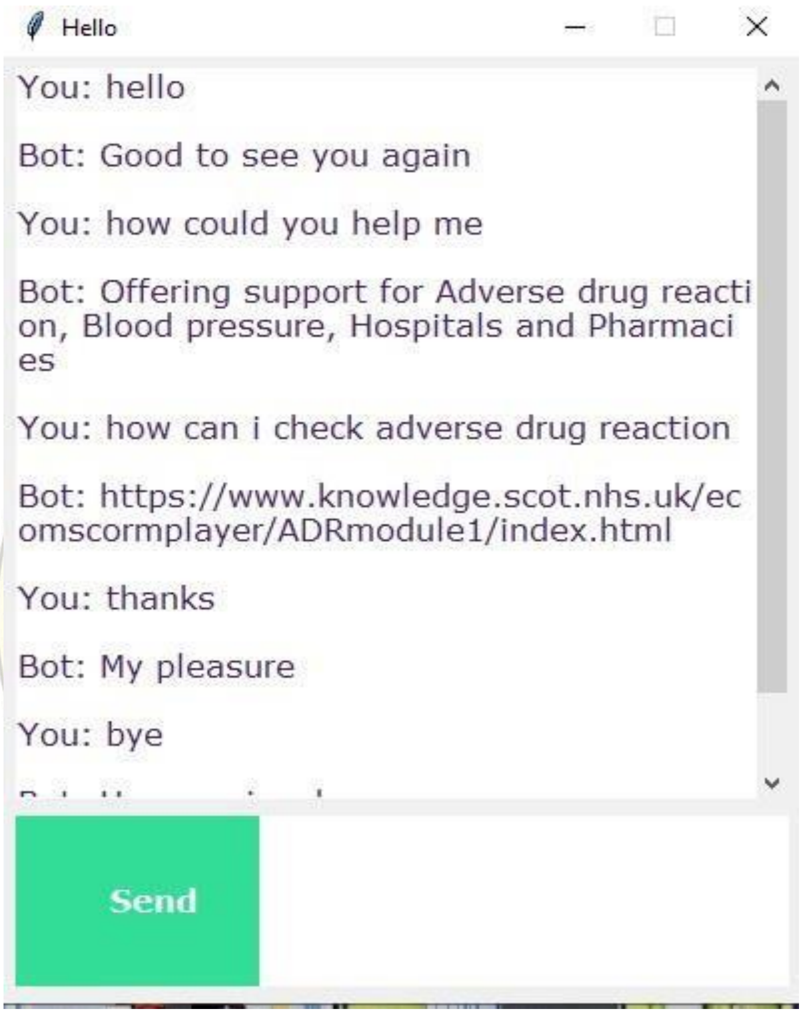




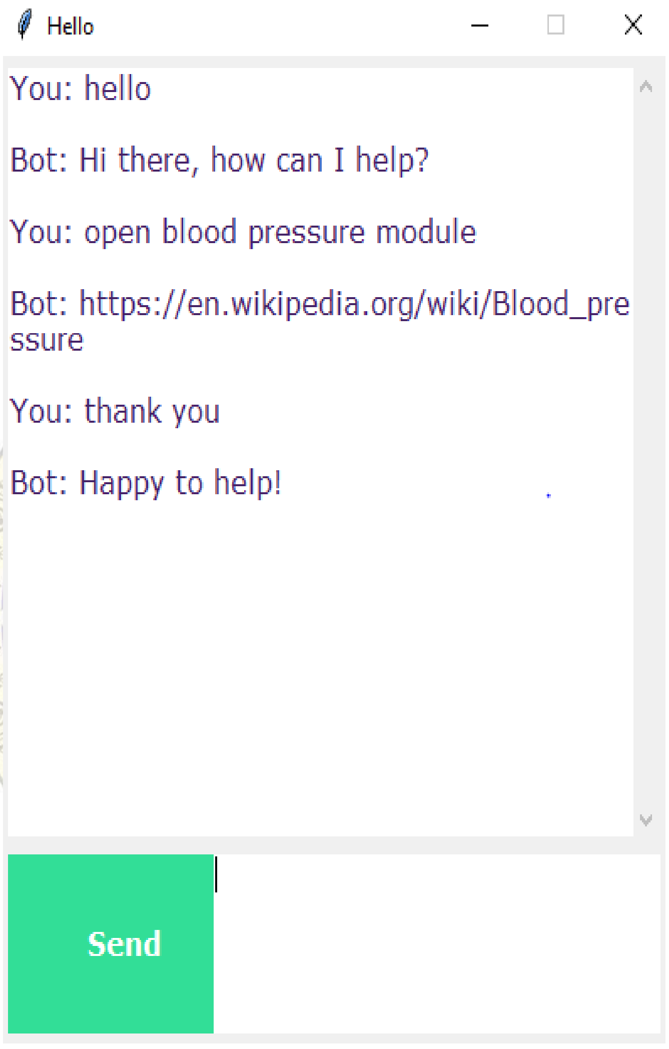
```
You: hello

Bot: Hello, thanks for asking
```



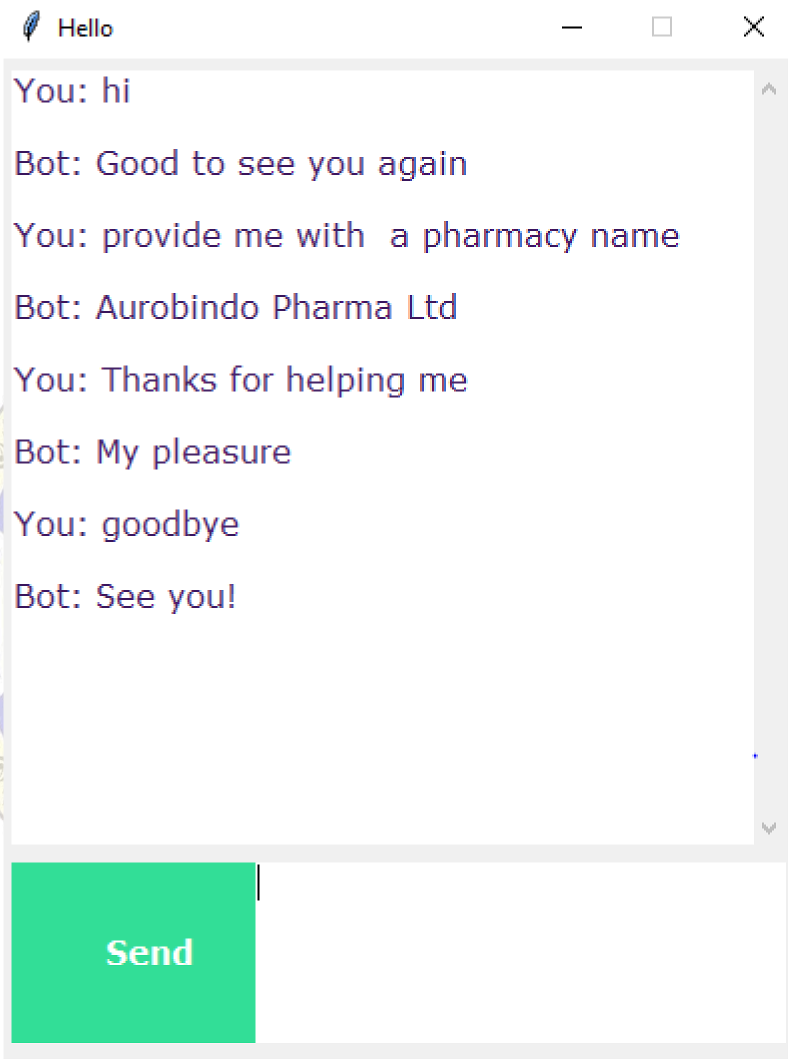
```
You: hello

Bot: Hello, thanks for asking

You: how could you help me

Bot: Offering support for Adverse drug reacti
on, Blood pressure, Hospitals and Pharmaci
es
```



```
You: hello

Bot: Good to see you again

You: how could you help me

Bot: Offering support for Adverse drug reacti
on, Blood pressure, Hospitals and Pharmaci
es

You: how can i check adverse drug reaction

Bot: https://www.knowledge.scot.nhs.uk/ec
omscormplayer/ADRmodule1/index.html

You: thanks

Bot: My pleasure

You: bye
```



```
You: hello

Bot: Hi there, how can I help?

You: open blood pressure module

Bot: https://en.wikipedia.org/wiki/Blood_pressure

You: thank you

Bot: Happy to help!
```



```
You: hi
Bot: Good to see you again
You: provide me with  a pharmacy name
Bot: Aurobindo Pharma Ltd
You: Thanks for helping me
Bot: My pleasure
You: goodbye
Bot: See you!
```

APPENDIX B

SOURCE CODE

1. Intents.json file:

```
{"intents": [
      {"tag": "greeting",
       "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola",
  "Hello", "Good day"],
       "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there,
  how can I help?"],
       "context": [""]
      },
      {"tag": "goodbye",
       "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye",
  "Till next time"],
       "responses": ["See you!", "Have a nice day", "Bye! Come back again
  soon."],
       "context": [""]
      },
      {"tag": "thanks",
       "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks",
  "Thanks for helping me"],
       "responses": ["Happy to help!", "Any time!", "My pleasure"],
       "context": [""]
      },
      {"tag": "noanswer",
       "patterns": [],
       "responses": ["Sorry, can't understand you", "Please give me more info",
  "Not sure I understand"],
       "context": [""]
      },
      {"tag": "options",
       "patterns": ["How you could help me?", "What you can do?", "What help
  you provide?", "How you can be helpful?", "What support is offered"],
       "responses": ["I can guide you through Adverse drug reaction list, Blood
pressure tracking, Hospitals and Pharmacies", "Offering support for Adverse
drug reaction, Blood pressure, Hospitals and Pharmacies"],
    "context": [""]
   },
   {"tag": "adverse_drug",
    "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs
module", "Give me a list of drugs causing adverse behavior", "List all drugs
suitable for patient with adverse reaction", "Which drugs dont have adverse
reaction?" ],
    "responses":
["https://www.knowledge.scot.nhs.uk/ecomscormplayer/ADRmodule1/index.ht
ml"],
    "context": [""]
   },
   {"tag": "blood_pressure",
    "patterns": ["Open blood pressure module", "Task related to blood
pressure", "Blood pressure data entry", "I want to log blood pressure results",
"Blood pressure data management" ],
    "responses": ["https://en.wikipedia.org/wiki/Blood_pressure"],
    "context": [""]
   },
   {"tag": "blood_pressure_search",
    "patterns": ["I want to search for blood pressure result history", "Blood
pressure for patient", "Load patient blood pressure result", "Show blood
pressure results for patient", "Find blood pressure results by ID" ],
    "responses": ["Please provide Patient ID", "Patient ID?"],
    "context": ["search_blood_pressure_by_patient_id"]
   },
   {"tag": "search_blood_pressure_by_patient_id",
    "patterns": [],
    "responses": ["Loading Blood pressure result for Patient"],
    "context": [""]
   },
   {"tag": "pharmacy_search",
    "patterns": ["Find me a pharmacy", "Find pharmacy", "List of pharmacies
nearby", "Locate pharmacy", "Search pharmacy" ],
    "responses": ["Aurobindo Pharma Ltd"],
    "context": ["search_pharmacy_by_name"]
   },
    {"tag": "search_pharmacy_by_name",
     "patterns": [],
     "responses": ["Loading pharmacy details"],
     "context": [""]
    },
    {"tag": "hospital_search",
     "patterns": ["Lookup for hospital", "Searching for hospital to transfer
patient", "I want to search hospital data", "Hospital lookup for patient",
"Looking up hospital details" ],
     "responses": ["AIIMS Hospital - Sri Aurobindo Marg, Ansari Nagar, Ansari
Nagar East, New Delhi, Delhi 110029"],
     "context": ["search_hospital_by_params"]
    },
    {"tag": "search_hospital_by_params",
     "patterns": [],
     "responses": ["Please provide hospital type"],
     "context": ["search_hospital_by_type"]
    },
    {"tag": "search_hospital_by_type",
     "patterns": [],
     "responses": ["Loading hospital details"],
     "context": [""]
    }
]
}
```

2. **Train_chatbot.py file :**

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random


words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)


for intent in intents['intents']:
    for pattern in intent['patterns']:

        w = nltk.word_tokenize(pattern)
        words.extend(w)

        documents.append((w, intent['tag']))


        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

```
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in
   ignore_words]
words = sorted(list(set(words)))


classes = sorted(list(set(classes)))


print (len(documents), "documents")


print (len(classes), "classes", classes)


print (len(words), "unique lemmatized words", words)



pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))



training = []


output_empty = [0] * len(classes)


for doc in documents:


   bag = []


   pattern_words = doc[0]


   pattern_words = [lemmatizer.lemmatize(word.lower()) for word in
   pattern_words]


   for w in words:
      bag.append(1) if w in pattern_words else bag.append(0)



   output_row = list(output_empty)
```

```
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])

random.shuffle(training)
training = np.array(training)

train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")


model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd,
   metrics=['accuracy'])


hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5,
   verbose=1)
model.save('chatbot_model.h5', hist)

   print("model created")
```

3. **Chatgui.py file:**

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))

def clean_up_sentence(sentence):

    sentence_words = nltk.word_tokenize(sentence)

    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    return sentence_words

def bow(sentence, words, show_details=True):

    sentence_words = clean_up_sentence(sentence)

    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:

                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):

    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]

    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result
```

```
def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res


import tkinter
from tkinter import *

def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)


base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)

ChatLog.config(state=DISABLED)

scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12",
height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )

EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")


scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)
base.mainloop()
```