

AI INFUSED CHATBOT FOR TREATMENT OF MENTAL ILLNESS

A PROJECT REPORT

Submitted by

ASWIN J (814820104003)

BHARATH E (814820104006)

SATHISHKUMAR S (814820104038)

VASANTHA KUMAR G (814820104045)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING, ARIYALUR

(A Constituent College of Anna University, Chennai)



ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2024



ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report entitled "**AI INFUSED CHATBOT FOR TREATMENT OF MENTAL ILLNESS** " is the bonafide work of **ASWIN J (814820104003), BHARATH E (814820104006), SATHISHKUMAR S (814820104038), VASANTHA KUMAR R (814820104045)** who carried out the project work under my supervision.

SIGNATURE

Dr. S.JAYANTHI, M.Tech.,Ph.D.,

HEAD OF THE DEPARTMENT

Assistant Professor

Department of Computer Science & Engg

University College of Engineering

Ariyalur-621 731.

SIGNATURE

Mrs. M. SUGANTHI, M.E.,

SUPERVISOR

Teaching Fellow

Department of Computer Science & Engg

University College of Engineering

Ariyalur-621 731.

Submitted for the University viva voce Examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our breathless thanks to our **Dr.P.Senthamil Selvan, B.Pharm, M.Pharm, Ph.D.**, Dean(i/c), University College of Engineering, Ariyalur for giving constant motivation in succeeding in our goal.

We acknowledge our sincere thanks to **Dr.S.Jayanthi M.Tech., Ph.D.**, Head of the Department (i/c), for giving us valuable suggestion and help towards us throughout this project.

We are highly grateful to thank our project coordinator **Dr.S.Jayanthi M.Tech., Ph.D.**, and our project guide **Mrs.M Suganthi M.E.**, Department of Computer Science and Engineering, University college of Engineering, Ariyalur for the coordinating us throughout this project.

We are very much indebted to thank all the faculty members of Department of Computer science and Engineering in our Institute, for their excellent moral support and suggestions to complete our project work successfully.

Finally our acknowledgment does our parents, sisters and friends those who had extended their excellent support and ideas to make our project a pledge one.

ASWIN J

BHARATH E

SATHISHKUMAR S

VASANTHA KUMAR G

ABSTRACT

AI-powered chatbot for treatment of mental illness offer personalized and empathetic mental health support, adapting responses based on user input. They provide a user-friendly interface and a confidential platform for users seeking assistance. Despite global mental health challenges, timely treatment remains difficult to access. AI has emerged as a promising tool in augmenting mental health care, particularly through chatbot. These chatbot can screen, diagnose, monitor, educate, deliver therapy, and intervene in crises. Technologies like NLP, machine learning, and sentiment analysis enable chatbot to interact with users effectively. Challenges such as privacy concerns and ethical considerations persist. Collaboration is crucial for overcoming these challenges and maximizing chatbot potential in mental health care. Through continuous innovation, AI chatbot's hold promise for improving mental health outcomes globally.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ACKNOWLEDGEMENT	i
	ABSTRACT	ii
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1	INTRODUCTION	
	1.1 OVERVIEW OF THE PROJECT	1
	1.2 OBJECTIVE	1
	1.3 LITERATURE SURVEY	2
2	SYSTEM REQUIREMENTS	
	2.1 HARDWARE REQUIREMENTS	9
	2.2 SOFTWARE REQUIREMENTS	9
	2.3 SOFTWARE ENVIRONMENT	10
3	DESCRIPTION OF PROBLEM	
	3.1 PROBLEM DESCRIPTION	13
	3.2 EXISTING SYSTEM	14
	3.2.1 DISADVANTAGES	14
	3.3 PROPOSED SYSTEM	15
	3.3.1 ADVANTAGES	15
4	DESIGN OF THE SYSTEM	
	4.1 SYSTEM ARCHITECTURE	16
	4.2 ER DIAGRAM	17
	4.3 UML DIAGRAM	18
	4.3.1 USE CASE DIAGRAM	18
	4.3.2 SEQUENCE DIAGRAM	20
	4.3.3 CLASS DIAGRAM	21

	4.4 DATA FLOW DIAGRAM	22
	4.5 FLOWCHART	23
5	SYSTEM IMPLEMENTATION	
	5.1 MODULE	
	5.1.1 AJAX CALL METHOD	25
	5.1.2 DATA PRE PROCESSING	26
	5.1.3 VECTORIZATION AND EMBEDDING	27
	5.1.4 DATASET PREPARATION	28
	5.1.5 TRAINING THE MODEL	29
6	SOFTWARE TESTING	
	6.1 UNIT TESTING	30
	6.2 SYSTEM TESTING	30
	6.3 INTEGRATION TESTING	30
	6.4 BLACK AND WHITE BOX TESTING	31
	6.5 PERFORMANCE TESTING	31
	6.6 USABILITY TESTING	32
	6.7 TEST RESULTS	32
7	SCREENSHOTS	33
8	CONCLUSION	35
9	FUTURE ENHANCEMENT	36
10	APPENDIX	37
11	REFERENCE	60

LIST OF FIGURES

FIGURE NO	LIST OF FIGURES	PAGE NO
4.1	SYSTEM ARCHITECTURE	16
4.2	ER DIAGRAM	17
4.3	UML DIAGRAM	18
	4.3.1 USE CASE DIAGRAM	18
	4.3.2 SEQUENCE DIAGRAM	20
	4.3.3 CLASS DIAGRAM	21
4.4	DATA FLOW DIAGRAM	22
4.5	FLOW CHART	23

LIST OF ABBREVIATIONS

AI	- Artificial Intelligence
NLP	- Natural Language Processing
CBT	-Cognitive Behavioral Therapy
NLTK	- Natural Language Toolkit
Spacy	- Advanced NLP library
BERT	- Bidirectional Encoder Representations from Transformers
GPT	- Generative Pre-trained Transformer
ML	- Machine Learning
IDE	- Integrated Development Environments
SQLITE 3	- Structured Query Language Lite
NumPy	- Numerical Python
AJAX	- Asynchronous JavaScript and XML
API	- Application Programming Interface
BoW	- Bag-of-Words
ReLU	- Rectified Linear Unit
Softmax	- Softmax Activation Function
CC	- Categorical Crossentropy
Adam	-Adaptive Moment Estimation
NN	- Neural Network

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The integration of artificial intelligence (AI) into mental health chatbots presents a ground breaking approach to addressing global mental health challenges. These AI-infused chatbots leverage cutting-edge technologies like natural language processing (NLP) and machine learning to provide personalized support to individuals in need. By offering accessible and scalable mental health interventions, AI chatbots hold the promise of overcoming barriers such as stigma and geographical constraints.

They empower users to take control of their mental health by providing psychoeducation, coping strategies, and self-help tools. Despite their potential benefits, AI chatbots also face challenges, including privacy concerns and the need for rigorous validation. Ethical considerations surrounding the use of AI in mental health care must also be carefully addressed. Collaborative efforts between technologists, mental health professionals, and policymakers are crucial to ensure the ethical and effective implementation of AI chatbots.

1.2 OBJECTIVE

The primary objective of this project is to design, develop, and deploy an AI-infused chatbot tailored specifically for the treatment of mental illness. This chatbot will utilize cutting-edge artificial intelligence techniques to provide personalized support, intervention, and guidance to individuals experiencing mental health challenges. The overarching aim is to enhance accessibility, affordability, and effectiveness of mental health care by leveraging technology to deliver timely and evidence-based interventions. Implement AI algorithms to analyze user input, behavior, and historical data to deliver personalized support and interventions tailored to the individual's unique needs, preferences, and

challenges. Integrate evidence-based therapeutic techniques, such as Cognitive Behavioral Therapy (CBT) and mindfulness practices, into the chatbot's response algorithms to provide effective coping strategies and self-help tools.

1.3 LITERATURE SURVEY

[1]. Detection and analysis of stress-related posts in reddit's academic communities

In India, the prevalence of mental disorders is increasing over the year, at the same time the mental healthcare professionals shortage also in the rising trend. Smart technologies such as Artificial Intelligence (AI) play an important role in filling this Mental Health delivery gap. In this paper let us know about how the chatbot is one such technology used in mental healthcare delivery. Interestingly, chatbots are initially used mainly to deliver the mental health services such as psychotherapy, later is used in other industries also. The reviews show that Chatbots are widely used to manage anxiety, depression, stress and also to provide psychoeducation. However, it has its own limitation such as, it cannot think like a human with wisdom and empathy; and also the confidentiality of the data is very much serious concern. At the same time, these Chatbots will become an integral part of our lives in the coming years. And we need Chatbots that match our culture. In order, to benefit from this technological advancement, we should have a regulatory and assessment process in place

Advantages

- Accessible anytime: chatbots are digital robots that never get tired and keep running. Throughout the year, they will continue to operate every day without having to take a break
- Flexible attribute: chatbots have the benefit that it can quite easily be used in any industry

- Communication with the user is easy. Reduces human effort for reading files. Easy for maintenance
- Anonymity, privacy, and security could be maintained with a privacy-enhancing approach

Disadvantages

- Powell argues for the necessity of wisdom and empathic relationships in healthcare, suggesting that AI should supplement rather than replace health specialists.
- Many US doctors express skepticism about chatbots' ability to fulfill all patient needs, display human emotions, and provide comprehensive diagnosis and management.
- Clinicians raise concerns that patients may self-diagnose incorrectly and misunderstand diagnoses if they rely too heavily on chatbots for healthcare.

[2]. A mental health chatbot

A chatbot is a software that is used to establish interaction between a client/human and a PC/framework in a characteristic language like human talks. Chatbots talk with the user in a conversation as per the contribution of a human and reply to the user. Individuals are progressively utilizing voice aides and chatbots to associate with frameworks. The times of cooperating with an assistance just through a console are a distant memory. This exploration will see the way progresses in Artificial Intelligence and Machine Learning innovations are being utilized to work on an assortment of administrations. It will zero in on the development of chatbots as a mechanism for data appropriation specifically. To feature the essential differentiations between the two modalities of contact, spoken and message exchange, the chatbot was tried on an assortment of gadgets including Google Home and Assistant on Android gadgets. These analyses were directed to decide the utility of involving such advances

considering the new interest in first Year second Semester Independent Projects. Showing that chatbots might be utilized in a given space to further develop openness, reaffirming that they are in excess of a passing pattern with a down to earth application

Advantages

- Chatbots offer convenient and accessible platforms for individuals to seek support and information about mental health concerns.
- Chatbots provide a level of anonymity that encourages users to open up about their mental health struggles without fear of judgment or stigma.
- Users may feel more comfortable discussing sensitive topics with a non-human entity, preserving their privacy.
- Unlike traditional mental health services with specific operating hours, chatbots are available 24/7, providing immediate support even during non-business hours or weekends.

Disadvantages

- Chatbots cannot replace the empathy and understanding provided by human interaction, especially in dealing with sensitive mental health issues.
- Chatbots may struggle to accurately interpret the nuances of human language and emotion, potentially leading to misunderstandings or inappropriate responses.
- While chatbots can provide general advice and support, they may not be equipped to handle complex or severe mental health issues.
- Users with severe depression, anxiety disorders, or other serious conditions may require professional intervention and ongoing therapy, which a chatbot cannot provide.

[3]. Mental health chatbot with personalized recommendation system

This project aims to create a psychological chatbot website that offers personalized recommendations for users based on their mood and preferences. Users can express their feelings and receive motivational messages, beauty tips, movie and book recommendations, and more. The chatbot will suggest activities like music or meditation to help users manage stress. Additionally, a journal feature allows users to track their thoughts in real-time, aiding in identifying psychological patterns. By providing easily accessible and personalized support, the project aims to reduce stigma around mental health, improve self-awareness, and promote psychological development. The study is ongoing but shows promise in benefiting mental health through technology.

Advantages

- Mental health chatbots offer easy access to support, breaking down barriers to seeking help. Engagement and Motivation: Interactive features keep users engaged, fostering motivation in their mental health journey.
- Transparency about data privacy builds trust, ensuring users feel comfortable using chatbots. Robust measures protect user data, enhancing confidence in the reliability of chatbots.
- Identifying and mitigating biases promotes fairness and inclusivity in mental health support. Chatbots overcome geographical barriers, extending mental health support to underserved populations.
- Personalized recommendations enhance the effectiveness of mental health management. Chatbots offer cost-effective support, making mental healthcare more sustainable and accessible.

Disadvantages

- Chatbots may struggle to fully understand complex medical queries, leading to potential inaccuracies in information provided to users.
- Interactions with medical chatbots raise privacy and security risks for user data, with concerns about unauthorized access or misuse.
- Users may have skepticism or distrust towards chatbots, especially regarding sensitive health matters, despite efforts to build trust.
- Without professional oversight, there's a risk of misdiagnosis or inappropriate medical advice from chatbots, leading to adverse health outcomes.

[4]. Exploring the role of artificial intelligence in mental health

This paper introduces a Healthcare Chatbot powered by Artificial Intelligence (AI) to provide basic health advice, reducing the need for immediate doctor consultation and saving time and money. The Chatbot communicates with users using Natural Language Processing (NLP) and Sequential algorithms. Users can create profiles with symptoms, allowing the Chatbot to suggest doctors and send dosage reminders. It also offers mental health support, diagnosing diseases, suggesting remedies, and providing voice output music for relaxation and games. The Chatbot schedules appointments and offers various healthcare services like medical check-ups, therapy sessions, and lifestyle advice, including personalized fitness plans and mental health counseling.. Overall, the Healthcare Chatbot offers convenient and personalized healthcare services, potentially transforming the healthcare delivery system globally by providing accessible, affordable, and reliable care. . These analyses were directed to decide the utility of involving such advances considering the new interest in first Year second Semester Independent Projects.

Advantages

- The chatbot provides basic health information to users, offering easy access to healthcare knowledge without the need for appointments or waiting lists.
- By employing artificial intelligence and machine learning techniques, the chatbot can interpret natural language and respond to inquiries promptly, saving time for both users and healthcare professionals.
- Users can access the chatbot system anytime, anywhere, providing immediate assistance and support, even outside of traditional healthcare hours.

[5]. A chatbot system for mental health care

An artificial intelligence (AI) enabled feature which behaves similar as a human conversational partner, a Chatbot is a programme that uses natural language processing (NLP) and Machine learning (ML) techniques to analyse and understand the user's query and provide with intelligent responses that are human-like. Conversational agents that act as an intermediate between the user and the computer. The chatbot therapist is designed to provide relief to people suffering from mental illness (anxiety, depression or stress) with daily conversations and motivating them for better mental health using the cognitive behavioural therapy (CBT) method. CBT works by changing the thinking pattern and behaviour of people by working on their imagination, belief and attitude. These therapies are mostly free of cost and available anytime. People open up more in an online chat rather than a face to face conversation. Some of the conversational agents like Flow, Wysa, Woebot, joy and Talkspace are made available for the public sector. There is a need for the mechanism of using chatbots to make people understand about the options the technology is providing with and to maintain user motivation regularly.

Advantages

- Chatbot therapists like Woebot, Wysa, Joyable, and Talkspace provide accessible mental health support anytime, anywhere, through mobile devices and internet access.
- Chatbot therapy can be more affordable than traditional in-person therapy, making mental health support more accessible to a wider range of individuals.
- Users can engage with chatbot therapists at their own pace and schedule, without the need to make appointments or travel to a therapist's office.
- Platforms like Joyable offer structured programs, including mood tracking, activities, and one-on-one coaching, providing users with a comprehensive approach to mental health improvement.

Disadvantages

- Chatbot therapy relies on AI algorithms, which may not fully replicate the depth of human interaction and understanding provided by traditional therapy.
- While chatbot therapy can be effective for mild to moderate mental health issues, it may not be suitable for severe cases that require more intensive or personalized treatment.
- Users may become overly reliant on chatbot therapy and neglect seeking help from human professionals when necessary.
- Chatbot therapy may lack the empathy and emotional support that can be provided by human therapists, potentially leaving users feeling isolated or misunderstood in their struggles.
- Users may encounter technical glitches or limitations with chatbot platforms, affecting the quality and reliability of their therapy experience.

CHAPTER 2

2. SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should describe what the system should do and not how it should be implemented.

- Processor : Intel 7
- Ram : 8 GB (Minimum)
- Hard disk capacity : 50 GB

2.2 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a foundation for creating the software requirements specification. It is useful in estimating costs, planning team activities, performing tasks, and tracking the team's progress throughout the development activity.

- Operating System : Windows 11
- Front End : HTML,CSS
- Language : PYTHON
- Database : Sqlite 3

2.3 SOFTWARE ENVIRONMENT

Developing a software environment for an AI-infused mental health chatbot requires careful consideration of the interplay between programming languages, frameworks, libraries, and tools. Each component must harmonize seamlessly to facilitate the chatbot's understanding of complex human emotions and nuances in language. By leveraging a meticulously crafted combination of these elements, developers can empower the chatbot to provide empathetic and effective support to individuals seeking mental health assistance.

Programming languages

Python has become central in AI and NLP due to its flexibility and vast library support. Libraries like TensorFlow empower developers with tools for diverse tasks such as text classification and sentiment analysis. Python's clear syntax fosters collaboration and accelerates development, while its active community ensures ample resources for assistance. Its seamless integration with web frameworks and databases cements its position as the top choice for building empathetic AI chatbots tailored to mental health support, thanks to its adaptability and strong support network.

Natural Language Toolkit (NLTK):

NLTK (Natural Language Toolkit) offers a wide array of tools for processing text data, including tasks like tokenization, stemming, lemmatization, and parsing. It's a go-to choice for developers working on NLP projects, providing essential capabilities to extract insights from text.

Spacy is known for its efficient processing speed and advanced features like part-of-speech tagging, named entity recognition, and dependency parsing. It's ideal for tasks requiring real-time or large-scale NLP processing.

Transformers, developed by Hugging Face, utilize transformer-based architecture to achieve state-of-the-art performance in NLP tasks like text classification and sentiment analysis. These pre-trained models, such as BERT and GPT, revolutionize NLP by offering cutting-edge capabilities accessible to developers, including those working on mental health chatbots.

Machine Learning (ML):

Tensorflow is renowned for its flexibility in building and training neural networks, especially for NLP tasks. Its architecture supports various neural network types like RNNs, CNNs, and transformers, crucial for NLP.

One of Tensorflow's strengths is its scalability, suitable for projects of any size, from small experiments to large-scale deployments. Its distributed computing capabilities enable efficient training on massive datasets, essential for achieving high performance in tasks like language translation and sentiment analysis.

Additionally, Tensorflow offers pre-trained models and modules tailored for NLP tasks, speeding up development. Integration with other components like Tensorflow Serving facilitates seamless deployment across different platforms, ensuring accessibility for various deployment needs.

Development Tools:

Google Colab serves as a cloud-based hub, offering free access to GPUs and TPUs crucial for accelerating deep learning model training. With its browser-based interface, users can write and execute Python code seamlessly, eliminating the hassle of local setup. Colab's support for popular libraries like Tensorflow simplifies experimentation, making it an ideal platform for AI researchers and developers.

Visual Studio Code (VS Code) stands out as a feature-rich IDE, providing a suite of tools for coding, debugging, and testing. Its integrated terminal and Git integration streamline the development workflow, enhancing productivity.

The extensive ecosystem of extensions allows customization to support various programming languages and tasks, making it a versatile choice for developers across different domains.

Git, a powerful version control system, enables developers to track code changes, revert to previous versions, and merge contributions from multiple collaborators seamlessly. Coupled with hosting platforms like GitHub, Git facilitates collaborative development and version control. GitHub's features, including pull requests and code reviews, foster transparency and communication within development teams, ensuring code quality and facilitating effective collaboration.

Data Storage And Management:

SQLite 3 serves as an efficient database solution for storing user data and chat logs in AI-infused chatbot applications. Operating locally, SQLite eliminates the need for a separate database server, simplifying deployment. Its support for standard SQL queries and transactions ensures data integrity, making it reliable for chatbot applications. With its lightweight and self-contained file structure, SQLite is ideal for projects with moderate data storage needs, such as managing user profiles and conversation histories.

CHAPTER 3

3. DESCRIPTION OF THE PROBLEM

3.1 PROBLEM DESCRIPTION

Accessing effective mental health treatment is hindered by stigma, resource limitations, and barriers to care. Traditional therapy models struggle with cost, availability, and stigma, leaving many without help. Healthcare systems are overwhelmed, leading to long wait times and inadequate support. Underserved communities face additional challenges accessing care due to geographical and financial constraints. One-size-fits-all approaches don't meet diverse needs effectively. Personalized interventions are crucial for optimal outcomes. Scalable solutions are needed to reach more people and provide timely support without compromising quality. There's a pressing need for innovation in mental health care delivery to bridge the gap in accessibility and personalization. Technology can play a significant role in overcoming these challenges by offering remote and digitally-based interventions. Community-based initiatives and peer support networks can also provide valuable support and connection for individuals facing mental health challenges. Increased awareness and destigmatization efforts are essential to encourage more people to seek help and access available resources. Collaboration between governments, healthcare providers, technology developers, and community organizations is vital to creating comprehensive and inclusive mental health support systems.

3.2 EXISTING SYSTEM

Existing chatbot systems struggle with disjointed interactions and misunderstandings due to their inability to recall past conversations, limiting their effectiveness in providing continuous and personalized support. Users often experience dissatisfaction or upset with chatbot responses as a result of the

system's limited understanding of their emotions and concerns, highlighting the need for improved empathy and comprehension in AI-driven interactions. The chatbot's generic treatment approach, though broadly applicable, fails to account for the diverse and unique needs of users across different age groups, suggesting the necessity for customized interventions tailored to individual circumstances and preferences.

3.2.1 Disadvantages

Lack of Continuity: Disjointed interactions and potential misunderstandings arise due to the chatbot's inability to recall past conversations, hindering effective communication and support provision.

Limited Understanding: Users may feel frustrated or dissatisfied with chatbot responses as they fail to grasp the nuances of their emotions and situations, leading to a lack of empathy and personalized support.

Generic Treatment Approach: The use of a one-size-fits-all treatment approach overlooks the unique needs and preferences of individual users, resulting in suboptimal outcomes and reduced effectiveness in addressing their specific mental health concerns

3.3 PROPOSED SYSTEM

The proposed system will incorporate a memory feature to recall past conversations, ensuring continuity in interactions and allowing the chatbot to better understand the user's context and history. The system will prompt users to provide demographic information such as age, gender, and location. This data will enable the chatbot to personalize its responses and interventions according to the user's specific demographic profile. Leveraging the demographic information obtained, the system will propose targeted therapies that take into account age-specific considerations. For example, it may recommend cognitive-

behavioral techniques for adolescents, mindfulness practices for adults, and lifestyle modifications for seniors, ensuring that interventions are tailored to the unique needs and developmental stages of each age group.

3.3.1 Advantages

Enhanced User Engagement: Continuity in conversations fosters a sense of familiarity and trust, encouraging users to engage more openly with the chatbot over time, leading to deeper insights into their mental health needs and improved treatment adherence.

Customized Support: Personalized interventions cater to the unique characteristics and preferences of each user, increasing the relevance and effectiveness of the support provided, leading to greater user satisfaction and a higher likelihood of positive outcomes.

Tailored Approach: By considering age-specific factors in treatment recommendations, the system ensures interventions are well-suited to the developmental stage and challenges faced by users, maximizing treatment efficacy and promoting better overall mental health outcomes across diverse age groups.

CHAPTER 4

4. DESIGN OF THE SYSTEM

4.1 SYSTEM ARCHITECTURE

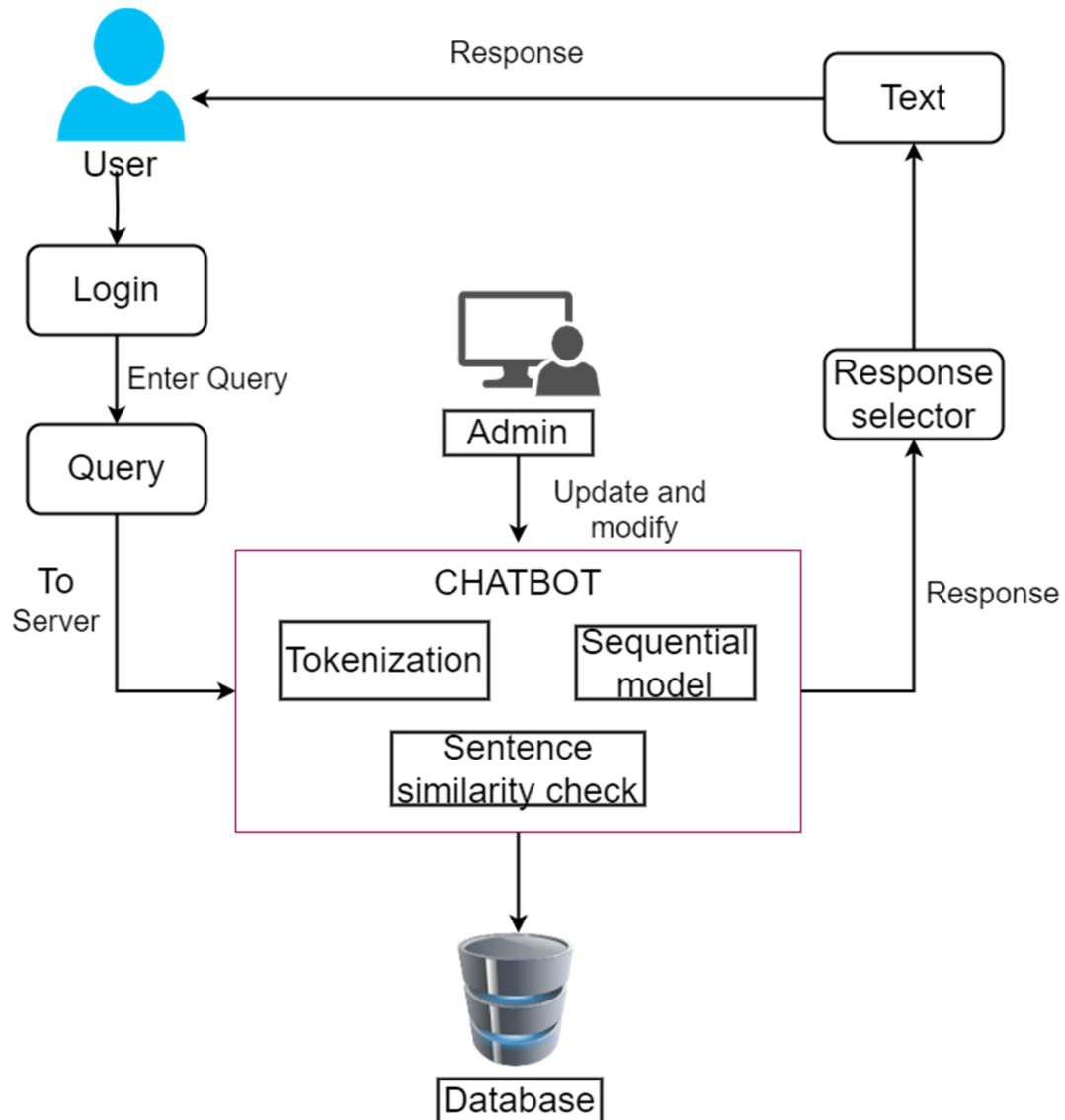


Fig 4.1.1: Architecture Diagram

4.2 ER DIAGRAM

An ER (Entity-Relationship) diagram serves as a graphical depiction of a database's structure, displaying entities as rectangles representing objects or concepts like people or events, with attributes such as "Name" or "Age" depicted within. Relationships between entities are illustrated by lines, showcasing how they're interconnected, whether in one-to-one, one-to-many, or many-to-many relationships, and cardinality symbols like "1" for one or "M" for many indicate the quantity of instances associated between entities. ER diagrams are instrumental in visually conveying database designs, facilitating comprehension of system architectures and fostering accurate database development.

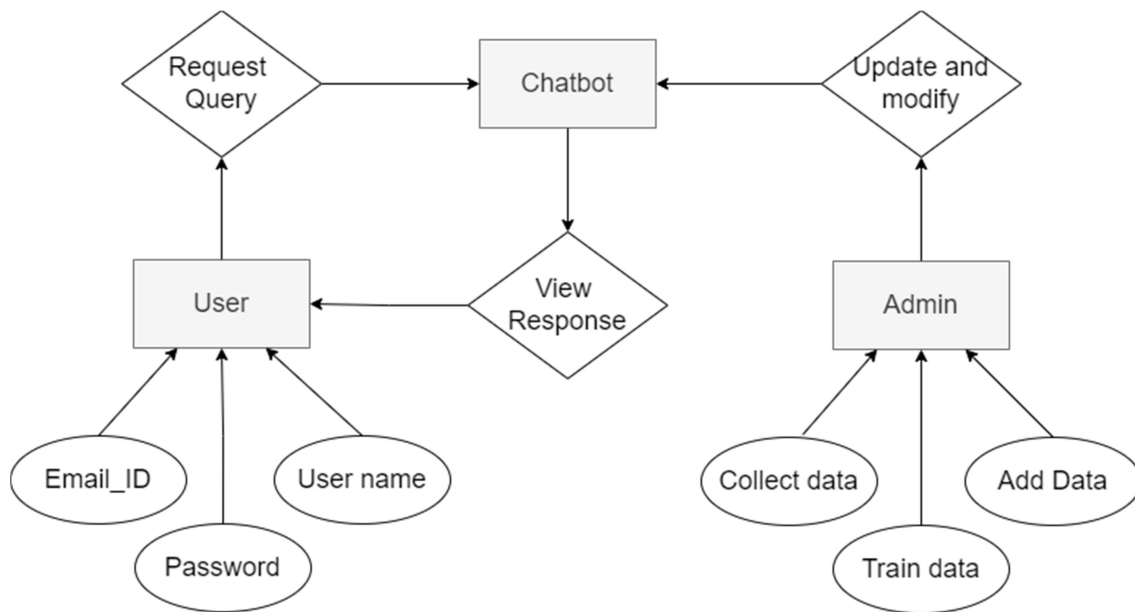


Fig 4.2.1: ER diagram

4.3 UML DIAGRAMS

A Unified Modeling Language (UML) diagram is like a map for software developers. It uses symbols and shapes to represent different parts of a computer program. For example, a class diagram shows the different types of things in the program, like buttons or text boxes. A use case diagram shows how people or other programs interact with the program. Sequence diagrams show the order in which things happen, like when one part of the program talks to another. There are many types of UML diagrams, each with its own purpose, such as showing how a program works or how different parts of it fit together. Overall, UML diagrams help software developers understand, plan, and build computer programs more effectively.

4.3.1 Use Case Diagram

A use case diagram visually outlines how users interact with a system to accomplish specific tasks. Users are represented as stick figures, while actions they perform are shown as ovals labeled with task descriptions like "log in" or "send a message." Lines connecting users to actions illustrate the flow of interaction, depicting how users navigate through system functionalities. This diagram aids stakeholders, designers, and developers in understanding system functionality and aligning it with user needs. It serves as a roadmap for system design and development, ensuring the final product meets user requirements. Additionally, it facilitates communication among project stakeholders by providing a common visual representation of system functionality and user interactions.

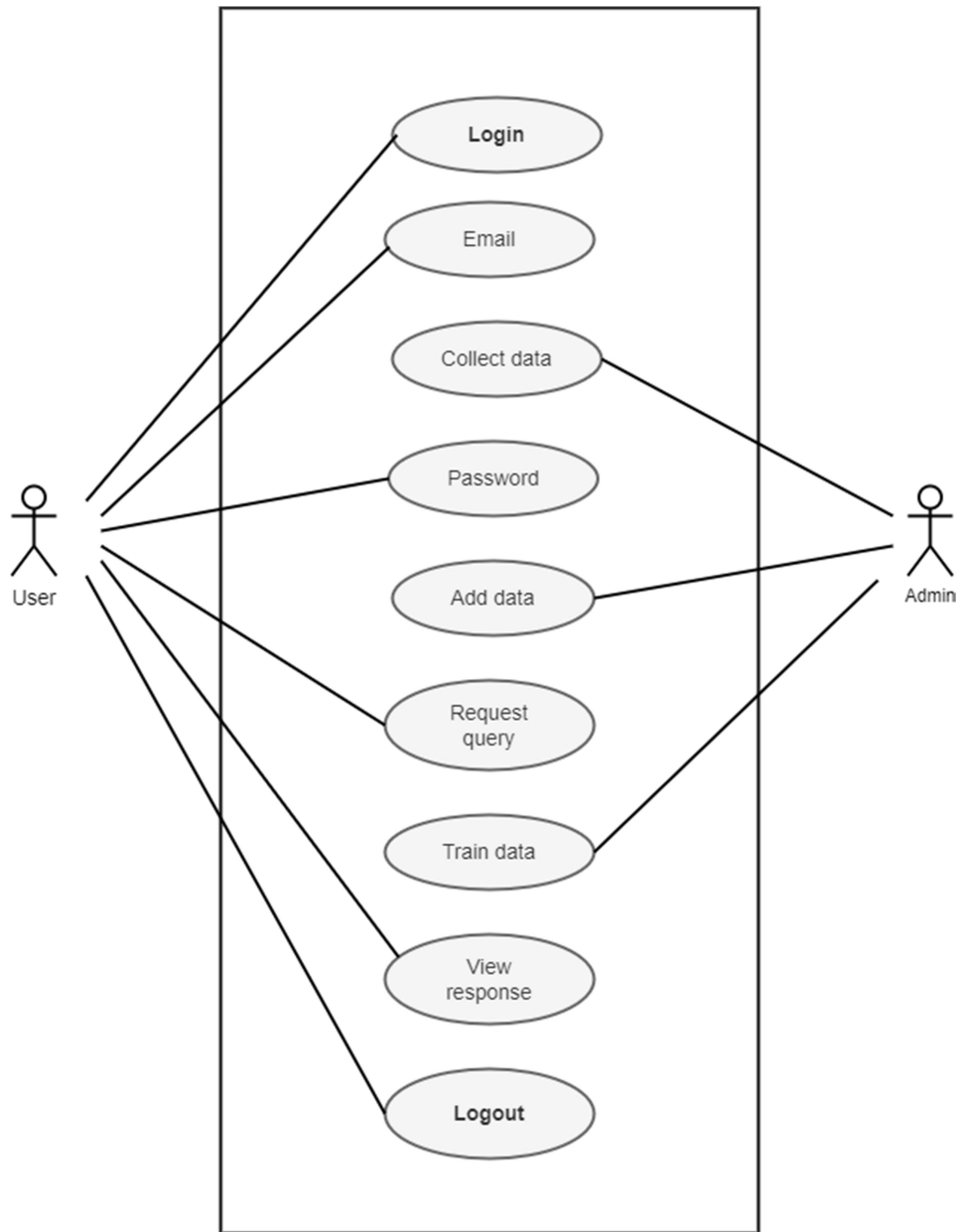


Fig 4.3.1.1: Use Case diagram

4.3.2 Sequence Diagram

A sequence diagram is a type of Unified Modeling Language (UML) diagram that depicts the interactions between objects or components in a system over time. It illustrates the flow of messages between objects or components in a chronological sequence, showing how they collaborate to accomplish a specific task or scenario. In a sequence diagram, objects are represented as boxes or lifelines, and messages exchanged between them are depicted as arrows. The sequence of messages and their order of execution are shown along the vertical axis, while time progresses from top to bottom. Sequence diagrams help visualize the dynamic behavior of a system, facilitating the understanding of system functionality, message passing, and the sequence of events during execution.

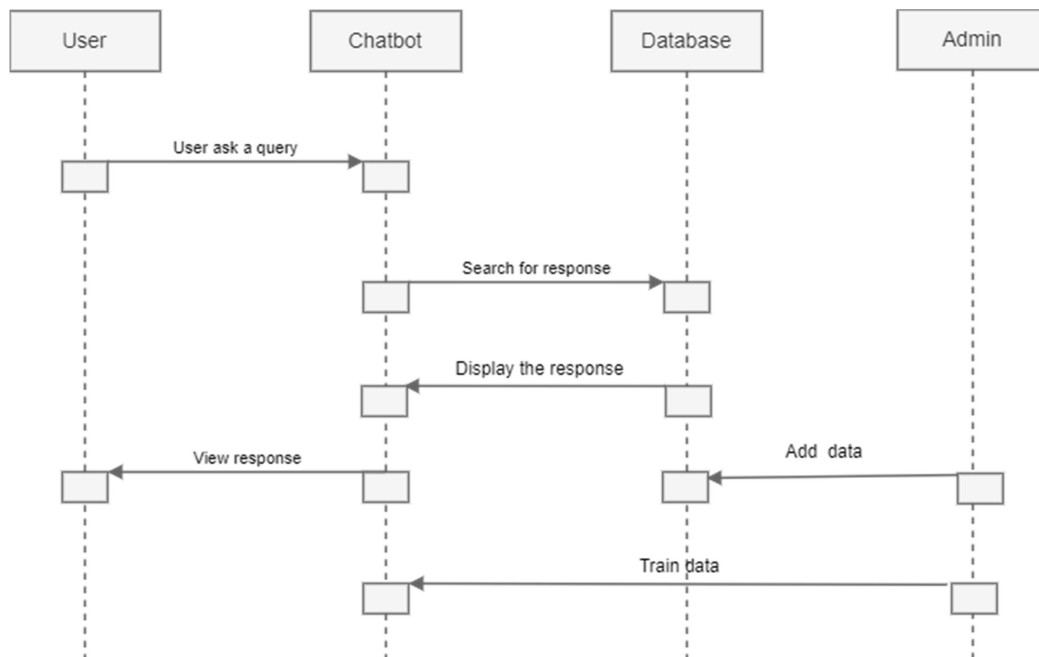


Fig 4.3.2.1: Sequence Diagram

4.3.3 Class Diagram

A class diagram is a type of Unified Modeling Language (UML) diagram that represents the structure of a system by illustrating the classes, attributes, methods, and relationships between them. It provides a visual overview of the objects and their interactions within the system. Classes are depicted as boxes, with each box containing the class name, attributes (variables), and methods (functions). Relationships between classes, such as associations, aggregations, and inheritances, are represented by lines connecting the classes, with arrows indicating the direction of the relationship. Class diagrams help software developers understand the structure of a system, design new systems, and communicate system architecture to stakeholders. They serve as a blueprint for designing and implementing object-oriented software systems.

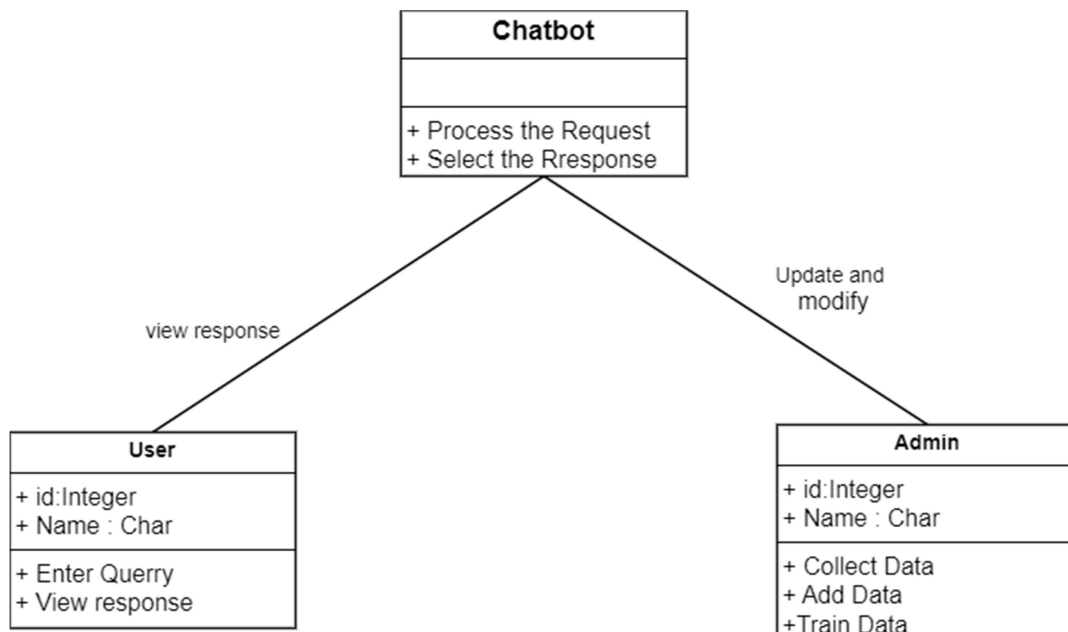


Fig 3.3.3.1: Class Diagram

4.4 DATAFLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation that illustrates the flow of data within a system. It shows how data moves from one part of the system to another, starting from its source, through various processes, and finally to its destination. DFDs consist of four main components: external entities, processes, data stores, and data flows. External entities represent sources or destinations of data, such as users or other systems. Data flows represent the movement of data between different components of the system. DFDs help software designers and developers understand the data flow and structure of a system, aiding in the analysis, design, and documentation of software systems.

Level 0

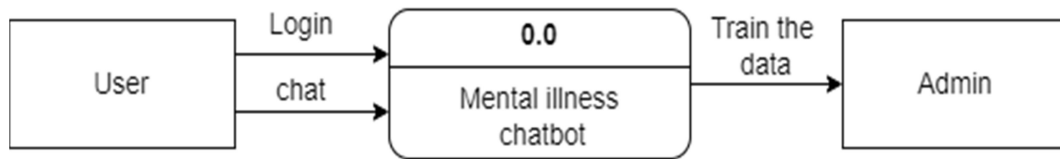


Fig 4.4.1: Data Flow Diagram

Level 1

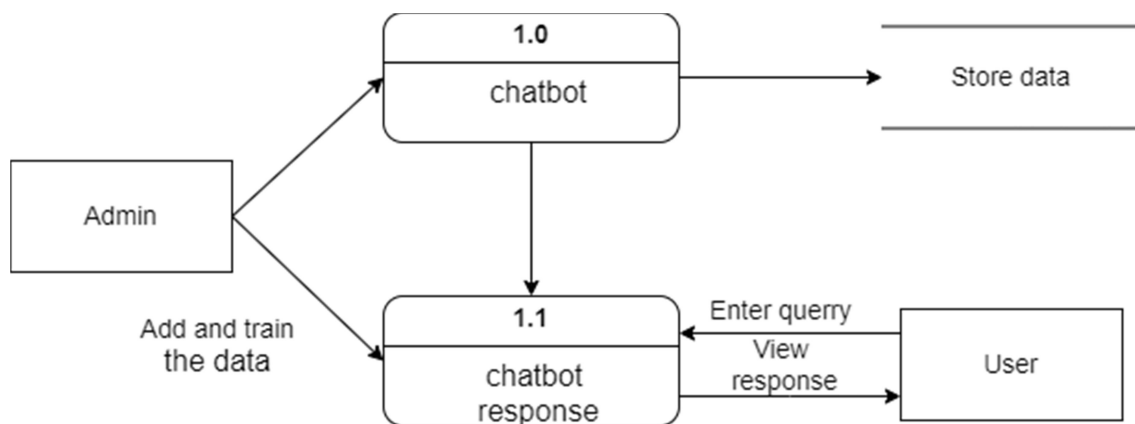


Fig 4.4.2:Data Flow Diagram

Level 2

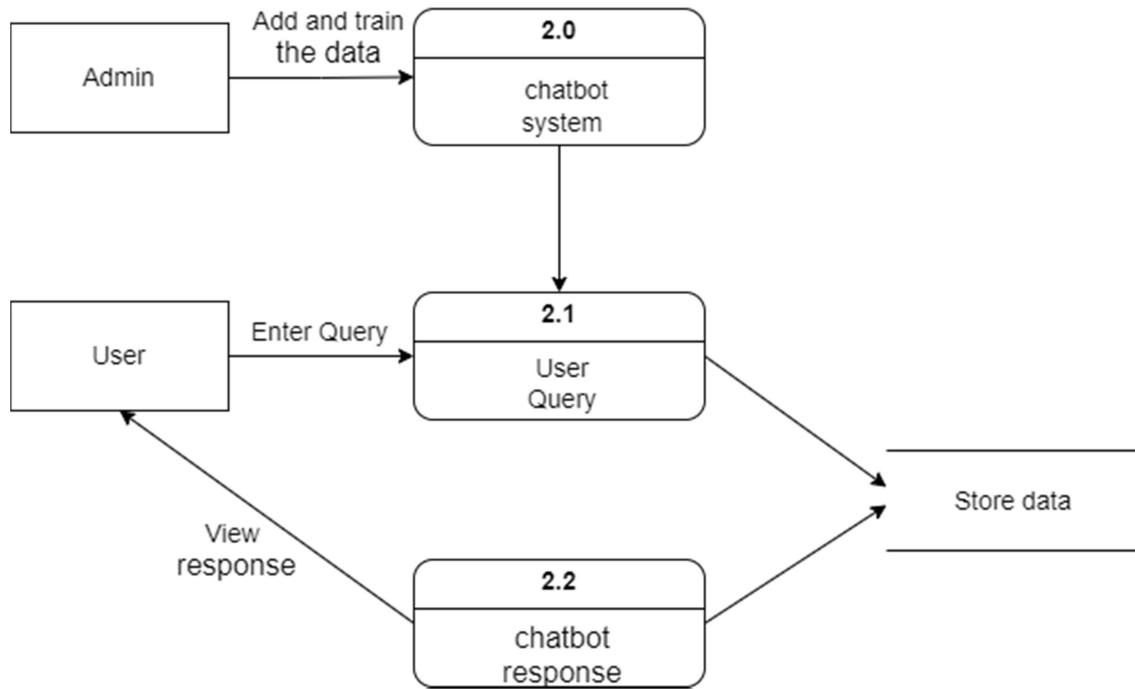


Fig 4.4.3:Data Flow Diagram

4.5 FLOWCHART

An ER diagram, short for Entity-Relationship diagram, is a visual tool used in database design to illustrate how different entities in a database relate to each other. Entities are represented as rectangles, each having attributes associated with it, while relationships between entities are shown as lines connecting them. These relationships can be one-to-one, one-to-many, many-to-one, or many-to-many. ER diagrams provide a clear overview of the structure of a database, helping designers to plan and implement databases effectively by understanding the connections between data entities.

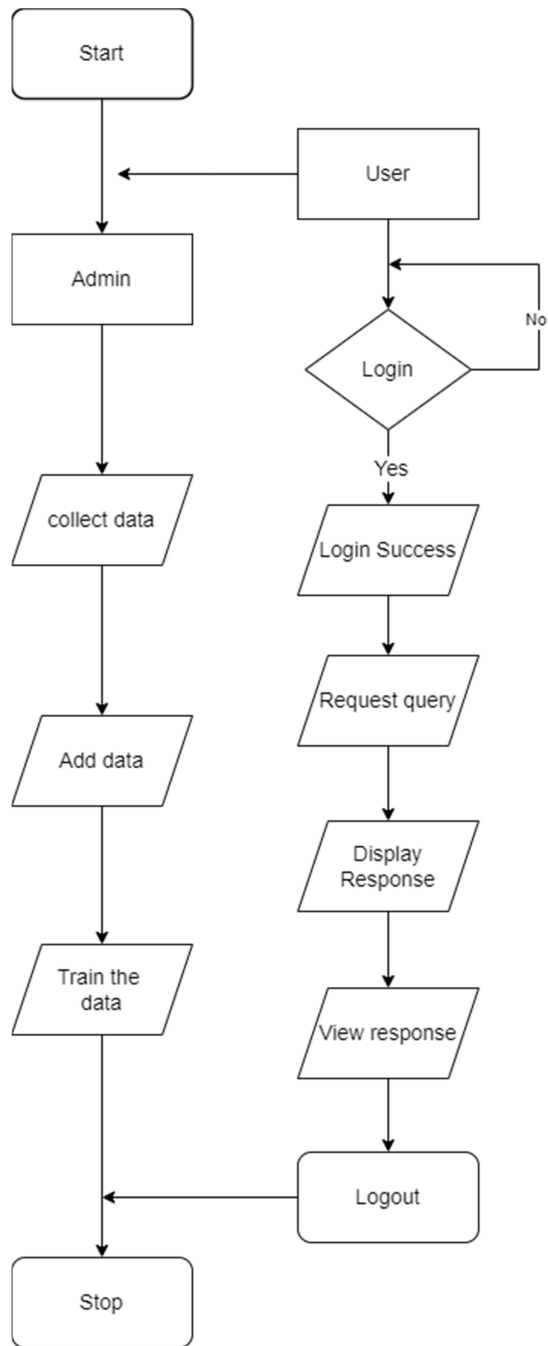


Fig 4.5.1Flow Chart

CHAPTER 5

5. SYSTEM IMPLEMENTATION

5.1 MODULE

- **Ajax Call Method**
- **Data Pre Processing**
- **Vectorization and Embedding**
- **Dataset preparation**
- **Training the Model**

5.1.1 AJAX CALL METHOD

Retrieve and Display Message:

When a user inputs a message into the chat interface and submits it, the system promptly retrieves the message. This retrieved message is then displayed within the interface, allowing the user to see their own input in real-time. This immediate display provides users with instant feedback, confirming that their message has been successfully received. It enhances the interactive experience by enabling users to review and refine their messages before continuing the conversation. This transparency in communication fosters a sense of control and engagement, facilitating effective interaction with the chatbot. Overall, this process contributes to a seamless and user-friendly chat experience.

Sending Request to Server:

After retrieving the user's message, the system initiates an AJAX request to send it to the server for processing. AJAX, or Asynchronous JavaScript enables data transmission without reloading the entire webpage, ensuring a seamless user experience. This asynchronous communication enhances responsiveness and efficiency by allowing the chat interface to update dynamically. Leveraging AJAX, the chat interface maintains interactivity

without disrupting the user's flow, contributing to a smooth and uninterrupted experience.

Receive Response:

Upon receiving the message, the server processes it based on predefined logic or algorithms. Once the server completes processing, it generates a response containing relevant data or information. This response data is then sent back to the client-side application, specifically to the chat interface.

Appending Response Data to the Chat Interface:

The response data received from the server is appended to the chat interface. This allows the user to view the server's response within the same conversation thread, enabling a continuous flow of communication. The response data could include text, images, links, or any other relevant content depending on the nature of the user's query and the server's capabilities.

5.1.2 DATA PRE-PROCESSING

Tokenization:

Tokenization is a crucial step in NLP, breaking down text into smaller units called tokens, which can be words, numbers, or punctuation. These tokens help computers understand human language by providing a structured representation of textual data. By segmenting text into tokens, NLP models can perform tasks like text classification and sentiment analysis more effectively. Ultimately, tokenization enables efficient analysis of text data, laying the foundation for various NLP applications.

Text Refinement:

Text refinement involves pre-processing the tokens obtained from tokenization to improve the quality of the text data for analysis. This step typically includes converting all tokens into lowercase to ensure consistency in text matching and analysis. Additionally, it involves removing unnecessary

characters such as punctuation marks, white spaces, and special characters that do not contribute to the semantic meaning of the text.

Stemming:

Stemming is a linguistic normalization technique used to reduce words to their base or root forms, known as stems. The purpose of stemming is to condense different variations of words into a common base form, thereby reducing the complexity and redundancy of the text data. This process helps in achieving better text analysis results by treating similar words with the same root form as identical, even though they may appear in different inflected or derived forms.

5.1.3 VECTORIZATION AND EMBEDDING

Vectorization:

Vectorization is a crucial step in natural language processing (NLP) where textual data is converted into numerical vectors that machine learning models can understand and process. In this process, each word or token in the text is mapped to a unique numerical value or position in a high-dimensional vector space. Vectorization enables computational models to perform mathematical operations on textual data, allowing them to analyze and make predictions based on the underlying patterns in the text. It facilitates various NLP tasks such as text classification, sentiment analysis, and information retrieval.

Bag-of-Words (BoW) Representation:

The Bag-of-Words (BoW) representation is a fundamental technique in NLP, where a document or piece of text is represented as a collection of word counts or frequencies. In this approach, the order of words is disregarded, and only the presence or absence of words in the text is considered. BoW representation treats each document as a "bag" of words, capturing the overall distribution of words across the text. Despite its simplicity, BoW representation

is widely used in text processing tasks due to its efficiency and effectiveness in capturing basic textual information.

Sequence Padding:

Sequence padding is a crucial pre-processing step in natural language processing (NLP) aimed at standardizing the length of input sequences. It ensures that all sequences have uniform dimensions, which is essential for machine learning models' proper functioning. By adding special tokens, typically zeros, to either the beginning or end of sequences, padding makes them consistent in length. This uniformity facilitates efficient batch processing and computation within the model, enabling streamlined training and inference processes. Ultimately, sequence padding helps maintain data integrity and improves the model's performance by ensuring consistent input shapes across all samples.

5.1.2 DATASET PREPARATION

The script reads intents from a JSON file, where each intent contains patterns (user inputs) and tags (intent labels). It tokenizes each pattern, lemmatizes and lowercases each word, and removes punctuation. This process prepares the data for training a chatbot to recognize user intents effectively. The processed data is organized into lists: words (unique lemmatized and lowercased words), classes (unique intent tags), and documents (pairs of tokenized patterns and intent tags). These lists serve as the vocabulary, intent categories, and training data for the chatbot, enabling it to understand user queries and provide appropriate responses.

5.1.4 Training the Model

The neural network architecture comprises three dense layers, signifying fully connected layers. The first layer is composed of 128 neurons, facilitating the model's ability to capture intricate patterns within the input data. Following

this, the second layer consists of 64 neurons, providing a more condensed representation space. Finally, the last layer is configured with the same number of neurons as the dataset's classes, empowering the model to predict class probabilities for each input. To mitigate overfitting, dropout layers are strategically incorporated after each dense layer. These dropout layers randomly deactivate a portion of neurons during training, compelling the network to learn more robust features and diminishing reliance on specific neurons.

Initially, Stochastic Gradient Descent (SGD) optimizer is employed for training. SGD is a widely-used optimization algorithm that iteratively updates the model weights to gradually minimize the loss function. This approach is well-suited for various applications and serves as a solid starting point for model training. Throughout the training process, which spans 200 epochs, the model refines its weights to minimize the discrepancy between predicted and actual class labels. Upon the completion of training, the model is preserved using the `model.save()` function provided by Keras. This action stores both the trained weights and architecture in an HDF5 file format, ensuring the model's integrity and readiness for deployment.

CHAPTER 6

6. SOFTWARE TESTING

6.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. It ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.3 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures

caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level - interact without error.

6.4 WHITE BOX AND BLACK BOX TESTING

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.5 PERFORMANCE TESTING

Performance testing evaluates a system's responsiveness, scalability, and reliability under various conditions, such as load, stress, and concurrency. It measures key metrics like response time, throughput, and resource utilization to identify bottlenecks and performance issues. Load testing ensures the system can handle anticipated user and transaction volumes without degradation. Stress testing pushes the system to its limits to assess its resilience and recovery capabilities. Endurance testing evaluates stability under sustained load, uncovering issues like memory leaks. Scalability testing assesses the system's ability to adapt to workload changes without sacrificing performance. Volume testing ensures efficient handling of large data volumes. Concurrency testing identifies contention issues and race conditions. Overall, performance testing is

crucial for ensuring software meets performance requirements and delivers optimal user experiences.

6.6 USABILITY TESTING

Usability testing is a crucial phase in the software development lifecycle aimed at evaluating the ease of use and user-friendliness of a software application, website, or product. This testing process involves observing real users as they interact with the software to perform specific tasks, identify usability issues, and gather feedback on their experience. The primary goal of usability testing is to assess the effectiveness, efficiency, and satisfaction of users when using the software, ultimately aiming to improve the overall user experience. Usability testing typically involves the following steps: defining test objectives and scenarios, recruiting representative users, conducting test sessions, observing user interactions, collecting qualitative and quantitative data, and analyzing the findings to identify usability problems and areas for improvement.

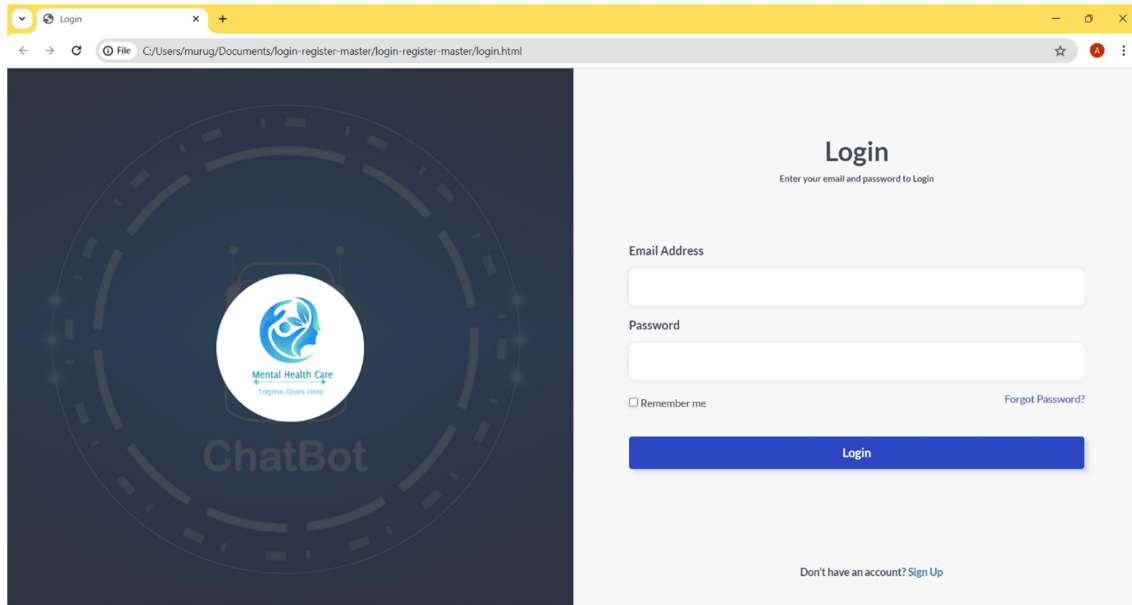
6.7 TEST RESULT

All test cases were successfully completed without encountering any issues. This indicates thorough testing, ensuring all expected outcomes were achieved. The absence of defects means the system functions as intended, boosting confidence in its reliability. This successful outcome reflects effective software development, guaranteeing a high-quality and dependable final product. Overall, completing the test cases is a positive step, signaling readiness for deployment.

CHAPTER 7

7. SCREEN SHOTS

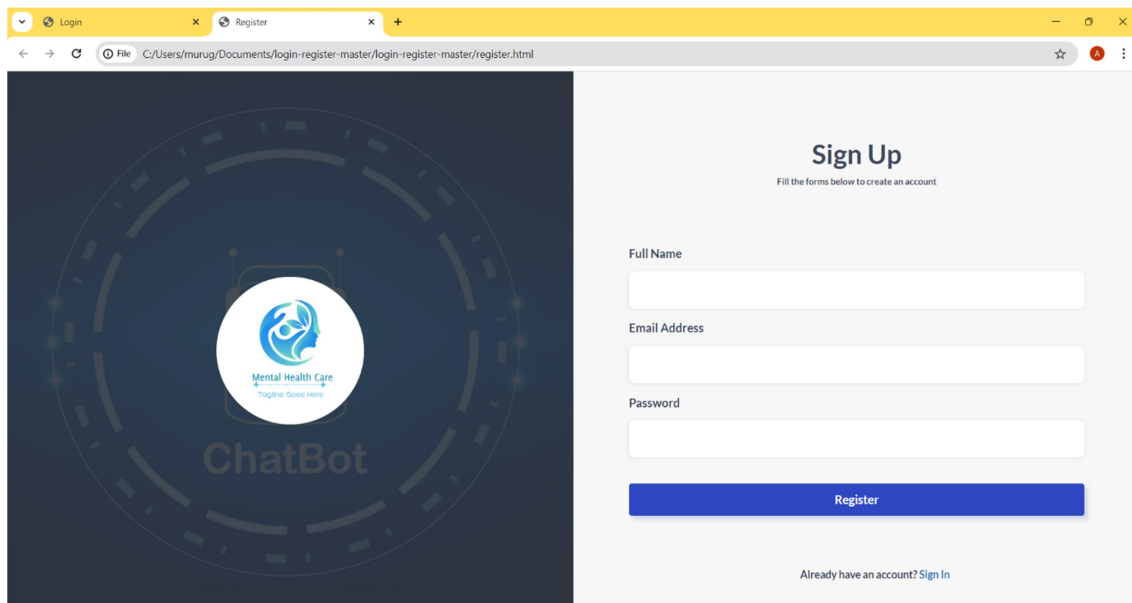
Login



The screenshot shows a web browser window with a yellow header bar. The address bar displays the file path: C:/Users/murug/Documents/login-register-master/login-register-master/login.html. The page is split into two main sections. On the left, there is a dark blue background with a large, faint circular graphic containing a white circle with a blue logo and the text 'Mental Health Care' and 'ChatBot'. On the right, the 'Login' form is displayed. It has a title 'Login' and a subtitle 'Enter your email and password to Login'. The form includes two input fields: 'Email Address' and 'Password'. Below the 'Password' field, there is a checkbox labeled 'Remember me' and a link 'Forgot Password?'. A blue 'Login' button is positioned below the input fields. At the bottom of the form, there is a link 'Don't have an account? Sign Up'.

Fig 7.1.1: Login

Register



The screenshot shows a web browser window with a yellow header bar. The address bar displays the file path: C:/Users/murug/Documents/login-register-master/login-register-master/register.html. The page is split into two main sections. On the left, there is a dark blue background with a large, faint circular graphic containing a white circle with a blue logo and the text 'Mental Health Care' and 'ChatBot'. On the right, the 'Sign Up' form is displayed. It has a title 'Sign Up' and a subtitle 'Fill the forms below to create an account'. The form includes three input fields: 'Full Name', 'Email Address', and 'Password'. A blue 'Register' button is positioned below the input fields. At the bottom of the form, there is a link 'Already have an account? Sign In'.

Fig 7.1.2 : Register

Chat Interface:

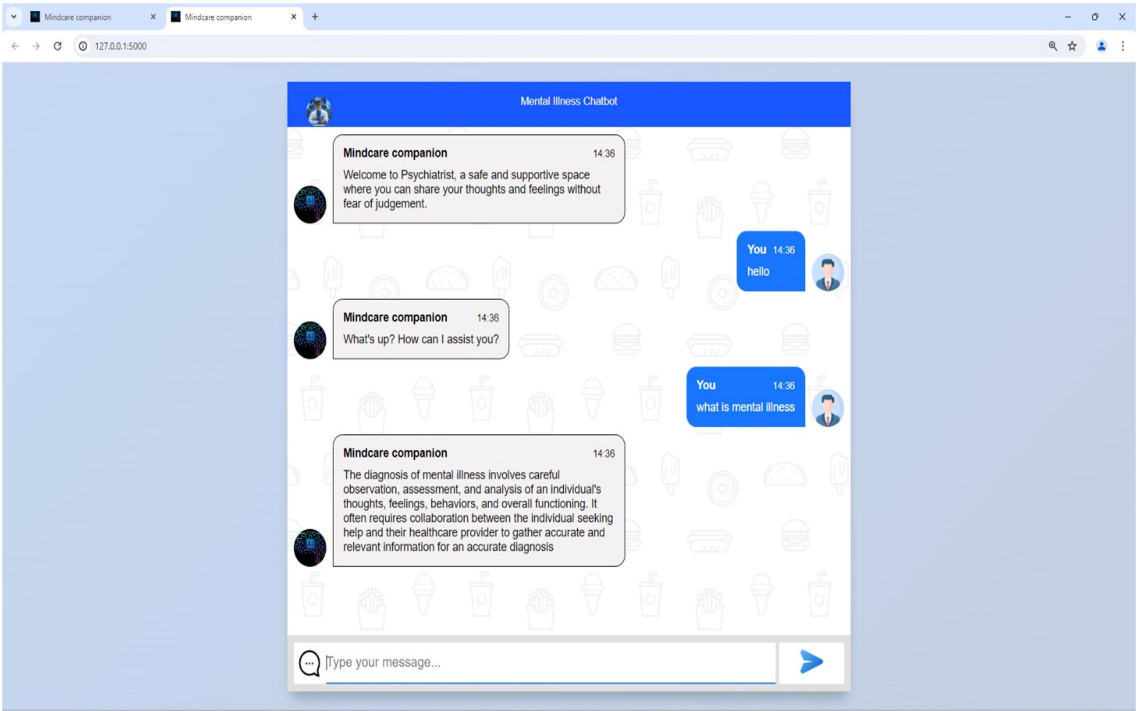


Fig 7.1.3 : Chat Interface

CHAPTER 8

8. CONCLUSION

In conclusion, the development and implementation of an AI-infused chatbot for mental illness treatment present a promising solution to address the growing demand for accessible and personalized mental health support.

This project utilizes Python and employs deep learning techniques such as Sequential Model and NLP to improve the chatbot accuracy

By leveraging artificial intelligence, natural language processing, and evidence-based therapeutic techniques, such a chatbot can provide continuous and scalable assistance to individuals grappling with mental health challenges.

Through features like memory recall, personalized interventions, and targeted therapies based on age-specific considerations, the chatbot can offer a holistic approach to mental wellness. Additionally, its ability to maintain continuity in conversations, understand user emotions, and recommend tailored treatment options enhances user engagement and satisfaction.

CHAPTER 9

9. FUTURE ENHANCEMENT

Integrating capabilities for voice, video, and image-based communication to offer more diverse and inclusive support options, accommodating users with different communication preferences and needs.

Incorporating advanced emotion recognition algorithms to detect and understand user emotions from text, voice tone, and facial expressions, enabling the chatbot to provide more empathetic and personalized responses.

Implementing features for real-time monitoring of user sentiment and mental health indicators, allowing the chatbot to proactively intervene or escalate support when signs of distress or crisis are detected.

Providing users with the ability to customize their treatment plans based on their preferences, goals, and progress, fostering a sense of autonomy and empowerment in managing their mental health journey.

CHAPTER 10

10.APPENDIX

SAMPLE CODING

App.py

```
from database import database1
import nltk
nltk.download('popular')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
from flask import Response
from keras.models import load_model
model = load_model('model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('texts.pkl','rb'))
classes = pickle.load(open('labels.pkl','rb'))
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the
sentence
```

```

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)

    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)

    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)

    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    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]

    # sort by strength of probability
    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):
    if ints:
        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
    else:
        return "Sorry, I didn't understand that."

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

from flask import Flask, render_template, request
app = Flask(__name__)
app.static_folder = 'static'
@app.route("/")
def home():
    return render_template("index.html")
@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)
@app.route('/register',methods=["GET","POST"])
def reg():
    if request.method=="POST":
        con=database1()
        cur=con.cursor()
        name=request.form["name"]
        pas=request.form["password"]
        cur.execute('insert into users(NAME,PASSWORD)

```

```

values(?,?)',(name,pas))
    con.commit()
    con.close()
    return Response("success")
else:
    return render_template('login.html')
if __name__ == "__main__":
    app.run(debug=True)

```

training.py

```

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
import tensorflow as tf
from keras.losses import CategoricalCrossentropy
from keras.optimizers import Adam
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']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))
        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmatize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in
ignore_words]
words = sorted(list(set(words)))

# sort classes
classes = sorted(list(set(classes)))

# documents = combination between patterns and intents
print (len(documents), "documents")

# classes = intents
print (len(classes), "classes", classes)

# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)

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

# create our training data
training = []

# create an empty array for our output
output_empty = [0] * len(classes)

# training set, bag of words for each sentence

```

for doc in documents:

 # initialize our bag of words

 bag = []

 # list of tokenized words for the pattern

 pattern_words = doc[0]

 # lemmatize each word - create base word, in attempt to represent related words

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

 # create our bag of words array with 1, if word match found in current pattern for w in words:

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

 # output is a '0' for each tag and '1' for current tag (for each pattern)

 output_row = list(output_empty)

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

 training.append([bag, output_row])

shuffle our features and turn into np.array

random.shuffle(training)

training = np.array(training, dtype=object)

create train and test lists. X - patterns, Y - intents

train_x = list(training[:,0])

train_y = list(training[:,1])

print("Training data created")

Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons

equal to number of intents to predict output intent with softmax

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'))
# Compile model. Stochastic gradient descent with Nesterov accelerated
gradient gives good results for this model
model.compile(loss=CategoricalCrossentropy(), optimizer=Adam(),
metrics=['accuracy'])
#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5,
verbose=1)
model.save('model.h5', hist)
print("model created")

```

intents.json

```

{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi", "hii", "hay", "hai", "Hello", "Hey there", "Howdy", "What's
up?"],
      "responses": ["Hello! How are you today?", "Hi there! What can I do for
you?", "Hey! How's it going?", "Howdy! What's on your mind?", "What's up?
How can I assist you?"]
    },
    {
      "tag": "morning",
      "patterns": ["Good morning", "Morning", "Hey, it's morning!", "How's your
morning?", "Did you sleep well?"],
      "responses": ["Good morning! How are you feeling today?", "Morning!
What brings you here?", "Hey, it's morning! How's your day going?", "How's

```

```

your morning so far?", "Did you have a good night's sleep?"]
    },
    {
        "tag": "afternoon",
        "patterns": ["Good afternoon", "Afternoon", "Hey, it's afternoon!", "How's
your day?", "What are you up to this afternoon?"],
        "responses": ["Good afternoon! How's your day going?", "Afternoon! What
can I do for you?", "Hey, it's afternoon! How's everything?", "How's your day
so far?", "What's on your agenda this afternoon?"]
    },
    {
        "tag": "evening",
        "patterns": ["Good evening", "Evening", "Hey, it's evening!", "How's your
evening?", "Did you have a good day?"],
        "responses": ["Good evening! How has your day been?", "Evening! What
brings you here?", "Hey, it's evening! How's everything?", "How's your evening
so far?", "Did you accomplish what you wanted to today?"]
    },
    {
        "tag": "night",
        "patterns": ["Good night", "Night", "Hey, it's night!", "How was your day?",
"Are you ready to rest?"],
        "responses": ["Good night! Get some proper sleep.", "Night! Sweet
dreams.", "Hey, it's night! How was your day?", "How was your day?", "Are
you ready to unwind and relax?"]
    },
    {
        "tag": "goodbye",
        "patterns": ["Bye", "See you later", "Goodbye", "Au revoir", "Sayonara",

```

```

"ok bye", "Bye then", "Fare thee well"],
  "responses": ["See you later.", "Have a nice day.", "Bye! Come back
again.", "I'll see you soon."]
},
{
  "tag": "thanks",
  "patterns": ["Thanks", "Thank you", "That's helpful", "Thanks for the help",
"Than you very much"],
  "responses": ["Happy to help!", "Any time!", "My pleasure", "You're most
welcome!"]
},
{
  "tag": "no-response",
  "patterns": [""],
  "responses": ["Sorry, I didn't understand you.", "Please go on.", "Not sure I
understand that.", "Please don't hesitate to talk to me."]
},
{
  "tag": "neutral-response",
  "patterns": ["nothing much"],
  "responses": ["Oh I see. Do you want to talk about something?"]
},
{
  "tag": "how are you",
  "patterns": ["how are you", "How are you doing?", "How are you
feeling?", "How's everything going?", "How's life treating you?", "What's new
with you?", "What's up?"],
  "responses": ["As an chatbot, I don't have personal experiences, but I'm
operational and ready to help. What can I assist you with today?"]
}

```

```

    },
    {
      "tag": "about",
      "patterns": ["Who are you?", "What are you?", "Who you are?", "Tell me
more about yourself.", "What is your name?", "What should I call you?",
"What's your name?", "Tell me about yourself" ],
      "responses": ["I'm Mental health care chatbot, your Personal Therapeutic AI
Assistant. How are you feeling today", "I'm Mental health care chatbot, a
Therapeutic AI Assitant designed to assist you. Tell me about yourself.", "I'm
Mental health care chatbot. I am a conversational agent designed to mimic a
therapist. So how are you feeling today?"]
    },
    {
      "tag": "skill",
      "patterns": ["What can you do?"],
      "responses": ["I can provide general advice regarding anxiety and
depression, answer questions related to mental health and make daily
conversations. Do not consider me as a subsitute for an actual mental healthcare
worker. Please seek help if you don't feel satisfied with me."]
    },
    {
      "tag": "creation",
      "patterns": ["Who created you?", "Who made you?", "Who built
you", "Who's behind your creation?", "Who developed you?", "Who brought you
to life?", "Who's your creator?", "Who's responsible for making you?"],
      "responses": ["I was created by a final year CSE student at UCEA batch
(2024).", "A final year CSE student at UCEA batch (2024) created me.", "A
student from the final year CSE at UCEA batch (2024) created me."]
    },
  },

```

```

{
  "tag": "how create",
  "patterns": ["How were you made?", "How were you created?", "How did
you come to be?", "How were you made?", "What made you?", "How did you get
here?", "What happened to create you?"],
  "responses": ["I was trained on a text dataset using Deep Learning &
Natural Language Processing techniques"]
},
{
  "tag": "name",
  "patterns": ["My name is ", "I am name.", "I go by "],
  "responses": ["Oh nice to meet you. Tell me how was your week?", "Nice to
meet you. So tell me. How do you feel today?", "That's a great name. Tell me
more about yourself."]
},
{
  "tag": "sad",
  "patterns": ["I am feeling lonely", "I am so lonely", "I feel down", "I feel
sad", "I am sad", "I feel so lonely", "I feel empty", "I don't have anyone"],
  "responses": ["I'm sorry to hear that. I'm here for you. Talking about it might
help. So, tell me why do you think you're feeling this way?", "I'm here for you.
Could you tell me why you're feeling this way?", "Why do you think you feel
this way?", "How long have you been feeling this way?"]
},
{
  "tag": "stressed",
  "patterns": ["I am so stressed out", "I am so stressed", "I feel stuck", "I still
feel stressed", "I am so burned out"],
  "responses": ["What do you think is causing this?", "Take a deep breath and

```

gather your thoughts. Go take a walk if possible. Stay hydrated", "Give yourself a break. Go easy on yourself.", "I am sorry to hear that. What is the reason behind this?"]

```
    },  
    {  
      "tag": "depressed",  
      "patterns": ["I can't take it anymore", "I am so depressed", "I think i'm  
depressed.", "I have depression"],  
      "responses": ["It helps to talk about what's happening. You're going to be  
okay", "Talk to me. Tell me more. It helps if you open up yourself to someone  
else.", "Sometimes when we are depressed, it is hard to care about anything. It  
can be hard to do the simplest of things. Give yourself time to heal."]  
    },  
  ]  
}
```

Login.html

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="utf-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <title>Login</title>  
    <!-- CSS -->  
    <link href="assets/css/bootstrap.min.css" rel="stylesheet">  
    <link href="assets/css/font-awesome.min.css" rel="stylesheet">  
    <link href="assets/css/custom.css" rel="stylesheet">
```



```

<link
href="https://fonts.googleapis.com/css?family=Lato:100,100i,300,300i,400,400i
,700,700i,900,900i&subset=latin-ext" rel="stylesheet">

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

<script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body>

<div class="container-fluid">
<div class="row">
<div class="col-md-6 panel-left">
<h1></h1>
</div>
<div class="col-md-6 panel-right">
<h1>Login</h1>
<h6>Enter your email and password to Login</h6>
<div class="form-group">
<label>Email Address</label>
<input type="email" class="form-control">
</div>
<div class="form-group">
<label>Password</label>
<input type="password" class="form-control">

```

```

</div>
<div class="form-group content">
  <input type="checkbox"> Remember me
  <a href="" class="pull-right">Forgot Password?</a>
</div>
<div class="form-group">
  <button type="submit" class="btn btn-primary">Login</button>
</div>
<h5>Don't have an account? <a href="register.html">Sign Up<a></h5>
</div>
</div>
</div>
<!-- jQuery & JS -->
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/custom.js"></script>
</body>
</html>

```

Register.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Login</title>
  <!-- CSS -->
  <link href="assets/css/bootstrap.min.css" rel="stylesheet">
  <link href="assets/css/font-awesome.min.css" rel="stylesheet">

```

```

<link href="assets/css/custom.css" rel="stylesheet">
<link
href="https://fonts.googleapis.com/css?family=Lato:100,100i,300,300i,400,400i
,700,700i,900,900i&subset=latin-ext" rel="stylesheet">
<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
<script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
<script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->
</head>
<body>
<div class="container-fluid">
<div class="row">
<div class="col-md-6 panel-left">
<h1></h1>
</div>
<div class="col-md-6 panel-right">
<h1>Login</h1>
<h2>Enter your email and password to Login</h2>
<div class="form-group">
<label>Email Address</label>
<input type="email" class="form-control">
</div>
<div class="form-group">
<label>Password</label>

```

```

        <input type="password" class="form-control">
    </div>
    <div class="form-group content">
        <input type="checkbox"> Remember me
        <a href="" class="pull-right">Forgot Password?</a>
    </div>
    <div class="form-group">
        <button type="submit" class="btn btn-primary">Login</button>
    </div>
    <h5>Don't have an account? <a href="register.html">Sign Up<a></h5>
</div>
</div>
</div>
<!-- jQuery & JS -->
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/custom.js"></script>
</body>
</html>

```

style.css

```

@import
url('https://fonts.googleapis.com/css2?family=Josefin+Sans&display=swap');
:root {
--body-bg: linear-gradient(135deg, #c6d8f3 0%, #c3cfe2 100%);
--border: 20px solid #ddd;
--left-msg-bg: #f3f1f1;
--left-msg-border: 1px solid #000;
--right-msg-bg: #1a76ff;

```

```

--right-msg-border: 1px solid #1a76ff;
}
html {
box-sizing: border-box;
}
*,
*:before,
*:after {
margin: 0;
padding: 0;
box-sizing: inherit;
}
body {
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
background-image: var(--body-bg);
font-family: sans-serif;
}
.title-img{
    height:40px;
    width:40px;
    border-radius:50%;
    position:relative;
    left:1%;
    float: left;
    margin-top: 20px;
    margin-left: 20px;

```

```

}
.msger {
display: flex;
flex-flow: column wrap;
justify-content: space-between;
width: 100%;
max-width: 867px;
margin: 25px 10px;
height: calc(100% - 50px);
/*border: var(--border);*/
border-radius: 20px;
box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}
.msger-header {
/* display: flex; */
font-size: 15px;
/*justify-content: space-between;*/
border-radius: 0px 0px 0 0;
text-align: center;
background: #1a57ff;
color: #ffffff;
float: right;
}
.msger-header-title{
    position: relative;
    top:30%;
    margin-right: 60px;
}
.msger-chat {

```

```
flex: 1;
overflow-y: auto;
padding: 10px;
}
.msger-chat::-webkit-scrollbar {
width: 6px;
}
.msger-chat::-webkit-scrollbar-track {
background: #ddd;
}
.msger-chat::-webkit-scrollbar-thumb {
background: #bdbdbd;
}
.msg {
display: flex;
align-items: flex-end;
margin-bottom: 10px;
}
.msg-img {
width: 50px;
height: 50px;
margin-right: 10px;
background: #ddd;
background-repeat: no-repeat;
background-position: center;
background-size: cover;
border-radius: 50%;
}
.msg-bubble {
```

```

max-width: 450px;
padding: 15px;
border-radius: 15px;
background: var(--left-msg-bg);
border: var(--left-msg-border);
}
.msg-info {
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 10px;
}
.msg-info-name {
margin-right: 10px;
font-weight: bold;
}
.msg-info-time {
font-size: 0.85em;
}
.left-msg .msg-bubble {
border-bottom-left-radius: 0;
}
.right-msg {
flex-direction: row-reverse;
}
.right-msg .msg-bubble {
background: var(--right-msg-bg);
color: #fff;
border: var(--right-msg-border);
}

```



```

border-bottom-right-radius: 0;
}
.right-msg .msg-img {
margin: 0 0 0 10px;
}
.msger-inputarea {
display: flex;
padding: 10px 10px 10px 10px;

/*border-top: var(--border);background: #3459fe;*/
font-family: sans-serif;
background: #e1e0e0;
border-radius: 0 0 0px 0px;
}
.msg-icon{
border: 2px solid #000; /* Border style and color */
padding: 5px; /* Padding to create space between image and border */
}
.msger-inputarea * {
padding: 0px;
border: none;
border-radius: 30px;
font-family: sans-serif;
font-size: 1.15em;
border-radius: 0px;
}
.msger-input {
flex: 1;
border: none;

```

```

border-bottom: 2px solid #1a76ff;
background: #ffffff;
font-family: sans-serif;
outline: none;
}
.msger-send-btn {
margin-left: 5px;
background: rgb(255, 255, 255);
color: #ffffff;
font-weight: bold;
cursor: pointer;
font-family: sans-serif;
transition: background 0.23s;
width: 100px;
}
.msger-send-btn:hover {
background: rgb(255, 255, 255);
border: 2px solid #1a76ff;
}
.msger-chat {
background-color: #ffffff;
}

```

Mental illness types:

1. Existential crises
2. Relationship difficulties
3. Identity issues
4. Grief reactions
5. Adjustment difficulties

6. Stress-related conditions
7. Mild cognitive impairment
8. Psychosomatic symptoms
9. Subclinical anxiety
10. Subclinical depression
11. Emotional dysregulation
12. Low self-esteem
13. Spiritual crises
14. Work-related stress
15. Burnout
16. Social isolation
17. Loneliness
18. Perfectionism
19. Imposter syndrome
20. Chronic procrastination
21. Compassion fatigue
22. Decision fatigue
23. Intergenerational trauma
24. Microaggressions
25. Vicarious trauma
26. Moral injury
27. Sleep disturbances
28. Financial stress
29. Performance anxiety
30. Body image dissatisfaction
31. Academic pressure
32. Social comparison

CHAPTER 11

11. REFERENCE

1. Abd-Alrazaq, A. A., Alajlani, M., Alalwan, A. A., Bewick, B. M., & Househ, M. G. (2019). An overview of the features of chatbots in mental health: A scoping review. *International Journal of Medical Informatics*, volume 132.
2. Alonso, S., & Góngora. (Year not specified). Analyzing Mental Health Diseases in a Spanish Region Using Software Based on Graph Theory Algorithms on Innovative Computing and Communications, p. 701–709.
3. Bayu Setiaji, Ferry Wahyu Wibowo. (2016). Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling. *Intelligent Systems Modelling and Simulation (ISMS) 2016 7th International*.
4. Cameron, G., Cameson, D., Megaw, G. (2018). Towards a chatbot for digital counseling. *HCI '17: Proceedings of the 31st British Computer Society Human Computer Interaction Conference*, pp 1-7.
5. Curry, L., Nembhard, I., Bradley, E. (2009). Qualitative and mixed methods provide unique contributions to outcomes research. *Circulation*, vol. 119, no. 10, pp. 1442–1452.
6. F. Laamarti, H. F. Badawi, Y. Ding, F. Arafsha, B. Hafidh, and A. E. Saddik, “An ISO/IEEE 11073 standardized digital twin framework for health and well-being in smart cities,” *IEEE Access*, vol. 8, pp. 105950–105961, 2020.
7. Greer, S., Ramo, D., Chang, Y. J., Fu, M., Moskowitz, J., Haritatos, J. (2019). Use of the chatbot “vivibot” to deliver positive psychology skills and promote well-being among young people after cancer treatment: randomized controlled feasibility trial. *JMIR Mhealth Uhealth*, 7, e15018.
8. Haghighian Roudsari, A., Afshar, J., Lee, W., Lee, S. (2022). PatentNet: Multi-label classification of patent documents using deep learning based

- language understanding. *Scientometrics*, vol. 127, no. 1, pp. 207–231.
9. Kowatsch, T., Nißen, M., Shih, C., Rüegger, D., Volland, D. (2017). Text-based healthcare chatbots supporting patient and health professional teams: preliminary results of a randomized controlled trial on childhood obesity. *Persuasive Embodied Agents for Behavior Change (PEACH2017) Workshop*, co-located with the 17th International Conference on Intelligent Virtual Agents (IVA 2017).
 10. Kretzschmar, K., Tyroll, H., Pavarini, G. (2019). Can your phone be your therapist? Young people’s ethical perspectives on the use of fully automated conversational agents (chatbots) in mental health support. *NeurOx young people’s advisory group*.
 11. Oh, K., Lee, D., Ko, B., Choi, H. (2017). A Chatbot for Psychiatric Counseling in Mental Healthcare Service Based on Emotional Dialogue Analysis and Sentence Generation. 2017 18th IEEE International Conference on Mobile Data Management (MDM), Daejeon, pp. 371-375. doi: 10.1109/MDM.2017.64
 12. Rathod, S., Pinninti, N., Irfan, M., Gorczynski, P., Rathod, P., Gega, L., et al. (2017). Mental Health Service Provision in Low- and Middle-Income Countries. *Health Serv Insights*, 10, 1–7.
 13. Statista. (2020). Number of mHealth Apps Available in the Apple Store from 1st Quarter 2015 to 2nd Quarter 2020. Retrieved from <https://www.statista.com/statistics/779910/health-apps-available-ios-worldwide/>.
 14. Statista. (2020). Number of mHealth Apps Available in the Google Play Store from 1st Quarter 2015 to 2nd Quarter 2020. Retrieved from <https://www.statista.com/statistics/779919/health-apps-available-google-play-worldwide/>
 15. S. Burchert, A. Kerber, J. Zimmermann, and C. Knaevelsrud, “Screening accuracy of a 14-day smartphone ambulatory assessment of depression

- symptoms and mood dynamics in a general population sample: Comparison with the PHQ-9 depression screening,” PLoS ONE, vol. 16, no. 1, Jan. 2021, Art. no. e0244955
16. Tricco, A. C., Lillie, E., Zarin, W., O’Brien, K. K., Colquhoun, H., Levac, D., et al. (2018). PRISMA Extension for Scoping Reviews (PRISMA-ScR): Checklist and Explanation. *Ann Intern Med*, 169(7), 467–473.
17. Wang, Y., Zhang, N., Zhao, X. (2022). Understanding the determinants in the different government AI adoption stages: Evidence of local government chatbots in China. *Social Science Computer Review*, vol. 40, no. 2, pp. 534–554.