ESOFT METRO CAMPUS

LONDON METROPOLITAN UNIVERSITY

CU6051ES - Artificial Intelligence

COURSEWORK 2

FocusFriend

ADHD Support BOT

TEAM MEMBERS

Name	Student ID	Esoft Registration No	E-mail
W V M M S Fernando	E027488	00224191	maxtinafernando@gmail.com
Kalaikumar Janarthana	E217463	00209353	kalaikumarjanarthana@gmail.com

GITHUB LINK

https://github.com/mf9001/focusfriend

Table of Contents

Table of Figures 5

List Of Tables 5

1. Project Overview 6

1.1 Introduction 6

1.2 Problem Description 6

1.3 Aim 6

1.4 Objectives 7

1.5 Target Audience 7

1.6 Chatbot Features 7

2. Research and Challenges 8

2.1 Literature review 8

2.2 User needs analysis 8

2.3 Challenges Faced in Building FocusFriend 9

3. Proposed System 9

4. UML Diagrams 10

4.1 Activity Diagram 10

4.2 Sequence Diagram 11

5. Pre-requisites and Requirements 12

5.1 Software Requirements 12

5.2 Hardware Requirements 14

6. Design Architecture 15

6.1 High-level Architecture of FocusFriend 15

6.2 Process of responding to a user query: 15

7. Functionalities of the main components 16

7.1 User Interface (UI) 16

7.2 Natural Language Processing (NLP) Engine 17

7.3 Database 18

7.4 BOT component 20

7.5 Machine Learning Module 20

8. P.E.A.S 22

9. Key design ideas 23
10. Algorithms 25
10.1 Flowchart 25
10.2 Pseudocode 26
11.Test Cases 28
12.Conclusion 33
11.1Potential Improvements 33
13.Source code 34
13.1 BOT Component 34
13.2 Training component 36
13.3 Intents 38
13.4 User Interface 38
References 42

Table of Figures

- **Figure 1: Activity Diagram** 10
- **Figure 2: Sequence Diagram of User** 11
- **Figure 3:Sequence Diagram of Developer** 11
- **Figure 4:High Level Architecture** 15
- **Figure 5: User Interface** 16
- **Figure 6: NLP Pipeline** 17
- **Figure 7: Intents and Greeting Query** 19
- **Figure 8: Intents: ADHD Related Query** 19
- **Figure 9: Neural Network** 21
- **Figure 10: Small Talk** 23
- **Figure 11: Random Response** 24
- **Figure 12: Rule Based response** 24
- **Figure 13: Rule Based response-Intent** 25
- **Figure 14: Flowchart** 25

List Of Tables

- **Table 1:Backend Technologies** 12
- **Table 2:Frontend Technologies** 13
- **Table 3:API Development Tools** 13
- **Table 4:Development Tools** 13
- **Table 5:Hardware requirements** 14
- **Table 6:PEAS** 22
- **Table 7:Test Case 01** 28
- **Table 8:Test Case 02** 29
- **Table 9:Test Case 03** 30
- **Table 10: Test Case 04** 31

1. Project Overview

1.1 Introduction

Artificial Intelligence (AI) has significantly influenced various industries, notably healthcare, over the past few decades. The rise of virtual assistants and AI-powered chatbots in hospitals, labs, pharmacies, and nursing homes underscores this transformation [2]. A chatbot, or a computer program simulating human conversation, has emerged as an important tool in customer service and healthcare management.

FocusFriend is a chatbot designed to assist individuals with ADHD (Attention Deficit Hyperactivity Disorder) and their families. ADHD is a neurodevelopmental disorder, affecting both children and adults. People with ADHD often have trouble paying attention and/or controlling their energy and impulses which can affect their daily lives and development in children [8]. Accurate information, timely support, and practical coping strategies are crucial for managing ADHD. FocusFriend aims to provide a reliable, accessible, and user-friendly platform to offer professional and specific answers to ADHD-related queries, thus enhancing the quality of life for those affected.

1.2 Problem Description

ADHD affects many individuals globally, yet many struggle to access reliable information, timely support, and effective coping strategies. Existing resources often fail to provide immediate, continuous assistance, essential for managing ADHD's diverse and fluctuating symptoms. Individuals with ADHD, including students and adults, face challenges like inattention, impulsivity, and hyperactivity, impacting their academic, professional, and personal lives. Misconceptions about ADHD due to inaccurate information further intensify stress and confusion. Overwhelmed healthcare professionals and support systems often result in delayed responses or insufficient support [1]. FocusFriend aims to bridge this gap by providing fundamental assistance to those in need.

1.3 Aim

The goal of FocusFriend is to provide an AI-driven chatbot that uses machine learning to offer support by providing accurate information to individuals with ADHD, helping to improve their quality of life.

1.4 Objectives

- Develop a user-friendly chatbot for individuals with ADHD.
- Ensure accurate and timely information delivery.
- Provide 24/7 accessible support.
- Match user queries and respond appropriately.

1.5 Target Audience

The target audience of this project are individuals suffering with ADHD, their parents, educators and caretakers.

1.6 Chatbot Features

FocusFriend is a sophisticated chatbot specifically designed to support individuals with ADHD. Below are some of the specific chatbot traits of FocusFriend.

1.6.1 Natural Language Understanding (NLU)

NLU uses AI to understand user inputs in sentences or speech. FocusFriend leverages advanced NLU to ensure meaningful and contextually appropriate conversations.

1.6.2 Intent Recognition

The NLU engine identifies the user's intent, whether seeking information or coping strategies, and responds accordingly.

1.6.3 Context Management

FocusFriend maintains coherent and relevant conversations by tracking historical context and handling interruptions effectively.

1.6.4 Logical Responses

Using advanced NLP techniques, FocusFriend ensures responses are logical, relevant, and based on evidence-based information and best practices for managing ADHD.

2. Research and Challenges

2.1 Literature review

Attention Deficit Hyperactivity Disorder (ADHD) is a neurodevelopmental disorder that significantly impact the lives of affected individuals. Traditional treatment approaches primarily involve pharmacological interventions and behavioral therapies. However, recent advancements in artificial intelligence (AI) and noninvasive technologies present new opportunities for revolutionizing the treatment of this disorder. This literature review explores the AI driven non-invasive approach for treating ADHD.

Mattingly, Wilson, and Rostain provide a comprehensive guide to ADHD treatment options for clinicians, covering both pharmacological and non-pharmacological strategies [7]. They discuss the importance of various medications, including stimulants and non-stimulants, and highlight the importance of individualized treatment plans. The authors also emphasize the role of behavioral interventions, psychological education, and lifestyle modifications in managing ADHD symptoms.

Christensen, Griffiths, and Korten focus on improving adherence to mental health interventions using web-based and mobile platforms in their paper [6]. Improving adherence in the use of web-based and mobile interventions for mental health and substance use disorders". They emphasize the role of digital interventions in supporting individuals with ADHD and other mental health disorders. The authors discuss various strategies to enhance user engagement and adherence, such as personalized content, interactive features, and real-time feedback.

Furthermore, Dalili and Zohuri discuss the potential of AI-driven noninvasive approaches to transform the treatment landscape for ADHD in their paper Revolutionizing Treatment: AI-Driven Noninvasive Approaches for ODD and ADHD [3]. The authors provide an overview of various AI methods, including machine learning and deep learning, that are being used to develop diagnostic tools for ADHD. These tools analyze behavioral and physiological data to identify ADHD symptoms with high accuracy. The review highlights the potential of these technologies to provide early and accurate diagnoses, which are crucial for effective intervention.

2.2 User needs analysis

Research to understand the needs of individuals with ADHD has revealed a multifaceted set of challenges and requirements. One study highlighted that ADHD affects not only cognitive and behavioral aspects but also social and emotional well-being, necessitating comprehensive support strategies tailored to these dimensions [7]. Moreover, resources such as The ADHD Centre provide valuable information and support, emphasizing the need for ongoing education, therapeutic interventions, and community support to effectively manage ADHD symptoms and improve quality of life.

2.3 Challenges Faced in Building FocusFriend

- **Quality of the data**: Obtaining sufficient and high-quality data (related to a sensitive topic like ADHD, covering a range of areas including symptoms, diagnosis, treatment etc.); to train the model was challenging.

- **Context Understanding**: ADHD support often involves understanding context beyond the input, and requires understanding emotional cues, previous interactions etc. Hence, building a model that can interpret and respond appropriately to match the context was challenging. In our model we have implemented Natural Language Understanding (NLU) technique 'context tracking' as one mechanism to tackle this challenge.

- **Model Training and Tuning**: Training a model by adjusting hyperparameters (eg: epochs, no of layers) to provide meaningful responses was challenging and required a higher time allocation.

- **Hardware Resources**: Since training and deploying ML models are computationally intensive, a relatively powerful CPU was required.

- **Latency**: Timely responses are important specially for a support chatbot. However, due to the computational requirement, when predicting the response, there was a delay. The algorithm needs to be improved further to resolve this issue.

3. Proposed System

FocusFriend is an innovative chatbot platform designed to support individuals managing ADHD through natural language processing (NLP) techniques. It provides a platform to respond to user queries related to ADHD. By combining machine learning algorithms with a database of ADHD related information, FocusFriend delivers accurate and reliable information through a user-friendly interface, to help users navigate ADHD challenges effectively.

4. UML Diagrams

4.1 Activity Diagram

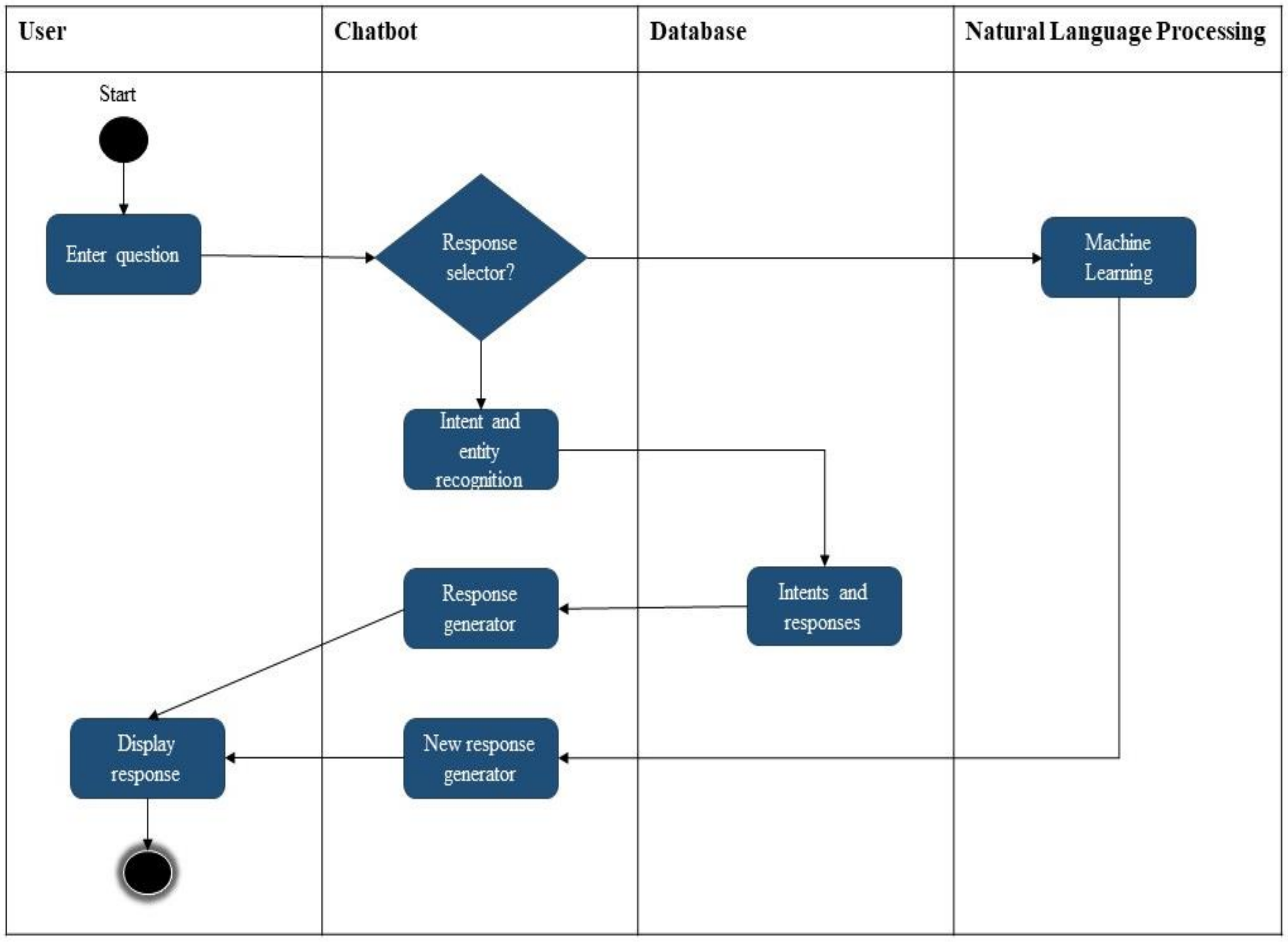


Figure 1: Activity Diagram

4.2 Sequence Diagram

4.3.1 Sequence Diagram of User

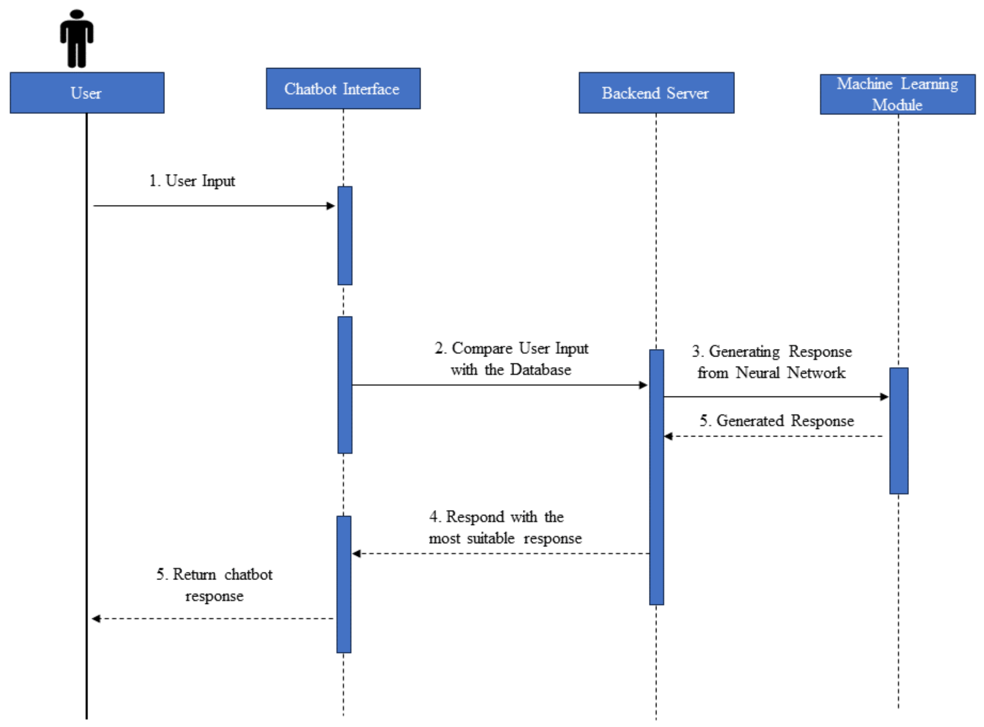

User
Chatbot Interface
Backend Server
Machine Learning Module
1. User Input
2. Compare User Input with the Database
3. Generating Response from Neural Network
5. Generated Response
4. Respond with the most suitable response
5. Return chatbot response

Figure 2: Sequence Diagram of User

4.3.1 Sequence Diagram of Developer

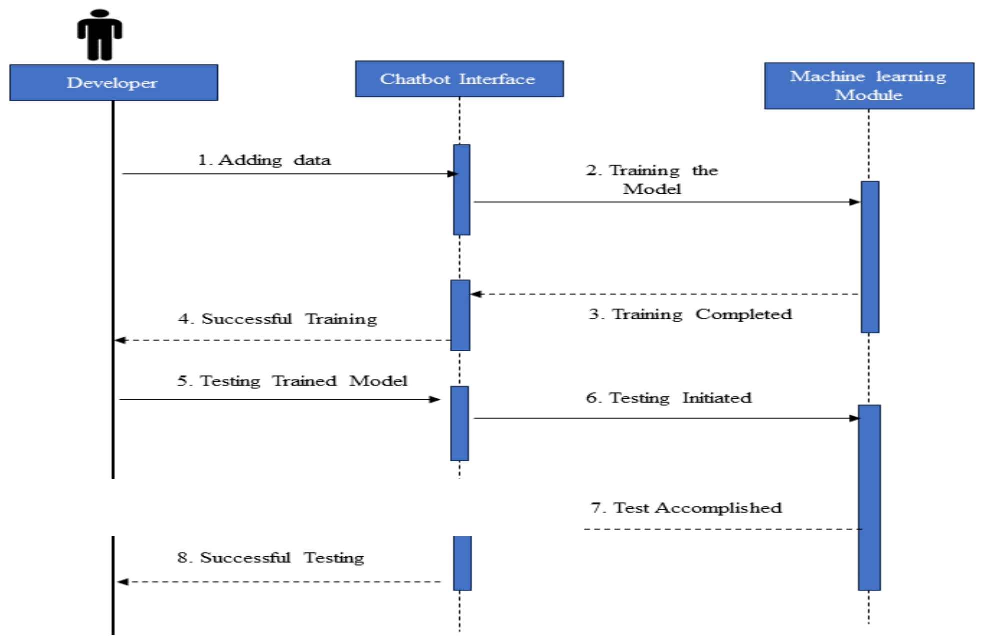


Figure 3:Sequence Diagram of Developer

5. Pre-requisites and Requirements

5.1 Software Requirements

Backend Technologies		
Technology	**Tool**	**Usage in the Project**
Primary Programming Language	Python	•Provides the core logic for handling user inputs. •Processes input with the trained machine learning model, and generates responses.
Machine Learning Platforms	Tensorflow, Keras	•Used to: -Build (define the architecture of the ANN(Artificial Neural Network)) -Train (using the dataset to learn patterns and generate meaningful responses); and -Deploy (save the trained model to a file and load it during runtime, to make predictions on user inputs); the machine learning model . •Keras (a high level API) is used to simplify the process of building and training the ANN.
	NLTK (Natural Language Toolkit)	•Used for data preprocessing (stop word removal and lemmatization) in the ML component and in the Bot component for preprocessing user query.
Python Libraries	Sklearn (Scikit-learn)- Machine Learning Library	•Used in Label Encoding (converts textual intent labels into numerical values) to be used in the machine learning algorithm to process and learn from the data.
	Pandas	• Used for data handling including training datasets. • Cleaning and preparing data for the machine learning model.
Data Handling	JSON (JavaScript Object Notation)	•JSON is used for data exchange between the frontend and the backend •Used to: -Send Data: The frontend sends user inputs to the backend in JSON format. -Receive Data: The backend responds with chatbot messages in JSON format. -Data File: Stores chatbot's responses (pre-defined intents, patterns and responses).

Table 1:Backend Technologies

Frontend Technologies		
Technology	**Tool**	**Usage in the Project**
Programming Languages	JavaScript	•Adds interactivity to the User Interface (UI). •Handles user inputs, makes asynchronous requests to the backend API using 'fetch' API and updates the web page with the chatbot's responses.
	HTML (HyperText Markup Language)	•Adds structure the content to the web page by defining layout and elements of the chatbot interface (Eg: input fields, buttons, and message display areas).
	CSS (Cascading Style Sheets)	•Used to style the HTML elements by adding layout, colors, fonts, and overall design of the web page to enhance the user experience.

Table 2:Frontend Technologies

REST API Development Tools		
Technology	**Tool**	**Usage in the Project**
Framework	Flask	•Provides communication between the frontend and the backend. •The REST API endpoints (created using Flask) allow the frontend to send user inputs to the backend and receive chatbot responses. These endpoints are used to handle the HTTP method GET.
API Testing Tools	Postman	•Ensures that the backend services are functioning accurately, and responses are as expected

Table 3:API Development Tools

Development Tools		
Technology	**Tool**	**Usage in the Project**
Integrated Development Environment (IDE)	VSCode	•For versatile development with extensions for Python, JavaScript, HTML, and CSS
Version Control	GitHub	•For repository hosting and collaboration

Table 4:Development Tools

5.2 Hardware Requirements

Hardware requirements		
Development Machine	Processor	Intel Core i5 or higher
	RAM	16GB or higher DDR4 RAM
	Storage	500GB SSD or higher

Table 5:Hardware requirements

6. Design Architecture

6.1 High-level Architecture of FocusFriend

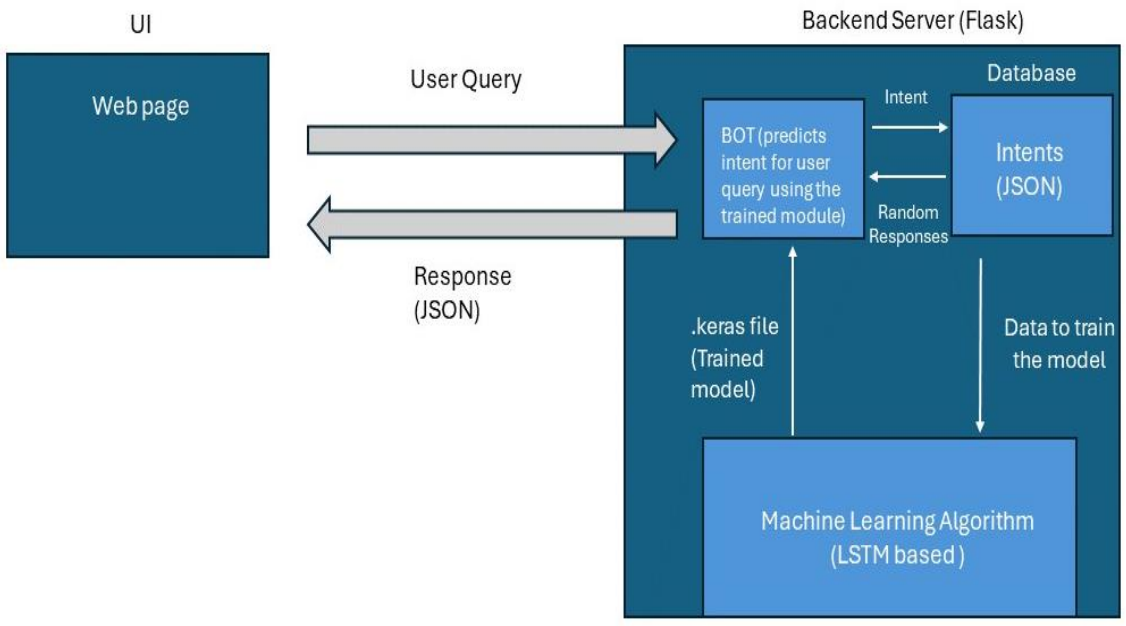

UI
Web page
User Query
Response (JSON)
Backend Server (Flask)
Database
BOT (predicts intent for user query using the trained module)
Intent
Random Responses
Intents (JSON)
.keras file (Trained model)
Data to train the model
Machine Learning Algorithm (LSTM based)

Figure 4:High Level Architecture

6.2 Process of responding to a user query:

- **User Query Input:** The user interacts with the chatbot via the UI
- **User Query Processing:** BOT receives the user input and preprocesses it (using NLP) to normalize the text.
- **Intent Prediction:** BOT predicting the intent behind the query using the trained model (provided by the machine learning component)
- **Context Management:** The chatbot maintains the context of the conversation.
- **Response Generation:** Based on the predicted intent and identified entities, BOT generates an appropriate response from predefined responses stored in a JSON database.
- **Response Delivery:** The generated response is to the user through the UI.
- **Learning and Improvement**: User interactions are logged and analyzed to continually improve the quality of the responses through retraining the models.

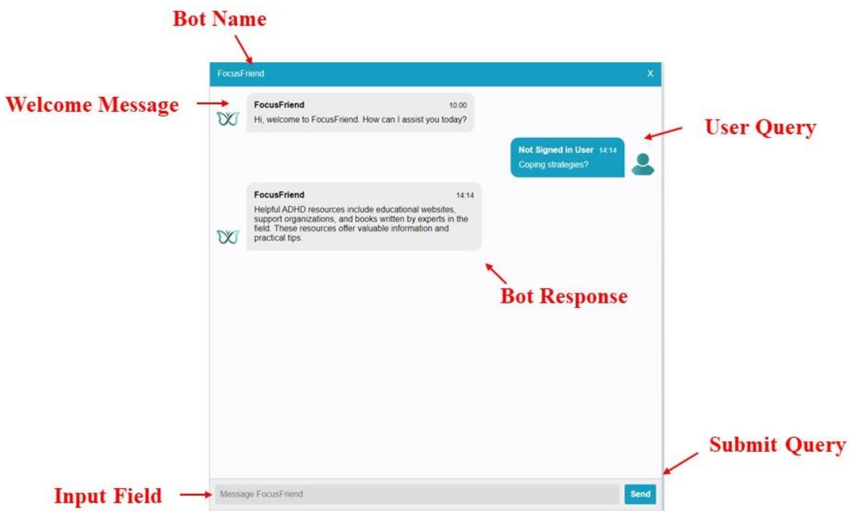
7. Functionalities of the main components

Functionalities of the main components of the chatbot are described below;

7.1 User Interface (UI)

- The locally hosted web page serves as the user interface (front-end) where users can type in their queries and get responses from the chatbot.
- When a user inputs a query into the input field and lick the ‘Send’ button, it is sent to the backend server, and the response from the server is displayed in the UI.
- AJAX (Asynchronous JavaScript and XML) was used to enable web pages to update dynamically, providing a more responsive and interactive user experience.

Technologies Used: HTML, CSS, JavaScript, AJAX



FocusFriend 10.00
Hi, welcome to FocusFriend. How can I assist you today?

Not Signed in User 14:14
Coping strategies?

FocusFriend 14:14
Helpful ADHD resources include educational websites, support organizations, and books written by experts in the field. These resources offer valuable information and practical tips.

Figure 5: User Interface

7.2 Natural Language Processing (NLP) Engine

NLP engine was used to understand and generate human-like text messages in the chatbot.

Processes of the NLP Engine:

7.2.1 Text Preprocessing:

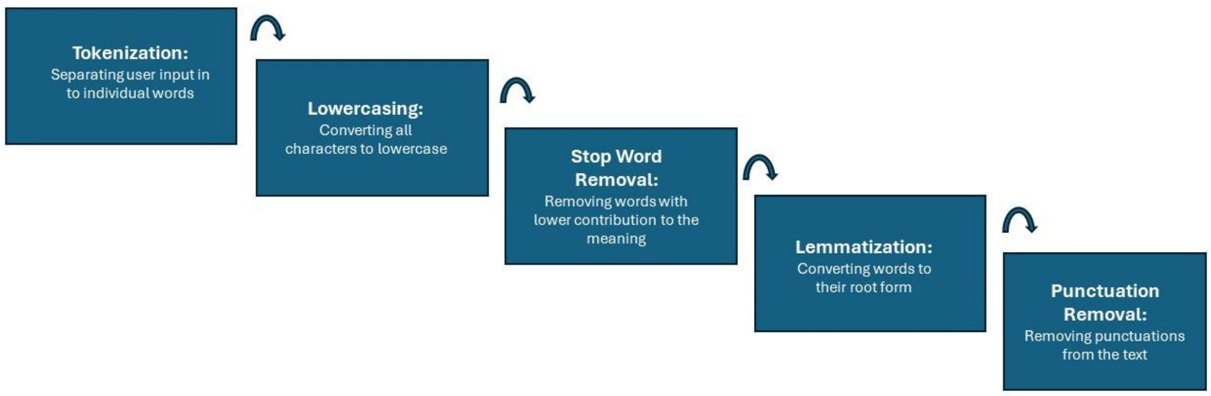

Tokenization: Separating user input in to individual words
Lowercasing: Converting all characters to lowercase
Stop Word Removal: Removing words with lower contribution to the meaning
Lemmatization: Converting words to their root form
Punctuation Removal: Removing punctuations from the text

Figure 6: NLP Pipeline

- **Tokenization**

Tokenization is the process of breaking down a text into smaller units (tokens). Generally, tokenization refers to separating a sentence into individual words. The aim is to transform the text into manageable form for further processing and analysis.

- **Lowercasing**

Lowercasing is converting all characters in the text to lowercase to ensure uniformity for the ease of processing.

- **Stop Word Removal**

Common words which contribute less to the meaning of the text are removed.

- **Lemmatization**

Reduces words to their dictionary form, considering the context.

- **Punctuation Removal**

Punctuation marks such as commas, periods, exclamation marks, etc. removed from the text, since they do not add significant value to the text. Eliminating them will reduce noise and improve efficiency in processing.

Example of Text Preprocessing:

"We are building a healthcare chatbot!"

1. Tokenization: ["We", "are", "building", "a", "healthcare", "chatbot", "!"]
2. Lowercasing: ["we", "are", "building", "a", "healthcare", "chatbot", "!"]
3. Stop Word Removal: ["building", "healthcare", "chatbot", "!"]
4. Lemmatization: ["building", "healthcare", "chatbot", "!"]
5. Punctuation Removal: ["building", "healthcare", "chatbot"]

7.2.2 Other NLP Processes used in FocusFriend:

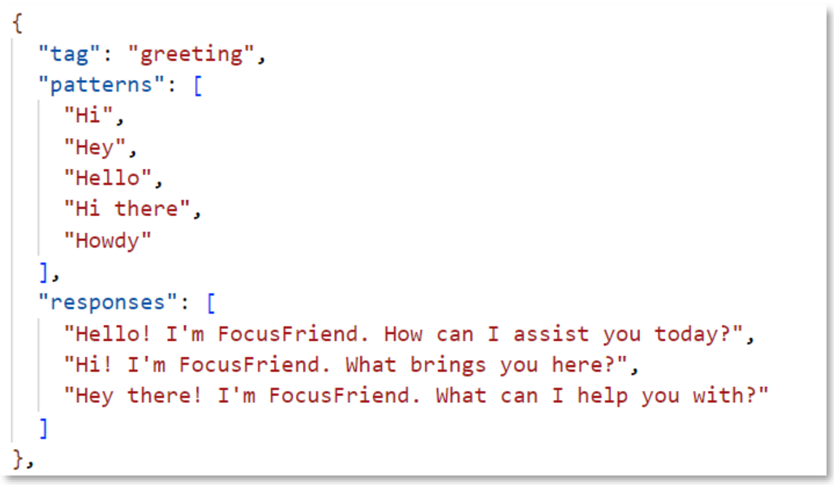
- **Feature Extraction:** Converting user inputs into machine understandable language.
- **Intent Recognition:** Recognizing intents from user inputs.
- **Reply generation:** Forming replies based on intent and entity identification.
- **Dialog management:** Managing coherence in a dialog.

Key Technologies Used: Python NLTK Library, TensorFlow/ Keras

7.3 Database

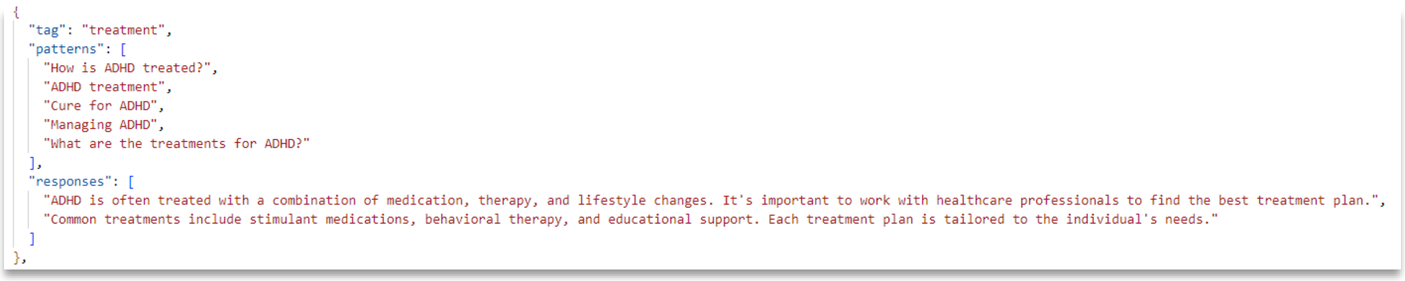
- The chatbot's database is stored in a JSON file and the BOT component interacts with the database using the python module ‘json’.
- The database contains a collection of intents. Each intent represents a category of user input that the chatbot can recognize and respond to.
- Key components of an intent (which is an object in the JSON array):
    - **tag**: A unique identifier for the intent
    - **patterns**: A list of example user inputs
    - **responses**: A list of potential replies that the chatbot can use when this intent is identified.

- JSON Database used to:
  - Comparing ‘patterns’ to find the best match for the ‘tag’, for user inputs (intent recognition).
  - Select a response from the ‘responses’ array associated with that intent (response generation).
  - Maintain coherence of the conversations through “context_set”(chatbot keeps track of a particular context for future use) (context management).



```
{
  "tag": "greeting",
  "patterns": [
    "Hi",
    "Hey",
    "Hello",
    "Hi there",
    "Howdy"
  ],
  "responses": [
    "Hello! I'm FocusFriend. How can I assist you today?",
    "Hi! I'm FocusFriend. What brings you here?",
    "Hey there! I'm FocusFriend. What can I help you with?"
  ]
},
```

Figure 7: Intents and Greeting Query



```
{
  "tag": "treatment",
  "patterns": [
    "How is ADHD treated?",
    "ADHD treatment",
    "Cure for ADHD",
    "Managing ADHD",
    "What are the treatments for ADHD?"
  ],
  "responses": [
    "ADHD is often treated with a combination of medication, therapy, and lifestyle changes. It's important to work with healthcare professionals to find the best treatment plan.",
    "Common treatments include stimulant medications, behavioral therapy, and educational support. Each treatment plan is tailored to the individual's needs."
  ]
},
```

Figure 8: Intents: ADHD Related Query

7.4 BOT component

- Receives user input from the UI and processes it to identify the user's intent.
- Using the trained machine learning model, predict the intent for the user query.
- Selects an appropriate response based on the predicted intent and predefined responses in the JSON database.
- Maintains the context of the conversation to ensure coherent and relevant interactions.

Key Technologies Used: Python, NLTK

7.5 Machine Learning Module

- Prepares and processes the training data from the JSON file.
- Utilizes the LSTM (Long Short-Term Memory) model to train the Neural Network on the processed data.
- Saves the trained model in a .keras file format for later use by the BOT component.
- Loads the trained model during runtime to make predictions on user queries.

  Technologies Used: TensorFlow/Keras, Scikit-learn, NLTK

7.5.1 Training the Machine Learning Model

(teaching the chatbot to make accurate predictions on input data)

- **Data Loading**: Loads collection of various user queries and their associated intents or responses from a JSON file .

- **Data Preprocessing**: Prepare the data for training using NLP processes.

- **Label Encoding**: Encoding text-based labels (tags) into numerical format (since the numerical data should be fed to the model)

- **Model Construction**: Construct artificial neural network (ANN) model Long Short-Term Memory (LSTM) for handling sequential data.

- **Training the Model:**
  - **Feeding the model with data**: Input the training data into the model in batches and make initial predictions.

  - **Error Calculation**: Calculates the error by comparing the model's predictions with the actual labels.

- **Iterative Learning**: Repeat the processes of feeding data, calculating errors, and updating parameters over many epochs (iterations) in order to make improved predictions.
- **Evaluation**: Evaluate model's performance using validation data to ensure chatbot responses well to unseen data.
- **Fine-Tuning**: Model can be fine-tuned by adjusting hyperparameters (Eg: no of epochs) or training the model further on additional data.

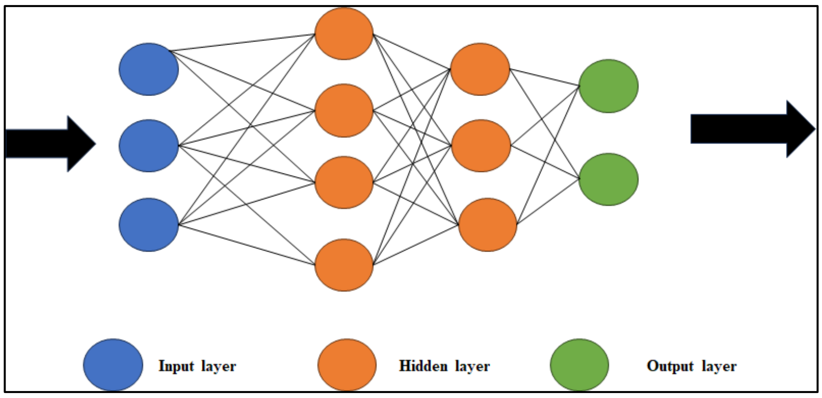


Figure 9: Neural Network

8. P.E.A.S

PEAS is a framework that helps define and analyse the key aspects of an AI system. It stands for Performance, Environment, Actuators, and Sensors. PEAS in the context of FocusFriend is as follows;

Component	Relevancy to FocusFriend
Performance Measure (Defines how the success of the agent is measured)	• Accuracy of responses to user inputs. • Response time to user queries. • The ability of the chatbot to handle unexpected inputs or errors smoothly. • User Satisfaction (measured through engagement rates, and retention).
Environment (External context in which the agent operates and interact)	• Platform the chatbot operates in (website, mobile app etc.) • Access to external resources (Eg: online resources, databases, APIs) • Adapt responses based on user history, preferences, and current context
Actuators (Mechanisms through which the agent affects the environment)	• Text output to user queries in terms of ADHD diagnosis, treatments, coping etc.
Sensors (Mechanisms used by agent to perceive and gather information about the environment)	• Continuously process and analyze user inputs to understand needs and provide relevant support. • Track user interactions to identify patterns and provide better responses.

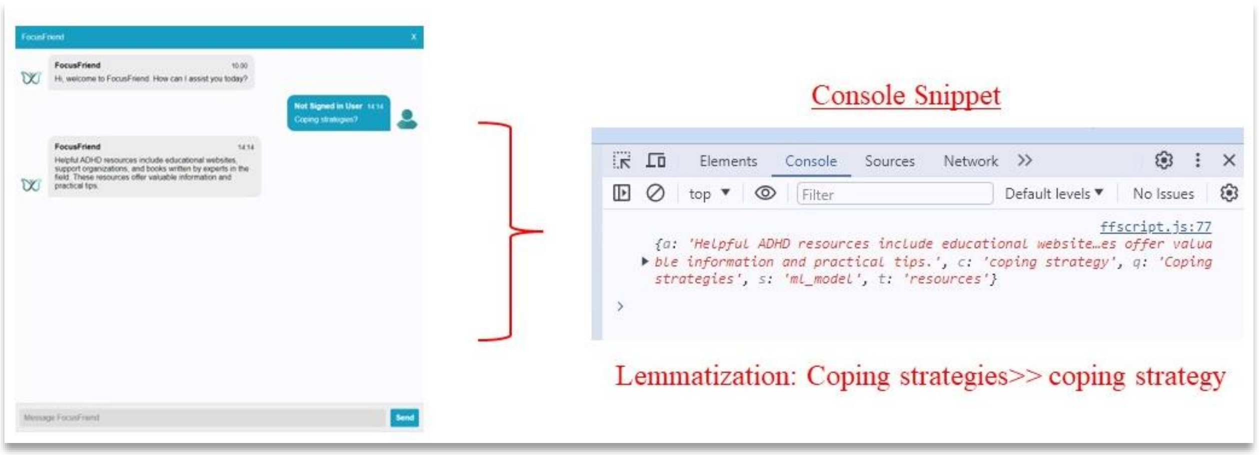
Table 6:PEAS

9. Key design ideas

Methodologies and technologies used to create an effective and responsive bot to provide support related to ADHD are as follows;

9.1 Lemmatization of User Inputs

Process of transforming words into their root form (Eg: Walking Walks, Walked => “walk”), to normalize text in order to understand the intent behind user inputs.



FocusFriend

FocusFriend 10:00
Hi, welcome to FocusFriend. How can I assist you today?

Not Signed in User 14:14
Coping strategies?

FocusFriend 14:14
Helpful ADHD resources include educational websites, support organizations, and books written by experts in the field. These resources offer valuable information and practical tips.

Message FocusFriend

Console Snippet

```
                                                         ffscript.js:77
  {a: 'Helpful ADHD resources include educational website…es offer valua
▸ ble information and practical tips.', c: 'coping strategy', q: 'Coping
  strategies', s: 'ml_model', t: 'resources'}
>
```

Lemmatization: Coping strategies>> coping strategy

Figure 10: Small Talk

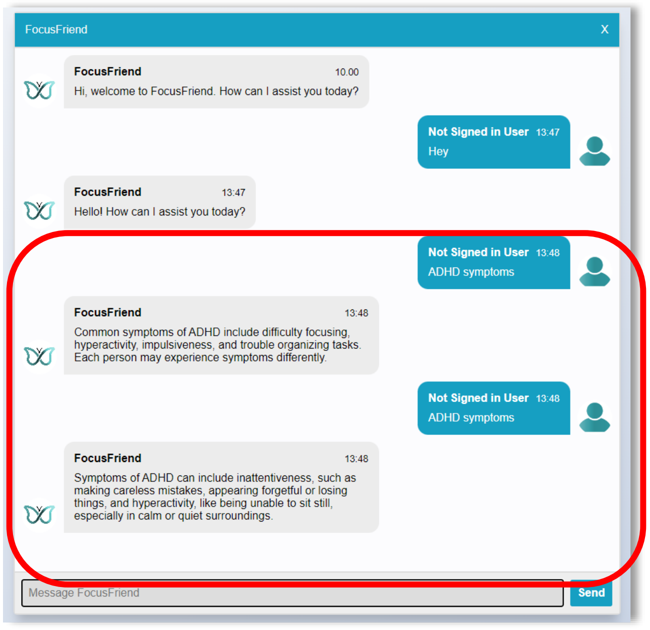
9.2 Small Talk

Small talk involves engaging users in casual conversation to improve the user engagement. (Eg: Greetings in FocusFriend)

9.3 Random Answers

Maintains the chatbot's persona and avoids repetitive responses by providing different answers to similar questions.

Example of random responses:


FocusFriend X

FocusFriend 10.00
Hi, welcome to FocusFriend. How can I assist you today?

Not Signed in User 13:47
Hey

FocusFriend 13:47
Hello! How can I assist you today?

Not Signed in User 13:48
ADHD symptoms

FocusFriend 13:48
Common symptoms of ADHD include difficulty focusing, hyperactivity, impulsiveness, and trouble organizing tasks. Each person may experience symptoms differently.

Not Signed in User 13:48
ADHD symptoms

FocusFriend 13:48
Symptoms of ADHD can include inattentiveness, such as making careless mistakes, appearing forgetful or losing things, and hyperactivity, like being unable to sit still, especially in calm or quiet surroundings.

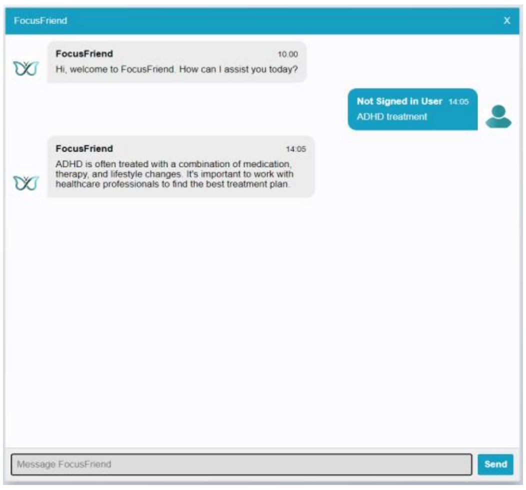
Message FocusFriend
Send

Figure 11: Random Response

9.4 Database Answers (Rule-based)

The chatbot can fetch and use information from an intents list stored in a JSON file to respond to user queries. This ensures accurate and relevant responses based on stored data.

Example of providing answers from the database:


FocusFriend x

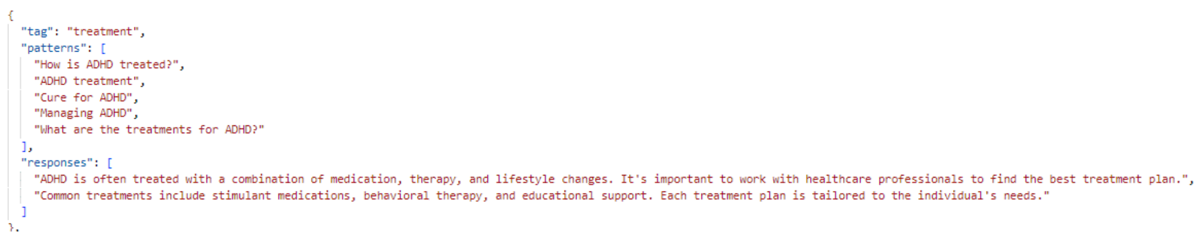
FocusFriend 10.00
Hi, welcome to FocusFriend. How can I assist you today?

Not Signed in User 14:05
ADHD treatment

FocusFriend 14:05
ADHD is often treated with a combination of medication, therapy, and lifestyle changes. It's important to work with healthcare professionals to find the best treatment plan.

Message FocusFriend
Send

Figure 12: Rule Based response



```
{
  "tag": "treatment",
  "patterns": [
    "How is ADHD treated?",
    "ADHD treatment",
    "Cure for ADHD",
    "Managing ADHD",
    "What are the treatments for ADHD?"
  ],
  "responses": [
    "ADHD is often treated with a combination of medication, therapy, and lifestyle changes. It's important to work with healthcare professionals to find the best treatment plan.",
    "Common treatments include stimulant medications, behavioral therapy, and educational support. Each treatment plan is tailored to the individual's needs."
  ]
},
```

Figure 13: Rule Based response: Intent

9.5 Suitable User Interface

Ensures that the chatbot interface is user-friendly with a clear visual hierarchy, making interactions intuitive and effective.

10. Algorithms

10.1 Flowchart

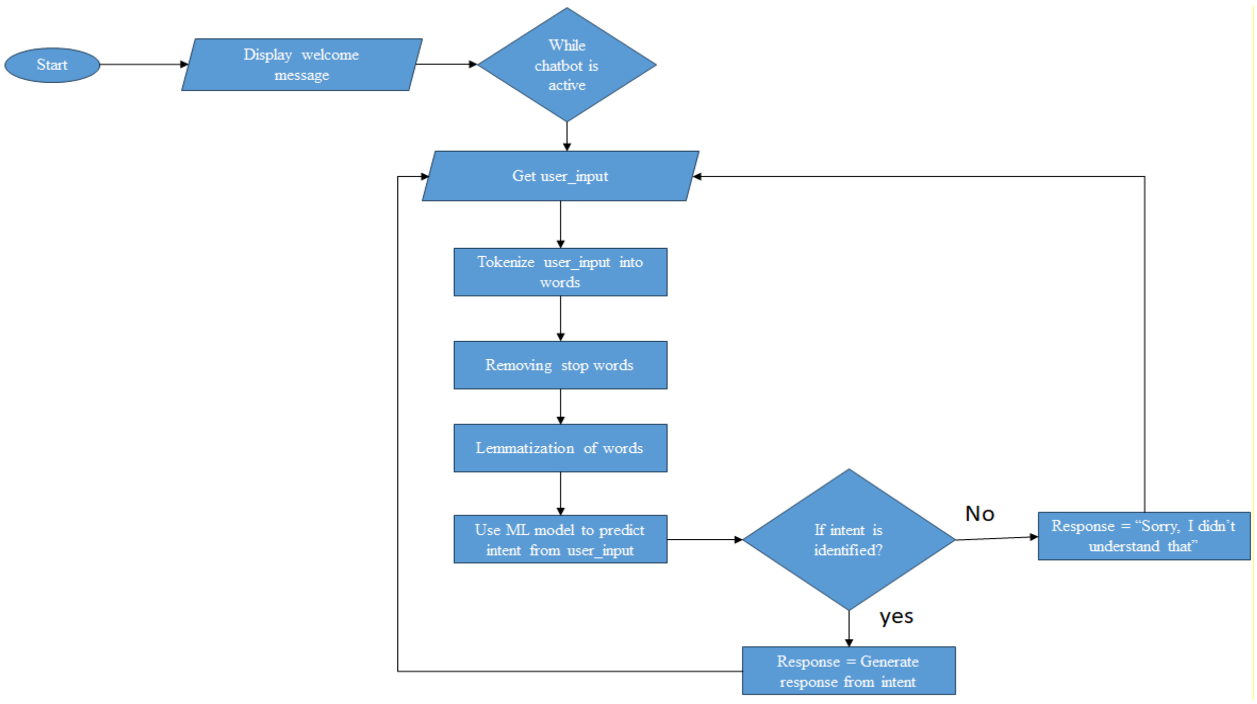

Start
Display welcome message
While chatbot is active
Get user_input
Tokenize user_input into words
Removing stop words
Lemmatization of words
Use ML model to predict intent from user_input
If intent is identified?
No
Response = "Sorry, I didn't understand that"
yes
Response = Generate response from intent

Figure 14: Flowchart

10.2 Pseudocode

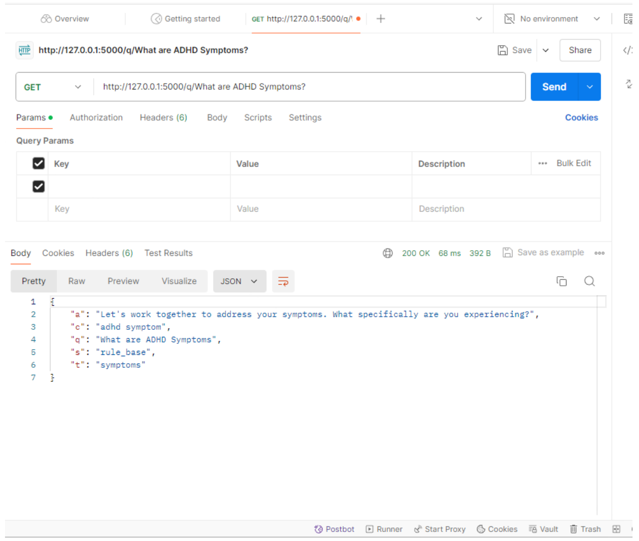
Intent Recognition

```
Initialize Libraries and Models:
    Load Natural Language Processing (NLP) library
    Load trained machine learning model for intent classification
Define a list of intents and associated keyword
Tokenize input_text into words
Remove stop words
Apply stemming or lemmatization to reduce words to their root form
Return preprocessed text
Convert preprocessed_text to a format suitable for the ML model
Use the ML model to predict intent
Return predicted intent
```

Chatbot Functionality

```
Initialize chatbot system
While chatbot is active:
    user_input = GetUserInput()
    intent = RecognizeIntent(user_input)
    If intent is identified:
        response = GenerateResponse(intent)
    Else:
        response = "Sorry, I didn't understand that."
Display response to user
```

11.Test Cases

Test Case No: 1		Test Tile: NLP Testing			
Description: Lowercasing, stop word removal, lemmatization, punctuation removal					
Step	Action	Expected System Result	Actual Result	Pass/ Fail	Comment
Checking if the output has followed the NLP steps	Enter query "What are ADHD Sytmptoms?" And check the output in Postman	"adhd symptom"	 GET http://127.0.0.1:5000/q/What are ADHD Symptoms? — 200 OK 68 ms 392 B — `{ "a": "Let's work together to address your symptoms. What specifically are you experiencing?", "c": "adhd symptom", "q": "What are ADHD Symptoms", "s": "rule_base", "t": "symptoms" }`	Pass	NLP steps implemented successfully in the code

```
{
    "a": "Let's work together to address your symptoms. What specifically are you experiencing?",
    "c": "adhd symptom",
    "q": "What are ADHD Symptoms",
    "s": "rule_base",
    "t": "symptoms"
}
```

Table 7:Test Case 01

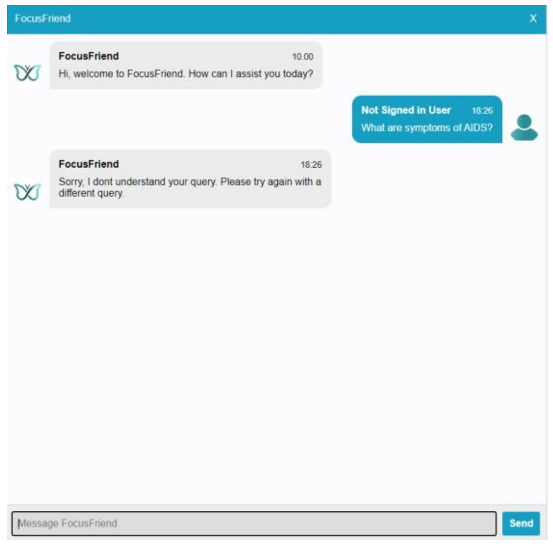
Test Case No: 2		**Test Tile: Invalid Query Testing**			
Description: Identification of invalid queries					
Step	Action	Expected System Result	Actual Result	Pass/ Fail	Comment
Input an invalid query (a word which is not in the vocablary used to train the model)	Enter query "What are symptoms of AIDS"?	Response of the bot should be "Sorry, I dont understand your query. Please try again with a different query."	 FocusFriend X FocusFriend 10.00 Hi, welcome to FocusFriend. How can I assist you today? Not Signed in User 18:26 What are symptoms of AIDS? FocusFriend 18:26 Sorry, I dont understand your query. Please try again with a different query. Message FocusFriend Send	Pass	Query was successfully identified as invalid and response was provided accordingly.

Table 8:Test Case 02

Test Case No: 3 **Test Tile: Invalid Query Testing**

Description: Rule-based output testing

Step	Action	Expected System Result	Actual Result	Pass/ Fail	Comment

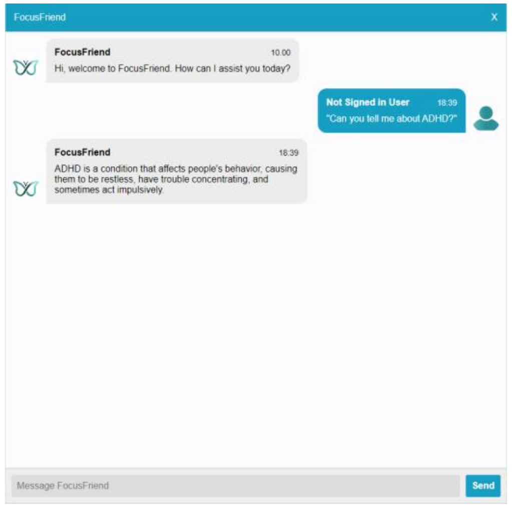
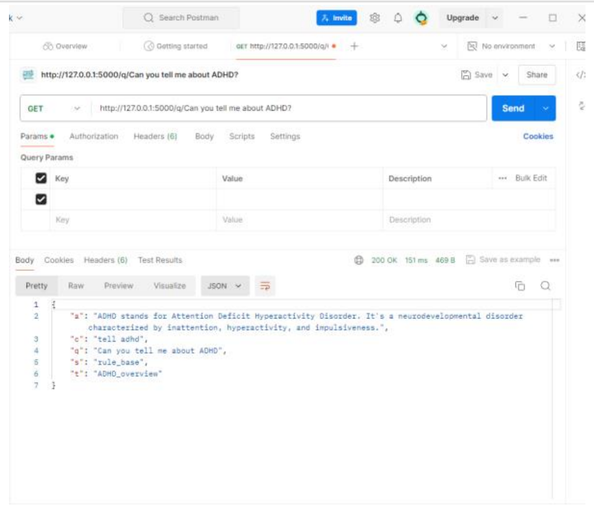
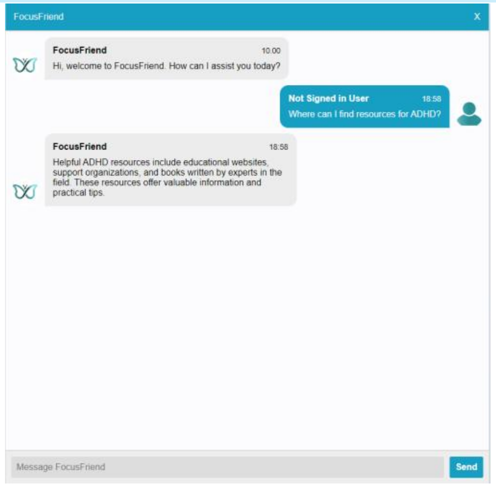
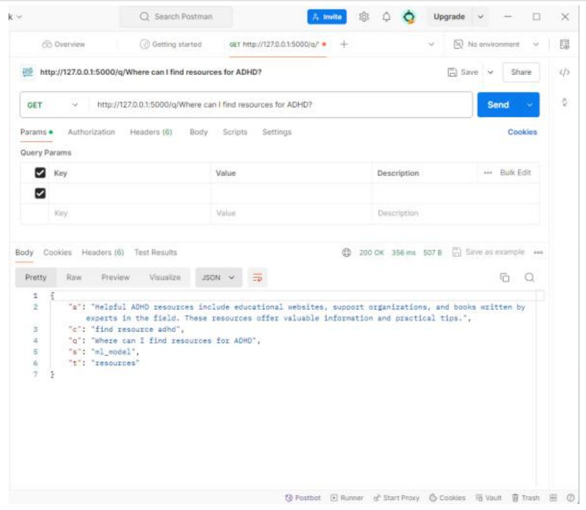
Entering a predefined user input	Enter query "Can you tell me about ADHD?"	One of the predefined responses under the corresponding tag for the user input entered	 FocusFriend
			FocusFriend 10.00
			Hi, welcome to FocusFriend. How can I assist you today?
			Not Signed in User 18:39
			"Can you tell me about ADHD?"
			FocusFriend 18:39
			ADHD is a condition that affects people's behavior, causing them to be restless, have trouble concentrating, and sometimes act impulsively.
			Message FocusFriend Send
			 http://127.0.0.1:5000/q/Can you tell me about ADHD?
			GET http://127.0.0.1:5000/q/Can you tell me about ADHD? Send
			Body 200 OK 151 ms 469 B
			{
			"a": "ADHD stands for Attention Deficit Hyperactivity Disorder. It's a neurodevelopmental disorder characterized by inattention, hyperactivity, and impulsiveness.",
			"c": "tell adhd",
			"q": "Can you tell me about ADHD",
			"s": "rule_base",
			"t": "ADHD_overview"
			}
				Pass	Chatbot's response matches the expected rule-based output

Table 9:Test Case 03

Test Case No: 4		**Test Tile: Invalid Query Testing**			
Description: ML Model testing					
Step	Action	Expected System Result	Actual Result	Pass/ Fail	Comment
Entering a user query which is not under the predefined user inputs	Enter query " Where can I find resources for ADHD?"	Providing an appropriate response by predicting the corresponding intent correctly	 	Pass	Chatbot correctly identifies slight variations to rule-based user input

FocusFriend

FocusFriend 10.00
Hi, welcome to FocusFriend. How can I assist you today?

Not Signed in User 18:58
Where can I find resources for ADHD?

FocusFriend 18:58
Helpful ADHD resources include educational websites, support organizations, and books written by experts in the field. These resources offer valuable information and practical tips.

Message FocusFriend

http://127.0.0.1:5000/q/Where can I find resources for ADHD?

GET http://127.0.0.1:5000/q/Where can I find resources for ADHD?

	Key	Value	Description
✓			
	Key	Value	Description

200 OK 356 ms 507 B

```
{
    "a": "Helpful ADHD resources include educational websites, support organizations, and books written by
        experts in the field. These resources offer valuable information and practical tips.",
    "c": "find resource adhd",
    "q": "Where can I find resources for ADHD",
    "s": "ml_model",
    "t": "resources"
}
```

Table 10: Test Case 04

12.Conclusion

FocusFriend chatbot project has successfully demonstrated the potential of AI driven support for individuals with ADHD. The chatbot's ability to respond to user inputs with appropriate and supportive messages aligns well with its objective to provide companionship and assistance in managing ADHD symptoms. However, the project recognizes areas of further improvement to meet the diverse needs of its users.

11.1Potential Improvements

- **Expanding the Database:**
  More content relevant to ADHD can be added to the database while providing links to external resources such as articles, tips, research studies etc.

- **Model Fine-Tuning:**
  Model fine-tuning involves refining the chatbot's machine learning model by adjusting parameters to improve its performance and enhance its understanding and response accuracy.

- **Enhanced NLP capabilities for Context and Entity Recognition:**
  More advanced NLP techniques can be incorporated to the to be able to understand the context of a conversation and identify specific entities (Eg: names, dates, symptoms, etc.) and provide more accurate and relevant responses.

- **Introducing Additional Features:**

– **Task and Medication Reminders**: features for setting reminders to help users manage daily tasks and medication schedules, addressing common forgetfulness.

– **Cognitive Behavioral Therapy (CBT) Coaching**: Integrating CBT coaching modules with interactive sessions and exercises to provide coping strategies for ADHD symptoms.

13.Source code

13.1 BOT Component

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import tensorflow as tf
import numpy as np
import pandas as pd
import json
import string
import random
import keras

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import LabelEncoder
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer


class ChatBot:

  def __init__(self):
    pass

  def text_preprocess(self, inp_text, translator, lemmatizer, stop_words):
    #STEP1: The letters of the sentence is coverted to a lower case
    i = inp_text.lower()
    #STEP2: Remove all the punctuations
    sentence = i.translate(translator)
    #STEP3: Split the words in the sentence to process those further (vector)
    sentence = sentence.split()
    #STEP4: Remove all stopwords
    stripped_sentence = [y for y in sentence if y not in stop_words]
    #STEP5: Lemmatize each word in the sentence (stop words removed)
    output = []
    for j in stripped_sentence:
      word = lemmatizer.lemmatize(j)
      output.append(word)

    #DEBUG=TRUE
    #print("inp_text -> ", inp_text, " stripped_sentence -> ", stripped_sentence, " output -> ", output)

    return output


  def ask_bot(self, query):

    #query = "What strategies can help adults with ADHD stay ADHD"
    #query = "Who are some famous figure2s with ADHD?"

    with open ('prod_intents.json') as content:
      intent_content = json.load(content)

    tags = []
    tagsforinputs = []
    responses = []
    single_resp = []
    input = []
    rulebase_i_t = []
    cleaned_query = ""

    #Extract Tags and Responses and create lists (one to one map "tag -> Response")
    for intent in intent_content['intents']:
```

```
  tags.append(intent['tag'])
  responses.append(intent['tag'])
  responses.append(intent['responses'])
  single_resp.append(intent['responses'])

for intent in intent_content['intents']:
  for lines in intent['patterns']:
    input.append(lines)
    tagsforinputs.append(intent['tag'])


#Store data in a table form using pandas DataFrame
data = pd.DataFrame({"tags": tags, "responses": single_resp})
data_i_t = pd.DataFrame({"inputs": input, "tags": tagsforinputs})


le = LabelEncoder()
y_train = le.fit_transform(data['tags'])

saved_model = keras.models.load_model('chatbot_2.keras')

#Text Preprocessing
stop_words = set(stopwords.words('english'))
translator = str.maketrans('', '', string.punctuation)
lemmatizer = WordNetLemmatizer()

index_counter = 0

for i in data_i_t['inputs']:
  processed_sentence = self.text_preprocess(i, translator, lemmatizer, stop_words)

  s = ' '.join([str(e) for e in processed_sentence])
  rulebase_i_t.append(s)
  rulebase_i_t.append(data_i_t.iat[index_counter,1])

  #Update the relevant input cell in the DataFrame
  data_i_t.at[index_counter, 'inputs'] = s
  index_counter = index_counter + 1

#Text Preprocessing - Tokenization
tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=2500)
tokenizer.fit_on_texts(data_i_t['inputs'])
train = tokenizer.texts_to_sequences(data_i_t['inputs'])

processed_sentence = self.text_preprocess(query, translator, lemmatizer, stop_words)
processed_sentence = ' '.join([str(e) for e in processed_sentence])
cleaned_query = processed_sentence

##Rulebase - Check if the query matches directly with the intentes Patterns
rulebase_resp = ''
rulebasematch = 1
try:
  rulebase_resp = rulebase_i_t.index(processed_sentence)
except ValueError:
  rulebasematch = 0

if rulebasematch==1:
  rulebase_resp_found_tag = rulebase_i_t[rulebase_resp + 1]
  selected_resp = responses.index(rulebase_resp_found_tag)
  answer = random.choice(responses[selected_resp+1])

  chat_resp = {
    "q": query,
    "c": cleaned_query,
    "a": answer,
    "t": rulebase_resp_found_tag,
    "s": "rule_base"
  }

  return chat_resp
##EndOf RuleBase

processed_sentence = processed_sentence.split()

train_query_input = tokenizer.texts_to_sequences(processed_sentence)
```

```
    train_query_input_2D = []
    train_query_input_2D.append(train_query_input)

    #Check for unknown words and then throw an error
    for i in train_query_input_2D[0]:
      if len(i) == 0:
        answer = "Sorry, I dont understand your query. Please try again with a different query."
        chat_resp = {
        "q": query,
        "c": cleaned_query,
        "a": answer,
        "t": 'error:unknown_word_included',
        "s": "error_handling"
        }

        return chat_resp

    #Trainer ANN has 8 inputs, therefore the number of inputs are hardcoded rather than recalculation it based on the intent file to save time
    train_query_input = tf.keras.preprocessing.sequence.pad_sequences(train_query_input_2D, 8)

    output = saved_model.predict(train_query_input)
    output = output.argmax()

    response_tag = le.inverse_transform([output])[0]
    tagindex = responses.index(response_tag)
    answer = random.choice(responses[tagindex+1])

    chat_resp = {
        "q": query,
        "c": cleaned_query,
        "a": answer,
        "t": response_tag,
        "s": 'ml_model'
    }

    return chat_resp
```

13.2 Training component

```
import tensorflow as tf
import numpy as np
import pandas as pd
import json
import string

from sklearn.preprocessing import LabelEncoder
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

def text_preprocess(inp_text, translator, lemmatizer, stop_words):
    #STEP1: The letters of the sentence is coverted to a lower case
    i = inp_text.lower()
    #STEP2: Remove all the punctuations
    sentence = i.translate(translator)
    #STEP3: Split the words in the sentence to process those further (vector)
    sentence = sentence.split()
    #STEP4: Remove all stopwords
    stripped_sentence = [y for y in sentence if y not in stop_words]
    #STEP5: Lemmatize each word in the sentence (stop words removed)
    output = []
    for j in stripped_sentence:
      word = lemmatizer.lemmatize(j)
      output.append(word)

    #DEBUG=TRUE
    #print("inp_text -> ", inp_text, " stripped_sentence -> ", stripped_sentence, " output -> ", output)

    return output

def train_ann():
    EPOCHS = 1000

    #Text Preprocessing
    with open ('prod_intents.json') as content:
      intent_content = json.load(content)

    tags = []
    inputs = []

    #Extract Tags and Patterns and create lists (one to one map "input -> tag")
    for intent in intent_content['intents']:
      for lines in intent['patterns']:
        inputs.append(lines)
        tags.append(intent['tag'])

    #Store data in a table form using pandas
    data = pd.DataFrame({"inputs": inputs, "tags": tags})

    #Text Preprocessing
    stop_words = set(stopwords.words('english'))
    translator = str.maketrans('', '', string.punctuation)
    lemmatizer = WordNetLemmatizer()

    index_counter = 0

    for i in data['inputs']:
      processed_sentence = text_preprocess(i, translator, lemmatizer, stop_words)

      #Update the relevant input cell in the DataFrame
      processed_sentence = ' '.join([str(e) for e in processed_sentence])
      data.at[index_counter, 'inputs'] = processed_sentence
      index_counter = index_counter + 1

    #Text Preprocessing - Tokenization
    tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=2500)
    tokenizer.fit_on_texts(data['inputs'])
    train = tokenizer.texts_to_sequences(data['inputs'])

    #Text Preprocessing - padding
    x_train = tf.keras.preprocessing.sequence.pad_sequences(train)

    #encoding tags (output of the ANN)
    le = LabelEncoder()
    y_train = le.fit_transform(data['tags'])

    print(y_train)
    return 0

    input_length = x_train.shape[1] #2D array - columns
    print('x_train.shape', x_train.shape)

    vocab = len(tokenizer.word_index)
    output_length = le.classes_.shape[0]
    print('le.classes_.shape', le.classes_.shape)

    no_of_tags = len(set(data['tags']))

    ## Construst ANN model##
    i = tf.keras.layers.Input(shape=(input_length,), name = "chatbot_input_layer") #input layer
    x = tf.keras.layers.Embedding(vocab+1,10)(i)
    x = tf.keras.layers.LSTM(no_of_tags, return_sequences=True)(x)
    x = tf.keras.layers.Flatten()(x)
    #x = tf.keras.layers.Dense(100, activation="relu")(x)
    x = tf.keras.layers.Dense(output_length, activation="softmax")(x)

    model = tf.keras.Model(inputs=i, outputs=x)
    model.compile(loss = "sparse_categorical_crossentropy", optimizer='adam', metrics=['accuracy'])
    train = model.fit(x_train, y_train, epochs = EPOCHS)

    #save the trained model
    model.save('chatbot_2.keras')


train_ann()
```

13.3 Intents

Examples of intents:

Greeting Intent

```
{
    "tag": "greeting",
    "patterns": [
      "Hi Hello",
      "Hello",
      "Hey",
      "Good day",
      "Howdy"
    ],
    "responses": [
      "Hello! How can I assist you today?",
      "Hi there! How are you doing?",
      "Hey! What can I help you with today?"
    ],
    "context_set": ""
  },
```

ADHD Related Intent

```
{
    "tag": "medication",
    "patterns": [
      "What medications are used for ADHD?",
      "ADHD medication",
      "Stimulant drugs for ADHD",
      "Non-stimulant ADHD medication",
      "I'm having trouble with my medication",
      "I keep forgetting to take my medication",
      "I need assistance with my medication"
    ],
    "responses": [
      "Common medications for ADHD include stimulants like Ritalin and Adderall, as well as non-stimulants like Strattera and Intuniv.
Consult with a healthcare provider for personalized advice.",
      "Stimulant medications are the most widely used treatments for ADHD, but non-stimulant medications can also be effective, especially
for those who do not respond well to stimulants.",
      "Managing medication can be challenging. Let's find a solution together. How can I assist you?",
      "Let's work on a strategy to help you remember your medication. What's been difficult for you?"
    ]
  }
```

13.4 User Interface

13.4.1 HTML/CSS

```
:root {
    --body-bg: linear-gradient(to bottom right, #f5f7fa 0%, #c3cfe2 100%);
    --msger-bg: #fff;
    --border: 2px solid #ddd;
    --left-msg-bg: #ececec;
    --right-msg-bg: #169fc2;
  }

  html {
    box-sizing: border-box;
  }

  *,
```

```css
*: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: Helvetica, sans-serif;
}

.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: 5px;
  background: var(--msger-bg);
  box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}

.msger-header {
  display: flex;    justify-content: space-between;
  padding: 15px;
  border-bottom: var(--border);
  background: #169fc2;
  color: #fafafa;
}

.msger-chat {
  flex: 1;
  overflow-y: auto;
  padding: 10px;
  background-color: #fcfcfe;
}
.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:last-of-type {
  margin: 0;
}
.msg-img {
  width: 50px;
  height: 50px;
  margin-right: 10px;
  background: #ddd;
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  border-radius: 50%;
  background-image: url(images/adhdbutterfly.png);
}
.msg-bubble {
  max-width: 450px;
  padding: 15px;
  border-radius: 15px;
  background: var(--left-msg-bg);
 }
 .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-bottom-right-radius: 0;
 }
 .right-msg .msg-img {
  margin: 0 0 0 10px;
 }

 .msger-inputarea {
  display: flex;
  padding: 10px;
  border-top: var(--border);
  background: #eee;
 }
 .msger-inputarea * {
  padding: 10px;
  border: none;
  border-radius: 3px;
  font-size: 1em;
 }
 .msger-input {
  flex: 1;
  background: #ddd;
 }
 .msger-send-btn {
  margin-left: 10px;
  background: #169fc2;
  color: #fff;
  font-weight: bold;
  cursor: pointer;
  transition: background 0.23s;
 }
 .msger-send-btn:hover {
  background: rgb(0, 123, 180);
 }
```

13.4.2 JavaScript

```
const msgerForm = get(".msger-inputarea");
const msgerInput = get(".msger-input");
const msgerChat = get(".msger-chat");



const BOT_IMG = "images/adhdbutterfly.png";
const PERSON_IMG = "images/person.png";
const BOT_NAME = "FocusFriend";
const PERSON_NAME = "Not Signed in User";

//The main - starting point
```

```
msgerForm.addEventListener("submit", event => {
  event.preventDefault();

  const msgText = msgerInput.value;
  if (!msgText) return;

  appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
  msgerInput.value = "";

  QueryChatBot_F(msgText).then(val => appendMessage(BOT_NAME, BOT_IMG, "left", val.a));

});

function appendMessage(name, img, side, text) {
  //   Simple solution for small apps
  const msgHTML = `
    <div class="msg ${side}-msg">
      <div class="msg-img" style="background-image: url(${img})"></div>

      <div class="msg-bubble">
        <div class="msg-info">
          <div class="msg-info-name">${name}</div>
          <div class="msg-info-time">${formatDate(new Date())}</div>
        </div>

        <div class="msg-text">${text}</div>
      </div>
    </div>
  `;

  msgerChat.insertAdjacentHTML("beforeend", msgHTML);
  msgerChat.scrollTop += 500;
}

function botResponse() {
  const r = random(0, BOT_MSGS.length - 1);
  const msgText = BOT_MSGS[r];
  const delay = msgText.split(" ").length * 100;

  setTimeout(() => {
    appendMessage(BOT_NAME, BOT_IMG, "left", msgText);
  }, delay);
}

// Utils
function get(selector, root = document) {
  return root.querySelector(selector);
}

function formatDate(date) {
  const h = "0" + date.getHours();
  const m = "0" + date.getMinutes();

  return `${h.slice(-2)}:${m.slice(-2)}`;
}

function random(min, max) {
  return Math.floor(Math.random() * (max - min) + min);
}

async function QueryChatBot_F(q){
  const url = 'http://127.0.0.1:5000/q/'+q;
  try {
    const response = await fetch(url);
    const resp = await response.json();
    console.log(resp);
    return resp;
  } catch (e){
    console.log("Error detected (custom) : ", e)
  }

}
```

References

1. https://www.nimh.nih.gov/health/topics/attention-deficit-hyperactivity-disorder-adhd
2. https://www.inbenta.com/articles/benefits-of-chatbots-in-healthcare-9-use-cases-of-healthcare-chatbots/
3. Dalili, S., & Zohuri, B. (2020). Revolutionizing Treatment: AI-Driven Noninvasive Approaches for ODD and ADHD. *Independent Psychiatric Registered Nurse, Encino, USA; Galaxy Advanced Engineering, Albuquerque, USA*.
4. 2. Campbell, J., Çerçi, S., & Cecchinato, M. E. (2020). ADHD and Knowledge Work: Exploring Strategies, Challenges and Opportunities for AI. *Northumbria University, Newcastle upon Tyne, UK*.
5. 3. Loh, H. W., Ooi, C. P., Barua, P. D., Palmer, E. E., Molinari, F., & Acharya, U. R. (2020). Automated detection of ADHD: Current trends and future perspective. *Author links open overlay panel*.
6. 4. Christensen, H., Griffiths, K. M., & Korten, A. (2020). Improving adherence in the use of web-based and mobile interventions for mental health and substance use disorders. *Internet Interventions, 5*, 90-95.
7. 5. Mattingly, G. W., Wilson, J., & Rostain, A. L. (2020). A clinician’s guide to ADHD treatment options. *Journal of Clinical Psychiatry*.
8. Mattingly, G. W., Wilson, J., & Rostain, A. L. (Year). A clinician’s guide to ADHD treatment options. *Journal/Book Title*.
9. The ADHD Centre. (n.d.). ADHD support and treatment services. Retrieved from https://www.adhdcentre.co.uk/.