

AN AI CHATBOT TO ENHANCE THE DAY-TO-DAY LIVES AND LIVELIHOOD OF PATIENTS AFFECTED BY ADHD.

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

JOSHUA RONALDINO P (REG.NO. 40110506)

CHRISTEN JONES S (REG.NO. 40110281)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI, CHENNAI - 600119**

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Joshua Ronaldino P (4011506)** and **Christen Jones S (40110281)** who carried out Project work entitled **"AN AI CHATBOT TO ENHANCE THE DAY-TO-DAY LIVES AND LIVELIHOOD OF PATIENTS AFFECTED BY ADHD."** under my supervision from **November 2023** to **April 2024**.

Internal Guide

Dr.P. AJITHA, M.E., Ph.D

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeyapoor Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva Voce Examination held on 16-04-2024

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I, **CHRISTEN JONES S (40110281)**, hereby declare that the Project Report entitled **"AN AI CHATBOT TO ENHANCE THE DAY-TO-DAY LIVES AND LIVELIHOOD OF PATIENTS AFFECTED BY ADHD."** done by me under the guidance of **Dr. P. Ajitha, M.E.,Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 16-10-2024

PLACE: Chennai

A handwritten signature in blue ink, appearing to read 'S. Christen Jones', written over a horizontal line.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **The Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr.L.Lakshmanan M.E., Ph.D, Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. P. AJITHA, M.E.,Ph.D**, for her valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Attention Deficit Hyperactivity Disorder (ADHD) is one of the world's most prevalent behavioral disorders. Symptoms include hyperactivity, impulsiveness, and difficulty in concentrating. Cognitive Behavioural Therapy (CBT) can help to reduce ADHD symptoms. This paper details the development of a software application capable of being used within CBT scenarios to increase ADHD sufferers' levels of attentiveness.

Users of the application try to increase their attention span by conversing with an Artificial Conversational Agent (chatbot), signaling each time it makes a conversational mistake with a button press. This healthcare chatbot system will provide patients with healthcare support online at all times. Companies have increasingly started adopting chatbots as a customer service support system to provide more personalized service experiences to customers. This technology is catching attention in the customer service context because chatbots can support multiple customers at once, can provide multilingual support, are available 24/7 with no extra cost, relieve humans from repetitive tasks, and can respond to a variety of troubleshooting queries without any latency. To gauge a happy life, it is essential to have a healthy body. However, getting a doctor's appointment for any health condition might be difficult.

The objective is to use AI to build a health chatbot that can identify illnesses and provide basic information about them. before calling a physician. Reducing healthcare costs and improving access to medical information through a medical chatbot is possible. Using the application will help ADHD sufferers to increase their attentiveness. Conversation logs produced by the application will provide discussion material for follow-up sessions with CBT professionals.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF ABBREVIATIONS	viii
	LIST OF FIGURES	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
	2.1 Use of Therapy-Based Approach	3
	2.2 Open Problems in Existing System	3
	2.3 Keeping Patients Interested	4
	2.4 Computerised Cognitive Behavioral Therapy	5
	2.5 What are Chatbots?	5
3	REQUIREMENTS ANALYSIS	7
	3.1 Feasibility Studies/Risk Analysis of the Project	7
	3.2 Software Requirements Specification Document	7
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Usage Scenarios	11
	4.2 Designing the Interface	12
	4.3 Modules used in the system	14
	4.4 Project Management Plan	15
	4.5 Setting up the Development Environment	17
	4.6 Building a distributable version	17

4.7	Testing	18
4.8	Participants	19
4.9	The User Testing Procedure	19
5	RESULTS AND DISCUSSIONS	20
5.1	Evaluation	20
5.2	Aims	20
5.3	Objectives	20
5.4	Suggested Objectives	21
5.5	Key Findings	23
5.6	Increased Level Of Attractiveness	23
5.7	A Sympathetic Avatar	24
5.8	Summary	24
6	SUMMARY AND CONCLUSIONS	25
6.1	Conclusions	25
6.2	Limitations of the Study	25
6.3	Recommendations For Future Research	26
	REFERENCES	27
	APPENDIX	29
A	Source Code	29
B	Screenshots	39
C	Research Paper	42

LIST OF ABBREVIATIONS

ADHD	Attention Deficit Hyperactivity Disorder
AIML	Artificial Intelligence Markup Language
A.L.I.C.E	Artificial Linguistic Internet Computer Entity
CBT	Cognitive Behavioural Therapy
CCBT	Computerised Cognitive Behavioural Therapy
GIMP	GNU Image Manipulation Program
MACH	My Automated Conversation Coach
OOP	Object-Oriented Programming
PNG	Portable Network Graphics
SAPI	Speech Application
TTS	Text-to-Speech
XML	Extensible Markup Language

LIST OF FIGURES

Figure No.	FIGURE NAME	Page No.
4.1	Architecture Diagram	13
4.2	Overview of the Interface	14
4.3	A comparison between the Waterfall method and the Agile method	16
4.4	Timeline Chart	16

CHAPTER 1

INTRODUCTION

As AI offers a suitable response to various challenges associated with this disease, it plays a crucial role in mental health. A fundamental concept of the AI-based mental health remedy and its impact on directly affecting social-emotional learning has also been examined in this study with the appropriate material and references.

The many types of AI that may unquestionably aid in general mental health education have already been specifically outlined in this research using all the data from existing books and publications. In addition, an AI-based chatbot platform has been built within the software system to identify various health factors that seem to be directly linked to mental health. The main challenges remained consistent with other AI-related healthcare systems, including privacy, confidentiality, and data integrity concerns. This research paper and project aims to solve the modern problems of people affected with ADHD (Attention-deficit/hyperactivity disorder).

This project aims to solve these problems that occur in a patient's day-to-day life, not predict but to find a cure or make life easier for the people that are affected by this problem. The Current artificial intelligence has developed to a point where programs can learn from humans and effectively simplify human conversations which is essential. One of the best-known examples of chatbots in recent history is Siri, the AI assistant that is part of Apple's standard software for its products. Siri took chatbot mainstream in 2011. Since then, brands in every sector have started to be conversational in user experience. This refers to an end-user experience in which your interaction with a firm or service is automated based on user prior behavior. If users are developing artificial intelligence applications like Alexa, which enables the use of voice to control devices. If you are a user, you can already interact with this Artificial Intelligence chatbot on popular messaging platforms like Facebook, Instagram and so on. Nowadays the use of chatbots has spread from user customer service to life and death risks. Chatbots are coming into the healthcare industry and can help to solve health problems.

Health and fitness chatbots have begun to gain popularity in the market. Previous year Facebook started allowing healthcare industries to create Messenger chatbots which would then communicate with users.

A text-to-text medical chatbot involves patients in online conversation considering their health problems which provides a set of personalized diagnoses based on their provided symptoms. These bots connect with the potential patients visiting the site, helping them discover specialists, book appointments, and get access to the correct treatment. This chatbot uses Natural language processing techniques to process and analyze the data and give the output inappropriately. It brings up the disease-related problems about whether the task mentioned above should be assigned to human staff. This healthcare chatbot system will always provide patients with healthcare support online. It helps to generate health data and automatically delivers the information of reports to medical management. By asking the questions in series it helps the patient by guiding what exactly the user is looking for.

Attention Deficit Hyperactivity Disorder (ADHD) is one of the world's most prevalent behavioral disorders. Studies found that it affects 5% of children (Polanczyk, 2007) and 3% of adults. (Fayyad, et al., 2007) Symptoms of ADHD include hyperactivity, anxiety, impulsiveness, and difficulties in concentrating. These symptoms can strongly interfere with the interpersonal relationships of an ADHD sufferer, affecting their performance at school or work. As there is no consensus agreement as to what causes ADHD, current treatments focus upon providing relief from symptoms. Patients are usually treated with a combination of medication and cognitive behavioral therapy (CBT). CBT sessions allow patients to acquire coping skills by talking through their problems with a trained therapist. These sessions often include elements of role-play where patients learn to see their behavior through the eyes of others. Computerised Cognitive Behavioural Therapy (CBT) is a form of treatment which provides CBT through technology. These software-based solutions, generally designed to increase mental focus, can be used to support traditional forms of therapy. Chatbots are computer programs designed to simulate a conversational partner. Users type in questions or statements and the program parses their input to find and return an appropriate response.

CHAPTER 2

LITERATURE SURVEY

There is no consensus as to what causes ADHD, but studies suggest many factors are to blame including genetics, brain injuries, and environmental factors. (National Institute of Health, 2014) NHS-approved ADHD treatments focus primarily on providing relief from symptoms. In the United Kingdom, doctors usually prescribe a combination of medication and therapy. (National Health Service, 2017) Medications are effective but can cause side effects such as mood swings, headaches, and insomnia. (Cascade, et al., 2010) Patients can also become dependent upon the drugs used to control their symptoms. (Graham, et al., 2011) Medications offer only temporary relief from symptoms and many find that therapy offers a longer-term solution.

2.1 USE OF THERAPY-BASED APPROACH

Cognitive Behavioural Therapy (CBT) sessions can help reduce ADHD symptoms. During these sessions, which often include elements of role-play, patients talk through their issues with a therapist to acquire coping skills. A 2014 study (Knouse, 2014) examined how skills-based treatments are given to adults with ADHD. Knouse presents motivation as a key factor in acquiring new skills. Patients must see the value of coping skills in practice and implement them through doing rather than understanding them purely theoretically. A related study (Knouse & Safren, 2010) found evidence that skills-based training provides ADHD sufferers with greater symptom relief than therapy alone. Skill exercises performed at home improved participants' abilities to regulate behavior and increased their capacity to cope even in the presence of negative emotional influences.

2.2 MINDFULNESS TRAINING AS AN ADJUNCT TO THERAPY

Mindfulness is a practice, often based upon Buddhist teachings, which trains people to remain consciously aware of who they are, where they are, and what

they are doing by observing what is going on around them and inside their minds. Mindfulness exercises commonly focus on observing breath patterns and paying non-judgemental attention to thoughts and emotions. (Moore, 2017) These meditational practices are thought to have a strong effect on developing the basic attentional processes. (Wu, et al., 2013) The Clinical Handbook of Mindfulness (Didonna, 2009) examines the subject in greater depth. It describes how the formal practices which develop mindfulness use narrow foci as an anchor to stop the mind from drifting. Firstly, the subject brings attention to the focus of the exercise, normally a form of sensory input such as breathing. If attention wanders, the subject notes that a distraction has occurred and lets it go. Focus is then placed back upon the attentional anchor. An article in the Futurist journal (Docksai, 2013) reports that Mindfulness Training helps ADHD sufferers cope with mind-wandering. Docksai also found that mindfulness can help students relax during tests and develop increased sensitivity to the needs of those around them. Another study (Mrazek, et al., 2012) found that daily participation in mindfulness exercises can reduce mind-wandering. During a two-week program, subjects were taught to focus on their breathing. Tests performed after initial training showed that improvements to concentration developed in subjects during the program could be redirected towards more challenging tasks.

2.3 KEEPING PATIENTS INTERESTED

ADHD sufferers often become bored with repetitive activities. (Kass, et al., 2003) When patients lose interest in treatment programs, this creates an obstacle to affecting positive change. Researchers at the University of Aarhus (Sondergaard, et al., 2016) found that of 151 people attending an ADHD unit 27% dropped out of treatment before completion and 42% missed 3 or more appointments. Software applications can provide one solution to the problem. Two studies on computer-assisted instruction (Mautone, et al., 2005) (Botsas & Grouios, 2017) found that in addition to keeping users interested, it can increase sustained attention and improve the school and work performances of ADHD sufferers. As 90% of United Kingdom households have Internet access (Office for National Statistics, 2017).

2.4 COMPUTERISED COGNITIVE BEHAVIORAL THERAPY

Computerised Cognitive Behavioural Therapy (cCBT) is a method of providing CBT using technology. The majority of cCBT solutions are currently mobile phone applications or websites designed to fit into a patient's schedule. Applications to increase mental focus can be used with little guidance and produce measurable improvements. One study of a computerized cognitive training program (Smith, et al., 2009) found that it significantly enhanced the working memory and attention spans of a group of participants aged 65 and older. During training, participants performed six daily brain plasticity exercises. Exercises included elements of pattern matching, sequence recognition and identifying details which had been presented verbally as a short story. Although some participants exhibited symptoms of fatigue, headaches and frustration related to the exercises, most showed large improvements in their overall memory and information-processing skills. Related projects showing similar results include (Barnes, et al., 2010), (Gray, et al., 2012) and (Sahakian, et al., 2015) CCBT applications may also help people to develop their social skills. Slovák, Gilad-Bahrach and Fitzpatrick (Slovák, et al., 2015) interviewed and conducted design workshops with researchers and developers of Social and Emotional Learning programs for children to identify how curricula could benefit from the inclusion of supportive digital technologies. Workshop participants highlighted their belief that assistive technologies could be used to help learners develop skills and then removed as these skills became ingrained into their personalities.

2.5 WHAT ARE CHATBOTS?

Chatbots are computer programs designed to simulate a conversational partner. Users converse with the program which parses their input and provides appropriate responses. Chatbots can provide an alternative approach to therapist-led CBT sessions. Researchers at M.I.T. (Hoque, et al., 2013) developed and tested a virtual conversational agent to provide social skills training and improve the performance of job interview candidates. The MACH

(My Automated Conversation Coach) application allows users to participate in mock interviews. It uses an animated interviewer character to ask a series of preprogrammed queries based on common interview questions. Test participants took part in three mock interviews with the agent. The application then showed them videos of their performance alongside real-time information about the verbal and non-verbal behaviors which they exhibited in response to cues provided by the virtual agent. The researchers found this self-reflective feedback had positive effects. Participants who used the MACH application scored much better in follow-up interviews than a control group showed 30 minutes of educational videos. Part of the reason for MACH's success was due to the interviewer's avatar. It was designed to look artificial to overcome feelings of uneasiness which people often experience when interacting with objects that appear almost human.

This phenomenon is referred to as the 'uncanny valley' effect. (Mori, 2012) In his essay, Mori predicts that deliberately pursuing non-human designs will make it possible to create artificial entities with which humans can feel greater levels of affinity. Socially awkward patients may prefer computer-guided therapy as it spares them the embarrassment of face-to-face meetings with health professionals. (Kofmel, 2017) However, related research found that therapy programs with no human contact have higher drop-out rates than clinician-guided ones. (Farvolden, et al., 2005) and although people can benefit from applications used alone at home, most will need some form of human guidance to gain the maximum benefit from them. (Marks, et al., 2007)

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Feasibility Study:

1. **Technical Feasibility:** Determine if the technology required for the project is readily available and can be implemented effectively. This includes assessing the compatibility of the AI proctoring software with the existing infrastructure and whether the required hardware and software are accessible.
2. **Operational Feasibility:** Evaluate whether the project can be integrated into the daily operations of educational institutions. Consider factors like the ease of deployment, maintenance, and the ability of staff to operate and support the system effectively.
3. **Economic Feasibility:** Analyze the project's cost-effectiveness. This includes estimating the initial setup costs, ongoing maintenance expenses, and potential savings or additional revenue generated from implementing the system. It's important to assess whether the benefits outweigh the costs.
4. **Legal and Regulatory Feasibility:** Determine if the project complies with legal and regulatory requirements, such as data protection laws, privacy regulations, and educational standards. Identify any legal hurdles and assess the feasibility of addressing them.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

The application is built upon four major elements;

- The Python Programming Language.
- The PyGame multimedia library.
- Artificial Intelligence Markup Language.

- The PyAIML library.

This section looks at these elements and details the development of the application.

3.2.1 Python

Python is an interpreted high-level language popular amongst the scientific community. Part of the reason for this popularity is due to the large number of data analysis and machine learning libraries freely available from the Python Package Index repository. Well-written Python code is clear, concise and easy to maintain. It allows developers to use an Object-Oriented Programming (OOP) paradigm for creating interactive user interfaces. OOP simplifies keeping track of the states of different objects, and the Masumi application needs to run many processes simultaneously including animations, sound, interaction with the chatbot's kernel, and event handling for mouse and key presses.

3.2.2 PyGame

Although Python runs slower than compiled languages such as C or C++, it is easy to extend with compiled extensions written in different programming languages. The PyGame library is one such extension that allows developers to incorporate sound and graphics routines into their applications.

PyGame provides graphic applications with fast screen updates by allowing them to make calls to optimized C and Assembly code that runs over 100 times faster than pure Python. PyGame runs on all major platforms and can be used with graphic renderers including DirectX, X11 and OpenGL. Aside from handling graphics, it contains methods for detecting object collisions and playing sound effects.

3.2.3 AIML

Chatbots process natural language to formulate and return a response. ELIZA, one of the first programs to attempt this, simulates a mock Rogerian psychotherapist. (Weizenbaum, 1966) ELIZA tokenizes user input and calculates

the best response based on keyword distribution. As the program has no real-world knowledge or sense of context, it cannot answer questions or remember what it has been told previously. Artificial Intelligence Markup Language (AIML) is an extensible markup language (XML) which improves upon the ELIZA concept by allowing programs to import knowledge trees. (Wallace, 2003) (see Appendix K) These knowledge trees allow AIML to store responses to an unlimited amount of queries, and the specification enables programs to keep track of what is being said by users. The free availability of the open-source AIML library provided by the A.L.I.C.E Foundation⁷ was a key factor in choosing an AIML solution over alternatives such as RiveScript or ChatScript. Whilst these alternatives can produce chatbot personalities closer to that of an actual ADHD sufferer, they require these personalities to be created from scratch. This is an option which could be considered for future versions.

3.2.4 PyAIML

AIML is based on the stimulus-response model. Input (stimulus) provided by the user is matched with a pattern stored in a knowledge tree and a response is returned. (Mikic, et al., 2008) AIML interpreters provide an interface between AIML files and the applications using them. Whilst it is possible to create an AIML interpreter from scratch, many open-source interpreters are available in languages including Java, C++, Python and PHP. Using a third-party interpreter reduced development time at the cost of losing the extra features which could have been added to a bespoke implementation. As the Masumi application uses Python, the PyAIML interpreter was chosen. PyAIML is an open-source AIML interpreter implemented in Python. It contains all the features necessary to build and interact with a kernel created from AIML files. The version of PyAIML used is a fork of the original, translated by developer Konstantin Weddige from Python version 2 to Python 3.8 Although PyAIML is not supplied with extensive documentation, the library is well-commented and simple to use.

A chatbot kernel can be implemented in 6 lines of code;

Import the PyAIML library

```
import pyaiml
```

Create the PyAIML kernel and load the aiml files

```
kernel=pyaiml.Kernel() kernel.bootstrap(learnFiles="alice-startup.xml")
```

```
kernel.respond("load aiml b")
```

Respond to user input

```
while True: print( kernel. respond(input("Enter your message >> ") ) )
```

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 USAGE SCENARIOS

A usage scenario is a short list of the typical steps and actions involved during interaction with an application. Rather than describing how applications are implemented, usage scenarios concentrate on how they will be used. Software projects are complex and often get sidetracked by minor issues. This can cause them to run over schedule or fail to implement functionality in a manner acceptable to clients. A well-drafted usage scenario can communicate an application's needs clearly to its development team. It must contain a set of concisely written goals so that at any stage team members can examine whether their work fulfills project requirements. Three usage scenarios were identified for the application.

Therapist usage scenario

- Introduce the application to an unskilled user explaining its purpose.
- Explain how to run and use the application.
- Give the patient a copy of the application.
- Alternatively, the patient uses a version of the application installed on the therapist's computer.

Patient interaction scenario

- Install the application.
- Whenever the chatbot makes a conversational mistake, press a button to signal that it was recognised.
- All conversations and button presses are written to a log file.
- Close the application smoothly. Post-usage analysis scenario
- Patient presents a log file to the therapist.
- The therapist discusses a chatbot session with the patient to help them make better decisions next time.
- Patient applies lessons learned to their own life
- Run application.
- Talk to the chatbot as instructed by the therapist.

4.2 DESIGNING THE INTERFACE

Having identified user requirements, research was conducted for the interface design. A review of existing solutions showed that popular chatbot interfaces contain three major elements: a user input box, a chatbot response box, and an image or avatar. With this knowledge in hand, the design of the interface commenced. The design process began by creating pencil sketches. These rough designs were then refined using a professional online wireframing tool. Wireframes allow designers to plan the layout and functionality of their interfaces without becoming lost in details. (Grant, 2004)

After the basic layout had been finalized, different color schemes were tried out on a mockup of the interface. A mixture of dark colors and soft pastel pinks was chosen for the final design. This decision was influenced by the *Sandman* graphic novel series by British author Neil Gaiman; more specifically the color scheme used for the *Delirium* character. *Delirium* exhibits ADHD symptoms taken to the extreme. She is a chaotic and highly unfocused character whose 'hair and clothes shift from minute to minute with the only things being relatively consistent [being] her multi-coloured hair and eyes.' Next, the chatbot avatar was designed. Avatars can reduce the barriers and preconceptions between users and applications which arise in text-only interfaces. Whilst some chatbot applications attempt to engage users by emulating texting applications, those that create a deeper sense of interpersonal closeness tend to include an avatar. Keeping Mori's discussion of the uncanny valley effect in mind (Mori, 2012), it was decided that the avatar should be created in an iconic style to help reduce the uneasiness that first time chatbot users sometimes feel. As artificial entities with a humanlike appearance are attributed a level of experience which can escalate these feelings of uneasiness, the avatar design includes elements to convince users that it is naïve and less experienced than them. After much experimentation, it was concluded that a young female character with feline characteristics would be suitable for the avatar. Animal characteristics have been found to put people at ease and alter their tendency to focus on negative aspects of their character. A cat girl avatar was sketched with pencil and ink onto paper, then scanned and redrawn using the GIMP open-source image editor v4. This image would serve as the model for the animation designs.

Architecture Diagram.

This visual map of our processes can guide our actions, identify roadblocks, and streamline our activities for better efficiency.

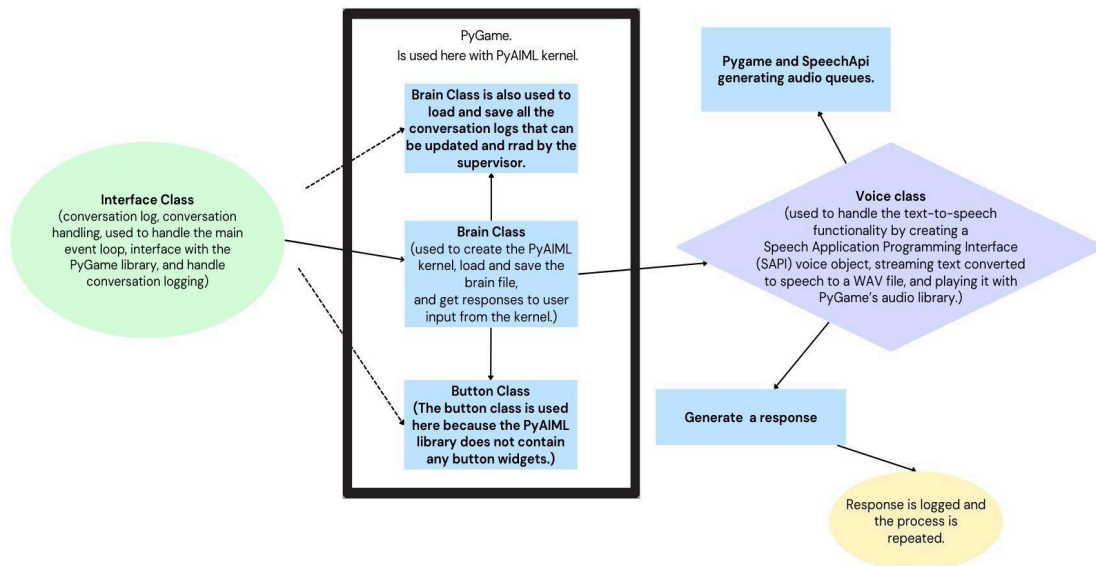


Fig: 4.1: Architecture Diagram

The first animation created for the avatar was a basic blink to prompt user interaction. Blinking reinforces the idea that conversation is a two-way activity, with short blinks drawing longer answers from the conversational partner. A head movement animation was added to reassure users that the chatbot pays attention and nine transparent PNG5 images of mouth shapes were created for lip synching with the text-to-speech conversion of the chatbot's replies. Lip-synching is the process where a character's mouth movements are matched to the phonemes of a speech track. (Sanders, 2018) Sanders identifies nine basic phoneme shapes for the English language - A, E, FV, LN, MBP, O, TS, UQ and WR. A simple line was drawn onto the chatbot avatar to represent the mouth at rest. The final part of designing the interface was choosing text fonts. Notepad font is used for text boxes and buttons, and Beyond Wonderland for the application's logo. Decorative fonts such as these commonly decorate book covers, and fit into the interface's comic book aesthetic. With the graphical user interface fully designed development could now begin in earnest.



Fig: 4.2: Overview of the Interface

4.3 MODULES USED IN THE SYSTEM

Development was to follow an object-oriented approach. Object-oriented programming is a paradigm modeled after the way objects interact with each other in the real world. Each object is a self-contained entity made from data which describes its attributes, and methods which allow it to perform operations. Within an Object-Oriented application, objects can communicate with one another or remain unaware of one another's existence. Hiding objects' inner workings make them simpler to incorporate, and a stub can be put in place during development to simulate their functionality so work can start upon other classes. This modular approach allows application components to be split up amongst a team for faster development. For this project, it allowed work upon the interface to commence whilst research was still being performed upon how to implement text-to-speech functionality and the PyAIML kernel. After examining project objectives and looking at the interface storyboard, it was decided to handle the application's functionality using 4 classes.

- The MasumiInterface class – used to handle the main event loop, interface with the PyGame library, and handle conversation logging.

- The MasumiBrain class – used to create the PyAIML kernel, load and save the brain file, and get responses to user input from the kernel.
- The MasumiVoice class – used to handle the text-to-speech functionality by creating a Speech Application Programming Interface (SAPI) voice object, streaming text converted to speech to a WAV file, and playing it with PyGame’s audio library.
- The Button class – this would need to be created as PyGame does not contain its own button widget class. Reference tables were created as methods and instance variables were added to the classes.

A folder structure was imposed to help manage the many source files and images used by the application. As work progressed, difficulties were encountered with text positioning. It was decided to include a third-party function to resize text and place it onto the interface within a PyGame surface object. All modules were tested and debugged before being incorporated into the working version of the application. The modular approach paid off, creating a finished application which is easy to maintain and extend.

4.4 PROJECT MANAGEMENT PLAN

Development projects must use some form of management technique to finish on schedule. Two project management methodologies were considered for developing the application, the Waterfall and Agile methods. The Waterfall Method keeps projects following a rigid sequence of pre-planned stages, only allowing team members to revert to a previous stage when necessary. Each stage is assigned start and end-points known as milestones. Agile methodology is more flexible than the Waterfall method. It uses iteration to speed up the development of working applications. Projects are broken into small tasks, usually worked on in intensive sprints of between two and four weeks.

The Waterfall method was chosen for this project for several reasons. As the project is the work of a single developer there was no need to coordinate the efforts of team members working on different modules. Project requirements were also unlikely to change during development. The third-party libraries used by the application are stable. Each has been rigorously tested by the open-source software community for over ten years and is compatible with the current version of the Windows 10 OS.

Although the use of the Agile methodology may have resulted in an application better suited to clients' needs, the decision to follow the structured Waterfall method helped complete the planned application on schedule.

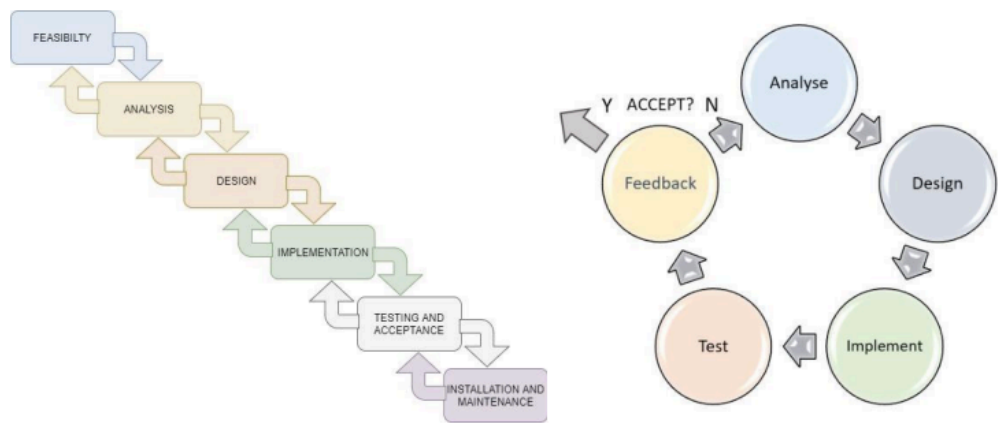


Fig: 4.3: A comparison between the Waterfall method and the Agile method.



Fig: 4.4: Timeline Chart

4.5 SETTING UP THE DEVELOPMENT ENVIRONMENT

To begin setting up the development environment, the latest version of Python 3 was downloaded and installed upon a desktop running the Windows 10 operating system. Next the PyCharm Integrated Development Environment Community Edition was installed. PyCharm Community Edition is free, well-maintained, and easy to use. It has many features which proved useful during development, for example, the intelligent refactoring functions which allow variable names to be updated application-wide. Finally, PyGame was installed via PyCharm's package manager, and the PyAIML library downloaded from its developer's GitHub page¹⁰. Work could now start upon developing the application.

4.6 BUILDING A DISTRIBUTABLE VERSION

After the final version of the application was ready a distributable setup file was created for the Windows 10 operating system. The first issue to resolve was the fact that Windows 10 has no native support for Python and although installing it is simple, non-technical people can be put off by the process.

PyInstaller¹¹ is an application which allows developers to create Windows, Linux and MacOSX executable files from Python programs. The PyInstaller application starts by analyzing the Python script to discover any dependencies required for it to run. Copies of these files are bundled with the script and a copy of the Python interpreter into a distributable folder. An executable file is then created which allows the application to be run with a single click.

With Python added to the Windows PATH environment variable, PyInstaller was run from a command-line shell opened in the same directory as the Masumi script. Once a distribution folder had been created, directories containing the avatar images, fonts and a prebuilt brain file were added. The Inno Setup ¹² application could then be run to package the distribution files into a simple installation program. This executable file installs the Masumi application files, creates a directory

structure, adds a program entry into the Start Menu, and creates a Desktop shortcut.

The installation application was then burned onto a compact disc along with a copy of the Masumi application source code, the GPL 3 license agreement, and a simple PDF installation instructions file. The application was now ready to be tested and, if necessary, revised.

4.7 TESTING

This section looks at tests performed upon the finished application. Testing was divided into a series of two phases to provide quantitative and qualitative feedback upon whether the application had met its objectives.

4.7.1 Formal Testing

Before commencing development, a suite of test cases was drafted to ensure the application would meet its requirements. The test plan includes visual inspections, input and output checks, and functionality tests. Test cases are generic enough to be reused if any third-party libraries are replaced or structural changes made to the application. Formal testing was carried out upon a working version of the application. Results can be found within the test log. As each class and function was vigorously tested before it was added to the working version of the application, the final version managed to pass all tests without error.

4.7.2 End User Testing

A small group of beta testers were recruited and taught to use the application. After using the application for ten minutes, participants provided the researcher with subjective feedback about their experience.

4.7.3 Test Documentation Design

Prior to recruiting participants for user testing, a participant Information sheet, consent form and application usability questionnaire were designed. The participant information sheet outlines the nature of the project, the role of participants including their rights and obligations, and details about how the feedback provided would be used.

4.8 PARTICIPANTS

A potential problem with recruiting end user testers was identified during the requirements analysis phase. Receiving ethical clearance to test the application upon actual ADHD sufferers would have proved difficult because the majority are below the age of 18. Ethical approval was granted after the proposed testing group was changed to adults from within the college who would be instructed to play the roles of ADHD sufferers. All test participants were college-educated males above the age of 18 from a diverse range of economic and cultural backgrounds.

4.9 THE USER TESTING PROCEDURE

Candidates for end user testing were approached and provided with a participant information sheet. After reading the document and having any questions answered by the researcher, those that wished to take part completed a consent form. From those approached, only three agreed to participate in the study. Candidates that refused stated that they were too busy to take part due to work or study commitments.

Participants were first instructed how to use the application, then held a 10-minute conversation with the chatbot whilst playing the role of an ADHD sufferer. After interacting with the application, participants completed a questionnaire about their experience.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 EVALUATION

This section examines whether the initial aims and objectives of the application were met and provides suggestions for improvements.

5.2 AIMS

This project aimed to build an application capable of being used within CBT scenarios to increase ADHD sufferers' levels of attentiveness. Below is a table which breaks up this aim into its three constituent parts, justifying that each part was met by the final application.

To Build an application:

The application was designed, developed and tested on schedule. It contains all the functionality expected from it.

To be capable of being used within CBT scenarios:

A single executable file was created for the application which allows it to be installed by a therapist, or upon the ADHD sufferer's own machine

To increase ADHD sufferers' levels of attentiveness:

The method of using buttons to register when the chatbot makes a conversational mistake can increase levels of attentiveness. Participants paid increasing levels of attention to looking for flaws whilst using the application than during normal conversation.

5.3 OBJECTIVES

Below is a list of the objectives for the application and a justification that they were met by the final product.

- The application was developed and tested upon three different computers

running the Windows 10 operating system. No compatibility issues were encountered.

- The interface uses elements common to web browsers and chat applications so should feel familiar to most users. Each element serves a specific purpose without obscuring other elements.
- Buttons have mouseover and mouse click animations to show that they are interactive. The use of the PyAIML and the A.L.I.C.E AIML libraries allows natural language to be parsed and return meaningful replies.
- Text-to-speech has been implemented using a Windows SAPI object and the PyGame audio library. Words are converted into speech and spoken clearly by an English voice provided by the Windows 10 operating system.
- All conversations and button presses are written to a log file stored on the user's computer.
- Source code follows the conventions outlined in the PEP 8 Style guide. (van Rossum & Warsaw, 2013) It is well-commented and divided into a collection of manageable classes and methods. Files are stored within a well-named directory structure.
- An installer was created using Inno Setup and tested on three Windows 10 machines without Python 3 added. The application installed smoothly and ran without problems on all machines.

5.4 SUGGESTED OBJECTIVES

An introduction screen should be displayed when the application is started up to assure users that it is running. At present there can be a delay of up to 30 seconds, depending upon the specification of the host computer. This delay is caused by the need to load the brain file into the PyAIML kernel. During testing, several participants found using the text input box difficult. They complained that it was not instantly identifiable, and that the font used was difficult to read. This issue can be resolved by adding a flashing cursor at the point of text insertion, changing the text box background to a lighter color, darkening the text color, and using a more conventional font such as Arial or Calibri. The font on the feedback buttons will also need to be changed.

Adding a menu bar will allow users to customize the application to suit their needs. Basic functionality should include closing the application, viewing help files, and reading information about the project and its development. Users should be able to toggle the logging functionality on and off from the menu bar and choose where the log files are saved. Although providing therapists with a logfile is seen as an integral part of the therapeutic use of the application, users may want to talk with the chatbot outside of this context to improve their general conversational skills. Another feature which users should be able to toggle on and off is text-to-speech. It should also be possible for users to change the pitch or speed of the voice from the menu bar, but this would also require changes to be made to the lip synchronization timings for the mouth animations.

At present there is only one chatbot personality available for use with the application. This is based upon the default A.L.I.C.E AIML library. These files are however fully customisable, allowing responses to questions and the ways it parses natural language to be altered to create a new personality. New chatbot personalities would require animations, mannerisms, and SAPI voice tags to be created to differentiate them from the current chatbot. Implementation of this feature would be more suited to a development team than a single developer. If this feature is added, users should be able to select which personality they wish to speak to via an introduction screen, or the proposed new menu bar.

Versions of the application should be made for the MacOS, Linux and Android operating systems. Although most of the current libraries used by the application are compatible with these platforms, difficulties will be created by its dependence upon the Microsoft SAPI text-to-speech libraries.

Individual versions could be created for each platform using native TTS libraries or by including a third-party library. Alternatively, a wrapper library which provides cross-platform access to TTS across different platforms such as pyttsx3 could be used. Although the current version of the application will be fit for purpose after a few minor modifications, it still requires extensive user tests involving input from diagnosed ADHD sufferers and their therapists before a marketable version can be produced.

5.5 KEY FINDINGS

This paper originally proposed that focussing attention on finding a chatbot's inappropriate responses could help users to develop their attentiveness. The results of user testing confirm this hypothesis to a certain degree but stress that more extensive testing is required before the application is suitable for open deployment.

The key findings of the project were;

- Levels of attentiveness increased in participants during user testing.
- Participants did not become bored during user testing and were comfortable conversing with the chatbot.
- Without the guidance of a trained therapist, participants may become distracted from the purpose of the exercise.

Individual versions could be created for each platform using native TTS libraries or by including a third-party library. Alternatively, a wrapper library which provides cross-platform access to TTS across different platforms such as pyttsx3 could be used. Although the current version of the application will be fit for purpose after a few minor modifications, it still requires extensive user tests involving input from diagnosed ADHD sufferers and their therapists before a marketable version can be produced.

5.6 INCREASED LEVELS OF ATTENTIVENESS

Chat Logs created during user testing (see Appendix S) indicate that participants' levels of attentiveness increased during their use of the application. For example, in excerpt 1 the participant is focused on the task given. He or she notices the chatbot's first inability to understand what was said immediately and signals this by pressing the appropriate button. Later in excerpt 1, the participant notices that the chatbot is using a generic response to a certain type of question. This observation is tested and then signaled by using the correct button press. The participant was able to remain focused on the purpose of their interaction with the chatbot.

5.7 A SYMPATHETIC AVATAR

Although instructed to converse with the chatbot for five minutes, participants wanted to use the application for much longer. The interface design and chatbot's personality kept participants interested throughout the test. During conversations with the chatbot, there were no long pauses whilst participants searched for the correct words to use. Knowing that their conversational partner was a computer program allowed testers to openly question what was being said. For example, in excerpt 2 the participant examines the chatbot's logic in a fast exchange until the program makes the mistake of changing the subject.

The choice of avatar design played a major role in keeping participants interested and interactive during their use of the application. Using a young female character with cat traits helped participants overcome the inhibitions sometimes felt when using a chatbot for the first time. This concurs with the research findings that animals can alter people's tendency to focus on the negative aspects discussed in the literature review, and Mori's prediction that by deliberately pursuing non-human designs it would be possible to create artificial entities which humans could feel greater levels of affinity towards.

5.8 SUMMARY

The findings from these preliminary tests support the developer's assertion that using the application could help ADHD sufferers increase their attentiveness. BSCTUC-6PD: Masumi - An Artificial Conversational Agent to help increase attentiveness in ADHD sufferers. Log file excerpts also show that the use of the application can reveal aspects of a user's personality which may not appear within the formal atmosphere of the CBT therapy session. The application was found to be fit for purpose. It can now be developed as part of a therapeutic approach which enables patients to examine and correct their interpersonal exchanges within a safe environment.

CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 CONCLUSIONS

This paper has documented the development and testing of a new approach to increasing attentiveness in ADHD sufferers. Test results indicate that attentiveness increases during use of the application and that it could be used within CBT scenarios involving a professional therapist. Masumi is designed to complement rather than replace traditional forms of CBT. It is intended to be used outside of a formal therapeutic environment and will allow patients to develop their conversational focus at home. One of the main benefits which the application will provide users is its portability. It can be used at any time of day when the patient is ready rather than having to depend upon the schedules of other people. ADHD sufferers can use the application for as little or as long as they want, it will always be available to them. Examination of the logged conversations between chatbot and patient created during short five to ten-minute sessions will provide therapists with material for evaluating their patient's progress.

6.2 LIMITATIONS OF THE STUDY

The major limitation of this study was its lack of access to actual ADHD sufferers. This was due to difficulties in gaining ethical approval. The researcher needed to compensate by gathering information from secondary sources, and by instructing test participants to play the role of an ADHD sufferer. Whilst participants played their roles convincingly enough, a lack of experience with ADHD may have led them to portray its symptoms in a stereotypical rather than informed manner. The number of participants who agreed to take part in testing was very disappointing and from a very narrow section of society. There were no female participants and the age range of the group was well above that of typical ADHD sufferers due to limitations imposed upon the study by the college's ethical research panel.

Another issue was the lack of collaboration from a professional CBT therapist. As therapists will be the main consumers of data generated by the Masumi application, their input would have proved invaluable.

6.3 RECOMMENDATIONS FOR FUTURE RESEARCH

Whilst this project has shown the potential benefits which the approach could provide to ADHD sufferers, further research must be performed before the application is ready for use within real-life scenarios. Future research needs to include long-term studies, and a larger group of test subjects should be recruited from a wider demographic range to judge whether the application is suitable for all users. Test subjects must be screened in advance to identify any potential mental health issues that may arise due to the unsupervised aspect of their interaction with the chatbot. Also, as during personality development hidden mental issues may come to the surface, 24-hour support must be made available to subjects and their immediate family. Research should be performed into the efficacy of alternative interface designs, and the effects of different chatbot avatars and personalities upon user interaction. Implementations for different platforms and technologies should also be examined. ADHD affects people from all walks of life. Its effect upon sufferers can be compounded by unfocussed environments such as television or the Internet. Whilst medications can provide a short-term solution to symptoms such as anxiety and hyperactivity, the concentration problems caused by ADHD can stop sufferers from achieving their true potential. It is hoped that the results of this project and the research which follows it will motivate the creation of alternative approaches to the treatment of ADHD symptoms; approaches which will develop the confidence and critical skills of people who were until quite recently wrongly labeled as lazy and disruptive elements.

REFERENCES:-

1. J. Berrezueta-Guzman, I. Pau, M. -L. Martín-Ruiz and N. Máximo-Bocanegra, "Assessment of a Robotic Assistant for Supporting Homework Activities of Children With ADHD," in IEEE Access.
2. C. Nash, R. Nair and S. M. Naqvi, "Machine Learning in ADHD and Depression Mental Health Diagnosis: A Survey," in IEEE Access.
3. S. Batsakis, E. Papadakis, I. Tachmazidis, T. Chen, G. Antoniou and M. Adamou, "Neuro Intel: A System for Clinical Diagnosis of Attention Deficit Hyperactivity Disorder (ADHD) Using Artificial Intelligence," 2023 IEEE Symposium on Computers and Communications (ISCC).
4. C. Nash, R. Nair and S. M. Naqvi, "Machine Learning and ADHD Mental Health Detection - A Short Survey," 2022 25th International Conference on Information Fusion (FUSION), Linköping, Sweden, 2022
5. A. Gupta and D. Malhotra, "AI-Based Techniques for the Detection of Neuro-Developmental Disorders: A Survey," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023.
6. J. Berrezueta-Guzman, I. Pau, M. -L. Martín-Ruiz and N. Máximo-Bocanegra, "Assessment of a Robotic Assistant for Supporting Homework Activities of Children With ADHD," in IEEE Access, vol. 9, pp. 93450-93465, 2021,
7. Sanders, A.-L. (2018) Basic Phonemes and Lip-Synching for Animation.
8. Saygin, A. P., Cicekli, I. & Akman, V. (2000) Turing Test: 50 Years Later. Minds and Machines, 10(4), pp. 453-518.
9. Schuetzler, R. M., Giboney, J. S., Grimes, M. & Nunamaker, J. F. (2018) The influence of conversational agent embodiment and conversational relevance on socially desirable responding. Decision Support Systems, Volume 114, pp. 94-102.
10. Shore, J. (2007) The Art of Agile Development. 1st ed. Sebastopol: O'Reilly Media, Inc.. Slovák, P., Gilad-Bahrach, R. & Fitzpatrick, G. (2015) Designing Social and Emotional Skills Training: The Challenges and Opportunities for Technology Support, Seattle: Microsoft Research. Smith, G. et al. (2009) A Cognitive Training Program Based on Principles of Brain Plasticity: Results

from the Improvement in Memory with Plasticity-based Adaptive Cognitive Training (IMPACT) Study.

11. Sondergaard, H. M. et al. (2016) Treatment dropout and missed appointments among adults with attention-deficit/hyperactivity disorder: associations with patient- and disorder-related factors. *Journal of Clinical Psychiatry*, 77(2), pp. 232-239.
12. Tantam, D. (2003) The challenge of adolescents and adults with asperger syndrome. *Child Adolescence and Psychiatric Clinics of North America*, Volume 12, pp. 142-163. The Sandman Wiki (2018) Delirium. University of Cambridge (2018)
13. Weisenbaum, J. (1966) ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), pp. 36-45

APPENDIX

A. SOURCE CODE

Log Database:

```
# Set logConversation to 0 if you do not want to write conversations to file
```

```
logConversation = 1
```

```
if logConversation:
```

```
    # Get the path to the log file folder
```

```
    newpath = os.path.expanduser("~\Documents")
```

```
    newpath = newpath + "\MasumiChatLogs"
```

```
    # Create a folder for the log files in the Documents folder
```

```
    # if it does not already exist
```

```
    if not os.path.exists(newpath):
```

```
        os.makedirs(newpath)
```

```
    # Create the path to the logfile
```

```
    now = datetime.now()
```

```
    logfilename = newpath + now.strftime("\logfile_%d%m%Y_%H%M.txt")
```

```
    print(logfilename)
```

```
SCREENSIZE = (1000, 600)
```

```
WHITE = (245, 245, 245)
```

```
RED = (255, 155, 155)
```

```
LIGHTRED = (255, 215, 225)
```

```
BLACK = (0, 0, 0)
```

```
TEXTPOS = (260, 570)
```

```
def createLogfile(self):
```

```
    """ Create a logfile for the session. """
```

```
    with open(logfilename, "a") as myfile:
```

```

myfile.write("\n\n-----\n")
myfile.write("  Session started: ")
myfile.write(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
myfile.write("\n\n-----\n")

```

```

def writeButtonPressToLogFile(self, message):
    """ Writes the text of the button pressed to the logfile.

```

```

:param message: The button's caption text.

```

```

    """

```

```

    """ """

```

```

    with open(logfilename, "a") as myfile:
        myfile.write("\n  BUTTON PRESS: ")
        myfile.write(message)

```

```

def writeClosingTimeToLogFile(self):

```

```

    """ Writes the time that the application closed to the logfile. """

```

```

    with open(logfilename, "a") as myfile:

```

```

        myfile.write("\n\n-----\n")
        myfile.write("  Session closed: ")
        myfile.write(datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
        myfile.write("\n\n-----\n")

```

AIML Related functions:

```

class MasumiBrain:

```

```

    """ Used to handle the application's use of the PyAIML library.

```

```

    Attributes:

```

```

        k: A PyAIML kernel.
        voice: A Microsoft SAPI voice.

```

```

    """

```

```

    def __init__(self):

```

```

        """ Initialises MasumiBrain. """

```

```

        self.loadBrain()
def createNewBrain(self):
    """ Loads an AIML knowledge tree into the PyAIML kernel
        and saves it to a brain file (.brn) for faster loading
        the next time that the application is opened.
    """
    self.k = pyaiml.Kernel()
    self.k.bootstrap(learnFiles="alice-startup.xml")
    self.k.respond("load aiml b")
    # Set the predicates for the session
    self.k.setPredicate('name', 'Human')
    for a, b in sv.preds:
        self.k.setBotPredicate(a, b)
    self.k.saveBrain("scripts\\masumibrain.brn")

def loadBrain(self):
    """ If a brain file (.brn) exists loads it into the PyAIML kernel,
        otherwise loads an AIML knowledge tree into the PyAIML kernel
        and saves it to a brain file (.brn) for faster loading
        the next time that the application is opened.
    """
    """ Load or create the Bots brain. """
    # Check first if the brain file exists
    (exitstatus, outtext) = subprocess.getstatusoutput('dir scripts\\masumibrain.brn')
    if exitstatus == 0:
        # Load the knowledge tree into the PyAIML Kernel
        self.k = pyaiml.Kernel()
        self.k.bootstrap(brainFile="scripts\\masumibrain.brn")
        # Set the predicates for the session
        self.k.setPredicate('name', 'Human')
        for a, b in sv.preds:
            self.k.setBotPredicate(a, b)
    else:
        self.createNewBrain()

```

```
def saveBrain(self):
    """ Writes out the knowledge tree stored in the PyAIML kernel
        to a brain file (.brn).
    """
    self.k.saveBrain("scripts\\masumibrain.brn")
```

Bot Response:

```
def getBotResponse(self, inputText):
    """ Sends a text string to the PyAIML kernel for a response
        from the stored knowledge tree.
    :param inputText: A text string.
    :return: A tidied response to the text string.
    """
    """ Get response and tidy it up. """
    # Log the user's text
    if logConversation:
        with open(logfilename, "a") as myfile:
            myfile.write("\nUser: ")
            myfile.write(inputText)
    a = self.k.respond(inputText)
    # Tidy up the bot's response
    tidiedResponse = " ".join(a.split())
    tidiedResponse = re.sub(r'\s([?!."])(?:\s|$))', r'\1', tidiedResponse)
    tidiedResponse = tidiedResponse.capitalize()
    # Log the bot's response
    if logConversation:
        with open(logfilename, "a") as myfile:
            myfile.write("\nBot: ")
            myfile.write(tidiedResponse)
    # If there was no response, change the subject :D
    if tidiedResponse=="":
        tidiedResponse="Lets talk about something else, please."
    return tidiedResponse
```

TTS Related Functions

Masumi Voice:

class MasumiVoice:

""" Used to handle the application's Text-to-Speech functionality.

Attributes:

engine: A SAPI SpVoice Interface object.

voice: A Microsoft SAPI voice.

"""

def __init__(self, voice):

""" Initialises MasumiVoice with a Microsoft SAPI voice.

:param voice: A Microsoft SAPI voice description.

"""

self.voice = voice

self.engine = CreateObject("SAPI.SpVoice")

Text To Speech:

def convertTextToSpeech(self, textToSpeak):

""" Creates a temporary Text-to-Speech WAV of a text, plays it
then deletes it

:param textToSpeak: A line of text to convert to speech.

"""

if self.voice:

self.createWAV(textToSpeak)

try:

soundObj = pygame.mixer.Sound('temp.wav')

soundObj.play() # Continues code during sound playing

subprocess.getoutput('del temp.wav')

except: # If there is an error create an audio error message

self.createWAV("Program Error")

soundObj = pygame.mixer.Sound('temp.wav')

soundObj.play()

subprocess.getoutput('del temp.wav')

```

def createWAV(self, textToSpeak):
    """ Creates a Text-to-Speech WAV from a text string.
    :param textToSpeak: A line of text to convert to speech.
    """

    # If there is no response (for example, because of recursion depth exceeded
error)
    if len(textToSpeak) <= 1:
        textToSpeak = "Lets talk about something else, please."

    # Windows pronounces bot's name as Mahjewmi
    textToSpeak = textToSpeak.replace("masumi", "Massoomi")
    # Write text out to speech wav
    stream = CreateObject("SAPI.SpFileStream")
    outfile = "temp.wav"
    stream.Open(outfile, SpeechLib.SSFMCreatForWrite)
    self.engine.AudioOutputStream = stream

    # Add voice tags to select the TTS voice
    textToSpeak = sv.voiceTags + textToSpeak
    self.engine.speak(textToSpeak)
    stream.Close()

```

Buttons and Fonts:

```

def createButtons(self):
    """ Creates the user feedback buttons. """
    self.btn_1 = Button("Changed the Subject",
                        (760, 30, 220, 60), (780, 52),
                        'fonts/notepad.ttf', 18)
    self.btn_2 = Button("Did not understand",
                        (760, 110, 220, 60), (785, 132),
                        'fonts/notepad.ttf', 18)
    self.btn_3 = Button("Nonsense Answer",

```

```

        (760, 190, 220, 60), (797, 212),
        'fonts/notepad.ttf', 18)
self.btn_4 = Button("Forgot what I said",
        (760, 270, 220, 60), (795, 292),
        'fonts/notepad.ttf', 18)
self.btn_5 = Button("Repeated Answer",
        (760, 350, 220, 60), (800, 372),
        'fonts/notepad.ttf', 18)
self.btn_6 = Button("Program Error",
        (760, 430, 220, 60), (815, 452),
        'fonts/notepad.ttf', 18)
self.btn_exit = Button("EXIT",
        (760, 510, 220, 60), (835, 525),
        'fonts/notepad.ttf', 32)

def createFonts(self):
    """ Creates the fonts used to display text on-screen. """
    self.miniResponseFont = pygame.font.Font('fonts/notepad.ttf', 13)
    self.userInputFont = pygame.font.Font('fonts/notepad.ttf', 18)
    self.responseFont = pygame.font.Font('fonts/notepad.ttf', 22)
    self.logoFont = pygame.font.Font('fonts/Beyond_Wonderland.ttf', 150)

```

Avatar Animations:

```

def loadBaseMasumilImages(self):
    """ Loads the blink/default images for the chatbot's avatar animation. """
    self.blinkImages = []
    for img in range(17):
        filename = "img\\blink\\neko%s.png" % str(img + 1)
        self.blinkImages.append(pygame.image.load(filename).convert())

def loadHeadMovementImages(self):
    """ Loads the images to animate the chatbot's head nod/move animation. """
    n_1 = pygame.image.load('img/shake/neko0.png').convert_alpha()

```

```

n_2 = pygame.image.load('img/shake/neko2.png').convert_alpha()
self.nodImageList = [n_1, n_1, n_1, n_1, n_1, n_1, n_1, n_1, n_1,
                    self.blinkImages[0], self.blinkImages[0], self.blinkImages[0],
                    self.blinkImages[0], self.blinkImages[0],
                    n_2, n_2, n_2, n_2, n_2, n_2, n_2, n_2, n_2,
                    self.blinkImages[0], self.blinkImages[0], self.blinkImages[0],
                    self.blinkImages[0], self.blinkImages[0]]

def loadMannerism2Images(self):
    """ Loads the images for the Mannerism2 animation (a mouth movement
resembling
    a goldfish which is activated after a long period of user inactivity).
    """
    m2_1 = self.mouthImages[5]
    m2_2 = self.mouthImages[6]
    self.mannerism2ImageList = [m2_1, m2_1, m2_1, m2_1, m2_2, m2_2,
                                m2_2, m2_2, m2_1, m2_1, m2_1, m2_1, 0, 0, 0, 0, 0, 0,
                                m2_1, m2_1, m2_1, m2_1, m2_2, m2_2,
                                m2_2, m2_2, m2_1, m2_1, m2_1, m2_1, 0, 0, 0, 0, 0, 0]

def loadMouthShapeImages(self):
    """ Loads the images used for the chatbot's speech animation. """
    mouthshapes = ['img\\talk\\FV.png', 'img\\talk\\TS.png', 'img\\talk\\E.png',
                  'img\\talk\\LN.png', 'img\\talk\\A.png', 'img\\talk\\UQ.png',
                  'img\\talk\\O.png', 'img\\talk\\WR.png', 'img\\talk\\MBP.png']
    self.mouthImages = []
    for img in mouthshapes:
        self.mouthImages.append(pygame.image.load(img).convert_alpha()) #
sprite in png format

def renderBlinkAnimation(self):
    """ Determines which blink/default animation image to display
    and blits it to the PyGame surface.
    """

```

```

# Set blink to active if appropriate
if self.currentTime - self.startTime > self.blinkRate:
    self.blinkActive = True
# Set the appropriate blink image
if self.blinkActive:
    self.blinkIndex += 1
    if self.blinkIndex > 16:
        self.blinkIndex = 0
        self.blinkActive = False
        self.startTime = self.currentTime
# Only blink if head is not moving
if not self.nodActive:
    Masumi_IMG = self.blinkImages[self.blinkIndex]
    self.screen.blit(Masumi_IMG, (0, 160))

def renderHeadMoveAnimation(self):
    """ Determines which head nod/move animation image to display
    and blits it to the PyGame surface.
    """

    if not self.is_speaking and not self.blinkActive and self.currentMannerism == 1:
        self.nodIndex += 1
        if self.nodIndex > 27:
            self.nodIndex = 0
            self.nodActive = True
            self.screen.blit(self.nodImageList[self.nodIndex], (0, 160))
        else:
            self.nodActive = False
            self.nodIndex = 0

def renderLogo(self):
    """ Renders the logo text on-screen. """
    self.screen.blit(self.logoSurface, (190, 10))
def renderMannerism2Animation(self):

```

```
""" Determines which mannerism2 animation image to display
and blits it to the PyGame surface.
"""

if self.mannerism2Active and not self.is_speaking:
    m2IMG = self.mannerism2ImageList[self.mannerism2Index]
    if m2IMG != 0:
        self.screen.blit(m2IMG, (0, 160))
    self.mannerism2Index += 1
    if self.mannerism2Index > 35:
        self.mannerism2Index = 0
```

B. SCREENSHOTS



(A) First Response from the chatbot.



(B) Second Response from the chatbot.



(C) Third Response From The Chatbot



(D) Fourth Response From The Chatbot



(E) Fifth Response From The Chatbot



(F) Sixth Response From The Chatbot

C. RESEARCH PAPER

An AI Chatbot to enhance the day-to-day life and livelihood of Patients affected by ADHD (Hyper-Activity Disorder)

Joshua Ronaldino P
Department of Computer Science and
Engineering(CSE),
Sathyabama Institute of Science and
Technology,
Chennai, India
bluemoonjoshronald@gmail.com

Christen Jones S
Department of Computer Science and
Engineering(CSE),
Sathyabama Institute of Science and
Technology,
Chennai, India
s.christenjones@gmail.com

Dr. P. Ajitha M.E., Ph.D,
Department of Computer Science and
Engineering(CSE),
Sathyabama Institute of Science and
Technology,
Chennai, India
ajitha.it@sathyabama.ac.in

ABSTRACT

One of the most common behavioural diseases in the world is Attention Deficit Hyperactivity Disorder (ADHD). Research indicates that 3% of adults and 5% of children are affected (Polanczyk, 2007). (Fayyad and others, 2007)

ADHD symptoms include impulsivity, anxiety, hyperactivity, and trouble focusing. ADHD sufferers may experience significant disruptions in their interpersonal interactions, which may impact their academic or professional success.

Since there is no universal agreement on the origin of ADHD, current therapies concentrate on symptom reduction. Typically, patients get both medication and cognitive behavioural therapy (CBT) as part of their treatment.

In CBT sessions, patients work through their issues with a qualified therapist in order to develop coping skills. Frequently, these sessions incorporate role-playing exercises to teach patients how to perceive. A type of treatment called computerized cognitive behavioural therapy (cCBT) uses technology to provide CBT. These software-based remedies, which are often intended to improve mental concentration, can supplement conventional therapeutic approaches.

Computer programs called chatbots are made to mimic a conversation partner. When users enter queries or assertions, the software interprets and returns the most relevant answer.

Since there aren't any solutions on the market that let people with ADHD practice and improve their concentration during role-playing discussions, a chatbot-based cCBT solution may be created to fill this need.

I. INTRODUCTION

According to the proposal, the goal of this project is to develop an application that can be utilized in CBT settings to raise the attention levels of people with ADHD.

When using the program, users will concentrate on how an Artificial Conversational Agent (chatbot) speaks with them. They will also be able to identify conversational errors made by the chatbot, such as it wandering off the subject, by pressing a button.

Users will become more attentive and learn to refrain from employing incorrect comments and behaviours during conversations if they concentrate on identifying the chatbot's inappropriate responses.

II. RELATED WORKS

A. Chatbot System for Healthcare

Artificial intelligence was not an established field prior to now. The introduction of chatbots has expedited the resolution of user complaints. The healthcare industry is one that is quickly embracing automated chatbots for internet applications on a global scale. Hospitals treat a wide range of ailments by admitting patients. Medics aren't always available since nursing takes a lot of time.

To tackle this issue, chatbots for medicine were developed. Since the accuracy of the output is crucial, these chatbots are evaluated and trained using actual datasets. AI-powered chatbots are precise, trustworthy, and fast. The user responds to the relevant inquiries and gets feedback in light of their responses. Should a user make a small mistake.

B. Processing Natural Language

An area of artificial intelligence called natural language processing, or NLP, enables robots to learn, read, and comprehend human languages. Chat apps are used by a variety of industries, including banking, education, finance, and so on, to promote their goods and find

solutions to issues. The majority of nations, including China, India, Korea, Germany, Spain, and others, have their own original language. Language barriers arise when users engage with one another. A range of languages are provided by NLP-based autonomous systems to lessen this kind of complexity and facilitate verbal and nonverbal communication.

NLP is utilized in the healthcare chatbot to parse text. The following NLP features are beneficial in the medical field:

1. Sentence tokenization: Using this technique, the entire sentence is broken up into smaller words and substrings. When a patient submits a question in the form of sentences, punctuation marks separate the sentences and turn a whole word into a token.

2. Word Tokenization:

Dictionaries contain a wide range of terms. The process of breaking up sentences is done. Tokens are allocated words. The chatbot provides the user with the best possible feedback while assisting with the classification of words based on the categories found in medical records.

3. Lemmatization and stemming: Stemming is the cutting off of words at their ends. Conversely, lemmatization is used in morphological analysis to extract words. The responses provided by the chatbot become meaningless if doctors enter inaccurate data in their database. The above-described techniques are utilized to correct data and give a patient a suitable text response.

III. LITERATURE REVIEW

Implementing the functional system and comprehending the topic of artificial intelligence both gain a great deal from the literature review. There are three articles that provide a literature overview along with explanations. In this study, we have discussed the use of healthcare chatbots by cancer patients. This essay explains the process of early cancer identification. The prognosis and survival of cancer patients are enhanced by early identification of the disease. Most individuals find their cancer when it has progressed to an advanced stage. The disease known as cancer is caused by abnormal cells that continue to divide and proliferate. Cancer patients lose up on having healthier, longer lives. In the medical sector, depression is fast becoming one of the most difficult stages. For this study, we performed a survey on medical self-diagnosis. This essay provides helpful advice and facilitates comprehension of contemporary health-related issues. In this instance, customer questions on mobile applications are handled by the Chabot. Artificial intelligence is the main technique used to create those intelligent bots. The goal of this branch of computer science is to build intelligent machines that have human-like behaviour and thought processes. Overweight people are far more likely to suffer from dangerous diseases and conditions. Not only are obesity rates rising in developed nations but also in developing nations. The cancer dataset could not be

utilized by the learning model. We also included medical support with the use of natural language processing and mobile development technology. The healthcare chatbot used and assessed many Android cell phones. Mobile conversations are becoming a more and more prevalent kind of communication. These conversations are special because they are really simple and effective in terms of communication. Therefore, a chatbot may be quite effective if it follows all of the rules of an instant messaging program. Chabots are often text-driven, with unified widgets, images, and user-friendly interfaces that encourage dialogue. Furthermore, the most successful businesses were able to win their customers' confidence via clarity. You may read more about these two in-depth in some of our previous blog posts. It may be tailored to speak to people the way you do, convey your brand identity and personality, and change its message in reaction to user input. These days, when it comes to the complexity and interaction of medicine, medical description websites are highly inventive and captivating. This study's and this article's objective is to discuss the problems that people with attention-deficit/hyperactivity disorder, or ADHD, are now experiencing. Instead of predicting the problems that a patient may face on a daily basis, the purpose of this research is to find a solution or make life easier for people who are affected by them.

1. LITERATURE SURVEY

1. Emmanuel Papadakis, University of Huddersfield, Huddersfield, UK, 2023, "Neuro Intel: A System for Clinical Diagnosis of Attention Deficit Hyperactivity Disorder (ADHD) Using Artificial Intelligence"

- In this study, an Artificial Intelligence (AI) system named Neuro Intel is used to demonstrate a completely functional method for diagnosing ADHD. Because of the promising outcomes of our study, Neuro Intel was created to help doctors diagnose adult ADHD. It uses machine learning to combine the knowledge of experienced clinicians with historical clinical data.

2. "Machine Learning and ADHD Mental Health Detection - A Short Survey," - [Writer: Christian Nash, Newcastle University, UK, 2023] Intelligent Sensing and Communications Research Group

- This work investigates the use of machine learning to several sensing methods, including electroencephalography (EEG) and functional magnetic resonance imaging (fMRI). Additionally, we investigate other methods for identifying ADHD, including computer-based exercises, health surveys, and medical records. The increasing awareness of mental health issues necessitates a review of the literature on ADHD in order to develop a trustworthy Artificial Intelligence (AI) based on machine learning. This can help medical professionals diagnose ADHD more quickly.

3. "AI-Based Techniques for the Detection of Neuro-Developmental Disorders: A Survey," - [Written in 2023 by A. Gupta and D. Malhotra]

- This study presents a number of automated diagnostic tools for the early diagnosis of ASD, ADHD, and CDD that make use of artificial intelligence techniques based on quantitative criteria. In order to give a more useful NDD analysis, the study's conclusion highlights open issues in the literature that require additional research.

4. "Role of Natural Language Processing in Improving Lives of Disabled," - Emily R. Phillips, author (2022)

- The function of natural language processing in the lives of resolute individuals is discussed in this essay. It presents a paradigm that may be just what those who are deaf or hard of seeing need to stay up with others who are able-bodied. It also discusses the detrimental effects of NLP on obstinate individuals and socially biased persons. Potential implementation errors have been fixed in addition to further author effort. To advance and be implemented, a system has to comprehend natural language, which is a challenging challenge because of natural language's numerous intricacies.

5. "Guided crossword-puzzle games aimed at children with Attentional Deficit: Preliminary results,"-[Writer: Anderson, Robert J. 2020]

- Based on the usability questionnaire, the final calculations yielded an average score of 87.11 out of 100. In addition to creating new guided mini-games for focusing on various areas, the current study should be replicated in order to better understand the demands of these users and improve behavioural skills like organization and attention in kids with and without ADHD.

These studies collectively provide a comprehensive view of the current landscape of AI-based ADHD assistance systems. They address a range of topics, including technical aspects, ethical considerations, legal implications, and the impact on students and educational institutions. These insights can inform the development and implementation of AI assistance systems, helping strike a balance between academic integrity and student privacy in the digital learning environment.

2. INFERENCE

The majority of machine learning publications on classifying ADHD patients involve datasets from both ADHD patients and healthy controls; however, research comparing data from other diseases is scarce. Many reports have indicated that comorbidities exist with ADHD. For example, there is more to epilepsy and ADHD than just comorbidity because of their long-standing correlation.

Paroxysmal activity in the EEG is now frequently detected in 15-20% of children who receive a diagnosis. More investigation is required for differential diagnosis in order to distinguish between the many illnesses or impairments associated with ADHD in addition to identifying anomalies.

However, the classification of ADHD as a neurodevelopmental condition suggests a delay in brain maturation, a partial explanation for the disorder's decline in adulthood, or alterations in the development seen in people who were initially diagnosed with the disorder as children. The progression of these alterations from ADHD kid to adult requires more investigation. In Clarke's study, individuals who had been diagnosed with ADHD as children as adults showed higher alpha and beta frequencies; also, their heightened global theta activity during childhood normalized. While no other studies have examined feature changes over an extended lifespan, this shows that EEG properties corresponding to normal brain growth should be incorporated into any decision-support system that diagnoses ADHD.

3. OPEN PROBLEMS IN EXISTING SYSTEM

There aren't many AI-based tools and systems available as of September that are meant to help people with ADHD (Attention-Deficit/Hyperactivity Disorder) and those who care for them. These systems were primarily designed to enhance several facets of everyday living, such as productivity, organization, and symptom management. Note that since then, there could have been more advancements in this sector. AI-Powered Applications for Organization and Time Management, Voice assistants, voice-activated cognitive behavioural therapy (CBT) apps, digital note-taking and organization tools, and virtual coaches. Our proposal for this project is to develop a system that can analyze and evaluate the behavioural science underlying an individual's behaviour. To design and train the software based on user convenience, we will leverage data systems from research in behavioural science. Several different Python libraries will be used.

1. Quantity and Quality of Data: Limited Information
Effective AI model training is hampered by the amount, diversity, and quality constraints of datasets pertaining to ADHD.

Labelling Difficulties: It can be difficult to adequately classify subjective features, and there may not be as much evidence of ADHD symptoms and behaviours that have been properly labelled.

2. Customization and Variability: ADHD Heterogeneity:
The intensity of symptoms varies widely between people. One of the biggest challenges is developing tailored AI systems that can handle this variety.

Adaptability: AI systems must be able to adjust over time to a person's evolving symptoms and behaviours, which calls for ongoing learning processes.

3. Interpretability and Trust: Black Box Problem: A lot of AI models are complicated and difficult to read, which makes it challenging for users and doctors to have faith in and comprehension of the choices these systems make.

Explainability: In medical settings, where comprehending the rationale behind a diagnosis or suggestion is critical, interpretable AI models are necessary.

4. Privacy and Ethical Issues:

Privacy: Data pertaining to ADHD is delicate and needs strong privacy safeguards. A major worry is making sure that personal data is handled securely.

Fairness and Bias: Data-based biases can be inherited by AI algorithms. It is a continuous struggle to ensure fairness and minimize biases in the diagnostic and treatment recommendations.

5. Clinical Practice Integration:

Clinical Validation: In order for AI-based ADHD systems to be widely accepted and used, it is imperative that their efficacy be verified in actual clinical settings.

Collaboration: To guarantee that the technology is in line with clinical workflows and practices, effective collaboration between AI researchers and healthcare practitioners must be established.

6. Extended Observation and Involvement:

Durability: Long-term user involvement might be difficult to maintain. One persistent challenge in AI design is maintaining user motivation to utilize the platform regularly.

Long-Term Monitoring: Tracking the development and results of AI treatments over time to evaluate their adaptability and long-term efficacy.

IV. REQUIREMENT ANALYSIS

1. FEASIBILITY STUDIES

1. Technical Feasibility: Assess if the project's technology can be applied successfully and is widely available. This entails determining if the necessary gear and software are available as well as how well the AI proctoring software works with the current setup.

2. Operational Feasibility: Assess the project's ability to be included in educational institutions' regular operations. Take into account elements such as personnel competence in system operation and support, maintenance, and deployment ease.

3. Economic Feasibility: Examine how cost-effective the proposal is. Estimating the system's initial setup costs, continuing maintenance costs, and possible savings or extra income from its implementation are all included in this. It's critical to determine whether the advantages exceed the disadvantages.

Assess the project's compliance with legal and regulatory requirements, including privacy legislation, data protection laws, and educational standards, to determine its legal and regulatory feasibility. Determine whether there are any legal obstacles and evaluate whether they can be overcome.

2. SOFTWARE REQUIREMENTS

Four main components make up the application:

- The Python programming language.
- The multimedia library of PyGame.
- Artificial Markup Language for Intelligence.
- The PyAIML module.

This section examines these components and provides an application development process overview.

Python

Python is a popular high-level language used for interpretation in the scientific community. Part of the reason for this popularity is the large number of free packages related to machine learning and data analysis that can be obtained from the Python Package Index repository. Well-written Python code is concise, easy to understand, and maintain. It gives programmers the ability to leverage the Object-Oriented Programming (OOP) paradigm to design interactive user interfaces. The Masumi program must run numerous processes simultaneously in order to handle events like mouse and key presses, sound, animations, and communication with the chatbot's kernel. OOP makes it easy to keep track of the states of many objects.

PyGame

Despite being slower to execute than compiled languages like C or C++, Python may be easily extended through the use of compiled extensions written in other programming languages. One such add-on that lets programmers include graphics and audio functions in their apps is the PyGame library.

PyGame enables visual programs to invoke optimized C and Assembly code that executes more than 100 times quicker than pure Python, resulting in rapid screen updates. All popular operating systems and graphic renderers, such as DirectX, X11, and OpenGL, are compatible with PyGame. It has functions for playing

sound effects and detecting object collisions in addition to managing images.

AIML

Chatbots use natural language processing to generate and deliver replies. A software named ELIZA was among the first to imitate a fictitious Rogerian psychotherapist. (Weizenbaum, 1966) Tokenizing user input and calculating the best response, ELIZA leverages the distribution of keywords. The program lacks context awareness and real-world knowledge, which makes it incapable of answering questions or remembering previous commands. Artificial Intelligence Markup Language (AIML), an extensible markup language (XML) based on the ELIZA concept, allows programs to input knowledge trees. Wallace (in 2003). Please see Appendix K. AIML can store an endless amount of question responses owing to these knowledge trees, and programs can monitor user comments thanks to the specification.

PyAIML

The stimulus-response model is the foundation of AIML. A response is sent after the user's input (stimulus) matches a pattern kept in a knowledge tree. (Mikic and others, 2008) An interface between AIML files and the programs that use them is provided by AIML interpreters. Although one may build an AIML interpreter from the ground up, there are a lot of open-source interpreters available for languages like Java, C++, Python, and PHP. The extra features that may have been included in a custom implementation were lost while using a third-party interpreter to shorten the development time. The PyAIML interpreter was selected as the Masumi application utilizes Python. An open-source Python interpreter for AIML is called PyAIML.

V. PROPOSED SYSTEM

1. Architecture Diagram

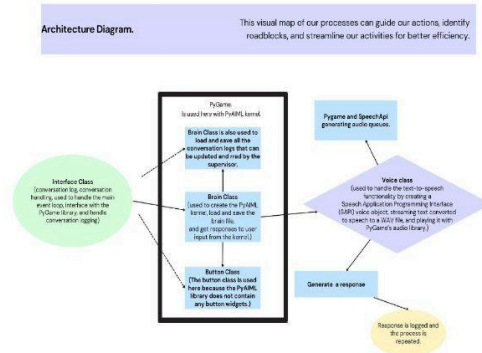


Figure 1: System Architecture for AI-BASED ADHD ASSISTANCE SYSTEM

2. DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

An object-oriented methodology was to be used in development. The paradigm of object-oriented programming is based on how real-world things communicate with one another. Every object is a self-contained entity composed of methods that enable it to carry out actions and data that define its properties. Objects in an object-oriented program have the option of interacting with one another or staying silent from one another. It is easier to include things when their inner workings are hidden. Additionally, a stub may be created during development to mimic an object's functionality while work is being done on other classes. A team may design an application more quickly by dividing up its components using this modular method.

3. MODULES

- **The MasumiInterface class**, which manages conversation logging, interfaces with the PyGame library, and controls the main event loop.
- **The MasumiBrain class**, which loads and saves the brain file, creates the PyAIML kernel and receives user input replies from the kernel.
- **The Speech Application Programming Interface (SAPI) voice object** is created by the MasumiVoice class, which then uses PyGame's audio library to play the voice-to-text text that has been converted to a WAV file.
- **The Button class** must be built since PyGame lacks a button widget class of its own. As methods and instance

variables were added to the classes, reference tables were generated.

Data Collection and Storage Module:

Gather and preserve medical information from a variety of sources, including research articles, medical imaging, patient histories, and electronic health records (EHRs).

Data Preprocessing Module:

Clean, normalize, and standardize the collected data to remove noise and inconsistencies.

Natural Language Processing (NLP) Module:

Process and analyze textual medical data including clinical notes, research articles, and patient records.

User Interface (UI) and User Experience (UX) Module:

Provide patient and healthcare provider-friendly interfaces so they may easily engage with the system. Make sure a variety of user groups may easily access and utilize the system.

Feedback and Improvement Module:

Collect feedback from users and continuously improve the system's performance and usability.

4. PROJECT MANAGEMENT PLAN

The implementation of management strategies is necessary for development efforts to be finished on schedule. When developing the application, both the Agile and Waterfall project management methodologies were taken into account. The Waterfall Method keeps projects moving ahead in a precise sequence of pre-planned phases by limiting team members to returning to a previous step only when absolutely necessary. Milestones are where each step is supposed to start and end. The agile technique is more adaptable than the Waterfall approach. Iteration is a tool for accelerating the development of useful apps. Projects are broken down into smaller tasks, which are often finished in intense sprints of two to four weeks.

The Waterfall technique was chosen for this project for a variety of reasons. The project is being worked on by a single developer, therefore there was no need to coordinate the efforts of team members working on different modules. Furthermore, it was improbable that project requirements would change as it was being created. Third-party libraries included in the program are trustworthy. For over a decade, the open-source software community has rigorously tested each one of them, and they are all compatible with the latest version of Windows 10. Even if the program that was developed may have been more client-focused had the Agile methodology been employed, the decision to adopt the

structured Waterfall process helped finish the desired application on time.

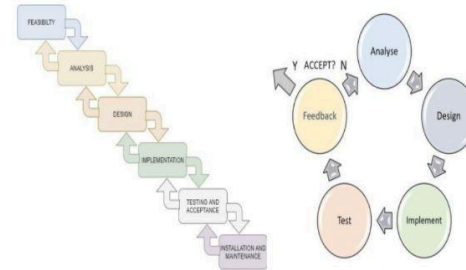


Figure 2: Comparison between the Waterfall method and the Agile method.

VI. CONCLUSIONS

The first proposal in this study was that users may improve their attentiveness by concentrating their attention on identifying improper replies from chatbots. Although the findings of user testing partially support this theory, they also emphasize that further testing is necessary before the program is ready for public release. The project's main conclusions were: Participants' attention levels rose throughout user testing.

Throughout the user testing process, participants remained engaged and felt at ease while interacting with the chatbot.

Participants may lose focus on the exercise's goal if they are not supervised by a licensed therapist.

Using third-party libraries or native TTS libraries, several versions might be made for every platform. An alternative would be to utilize a wrapper library like pyttsx3, which offers cross-platform TTS access across several platforms. After a few small tweaks, the present version of the program will serve its function, but before a commercially viable version can be created, comprehensive user testing incorporating feedback from therapists and diagnosed ADHD patients is still necessary.

Raised levels of attentiveness:

Chat Logs gathered during user testing revealed that the application increased users' attention spans (see Appendix S). For example, the subject in Extract 1 is fully focused on the task at hand. Immediately recognizing the chatbot's first inability to understand

what was said, he or she clicks the appropriate button to let others know. Later in extract 1, the participant finds out that the chatbot is giving a generic response to a certain type of inquiry. After this observation is confirmed, the relevant button is pressed to indicate it. It was possible for the person to stay focused on the purpose of their chatbot interaction.

Response towards the avatar:

Participants wanted to use the program for considerably longer even though they were told to speak with the chatbot for five minutes. The test participants were engaged by the chatbot's personality and interface design. There were no lengthy pauses when participants looked up the right words to employ during chats with the chatbot. It was possible for testers to freely challenge what was being stated because they were aware that their conversation partner was a computer program. For instance, in the second snippet, the participant quickly analyzes the logic of the chatbot until the software erroneously changes the topic.

Choosing an engaging avatar design was crucial in maintaining users' attention and engagement with the service.

The developer's claim that utilizing the program might assist ADHD sufferers become more attentive is supported by the results of these first testing. BSCTUC-6PD: Masumi, an Artificial Conversational Agent designed to help ADHD patients become more attentive. Excerpts from log files also demonstrate how the application's use can highlight facets of a user's personality that would not come through in the formal setting of a CBT therapy session.

The software was judged to be appropriate for its intended use. It may now be included into a treatment strategy that gives patients a secure setting in which to assess and improve their interpersonal interactions.

VII. RESULTS

The creation and testing of a novel strategy for improving attention in ADHD patients is described in this research. According to test results, using the program promotes attentiveness, and it may be applied in CBT circumstances where a licensed therapist is present. Masumi is intended to enhance conventional types of CBT, not to replace them. It helps patients improve their conversational attention at home and is meant to be used outside of a professional treatment setting. The mobility of the program is one of its primary advantages for users. Instead of relying on other people's schedules, the patient may utilize it whenever they are ready, day or night.

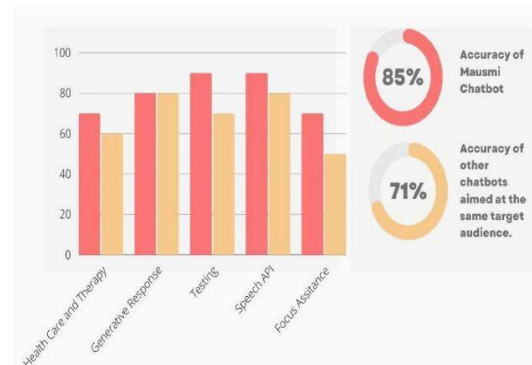


Figure 3: Comparison Graph: Masumi Chatbot to Other Chatbots

VIII. REFERENCES

- J. Berrezueta-Guzman, I. Pau, M. -L. Martín-Ruiz and N. Máximo-Bocanegra, "Assessment of a Robotic Assistant for Supporting Homework Activities of Children With ADHD," in IEEE Access.
- C. Nash, R. Nair and S. M. Naqvi, "Machine Learning in ADHD and Depression Mental Health Diagnosis: A Survey," in IEEE Access.
- S. Batsakis, E. Papadakis, I. Tachmazidis, T. Chen, G. Antoniou and M. Adamou, "Neuro Intel: A System for Clinical Diagnosis of Attention Deficit Hyperactivity Disorder (ADHD) Using Artificial Intelligence," 2023 IEEE Symposium on Computers and Communications (ISCC).
- C. Nash, R. Nair and S. M. Naqvi, "Machine Learning and ADHD Mental Health Detection - A Short Survey," 2022 25th International Conference on Information Fusion (FUSION), Linköping, Sweden, 2022
- A. Gupta and D. Malhotra, "AI-Based Techniques for the Detection of Neuro-Developmental Disorders: A Survey," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023.

CERTIFICATE

