

AI Therapist Using Natural Language Processing

Submitted in partial fulfilment of the requirements for the award of master of
Engineering / Technology degree in computer science and engineering

by

S.SRI RAMYA

(REG.NO.40310007)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING**

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

Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119

MAY-2022



SATHYABAMA

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

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 S. SRI RAMYA (Reg.no.40310007) who carried out the project entitled "**AI Therapist using Natural Language Processing**" under my / our supervision.
from Nov-2021 to May-2022.

Internal Guide

(Dr. G. Mathivanan B.Sc., B.Tech., M.E., PhD.,)

Head of the Department (Name in Capital letters with Seal & Signature)

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **S.SRIRAMYA** hereby declare that the Project Report entitled
“**AI Therapist Using Natural Language Processing**” done by me under the
guidance of Dr. G. Mathivanan B.Sc., B.Tech., M.E., PhD.), at Sathyabama Institute of
Science and Technology, is submitted in partial fulfilment of the requirements for the
award of **Master of Engineering in Computer Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I thank the Board of Management of SATHYABAMA for their kind encouragement in doing this project and 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.S.Vigneshwari M.E., Ph.D. and Dr.L.Lakshmanan M.E., Ph.D. , Heads** of the Department of Computer Science and Engineering for providing me the necessary support and guidelines at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide for his valuable guidance, (**Dr. G. Mathivanan B.Sc., B.Tech., M.E., PhD.,**) suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all teachers, technical staff and my fellow students of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Today's youth face a lot of stress in their life as well as their work. Various NLP techniques are employed in order to handle the stress of young people. Artificial Intelligence (AI) systems are developed in order to mimic human brain. A smart stress management system has been developed in order to mimic the role played by the counsellors. In the older systems of stress management, stress levels are detected and the appropriate solutions are provided. They provide user – friendly solutions but they are more time consuming and involves a lot of human power. So, a more intelligent stress management system has been developed which provides better results at a less time and with less human effort. It is a more robust system when compared to the previous one and it provides smarter solutions at a less cost.

TABLE OF CONTENT

S. NO	TITLE	PAGE NO
	ABSTRACT	
I.	INTRODUCTION.....	1
II.	LITERATURE SURVEY.....	3
III.	AIM.....	5
	a. OBJECTIVES.....	5
	b. SCOPE OF THE PRESENT INVESTIGATION.....	5
	c. EXISTING SYSTEM.....	6
	d. PROPOSED SYSTEM.....	6
IV.	EXPERIMENTAL OR MATERIALS AND METHODS.....	7
	a. PYTHON OVERVIEW.....	7
	b. PYTHON ENVIRONMENT.....	10
	c. METHODOLOGY.....	20
V.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS.....	30
VI.	SUMMARY AND CONCLUSIONS	31
VII.	REFERENCE	
VIII.	APPENDIX OF CODE	

LIST OF FIGURES

Fig 1 : System Architecture.....	22
Fig 2 : Dataflow Diagram.....	25
Fig 3 : Usecase Diagram.....	26
Fig 4 : Class Diagram.....	27
Fig 5 : Sequence Diagram.....	28
Fig 6 : Activity Diagram.....	29
Fig 7 : Collaboration Diagram.....	30

CHAPTER – 1

INTRODUCTION

Everybody experiences stress from time to time. Stress is what happens for your body when it feels threatened. In the simplest sense, it prepares the body to meet a threat by either fighting or fleeing. It is characterized by a racing heart, increased breathing rate, and heightened senses. All of these changes would be very important to an individual who was getting ready to face a life-threatening event. However, now that most of us live in a society where we rarely face truly life-threatening events, we don't necessarily need or want to have these reactions. In fact, most people's experience far too much stress in their everyday lives. Too much stress can tax the body eventually resulting in a compromised immune system, and other issues. Feeling sick, stuck, overwhelmed, depressed, anxious, or panicked are all reactions to stress that many people face [5]. These are not helpful emotions when what is actually required (and helpful) is the ability to think clearly and rationally. Many people spend much more time feeling worried, guilty, or ashamed for not doing things that they feel they ought to do or should do. The energy used to feed these negative emotions would be better spent focusing on completing your goals. To provide quick and cheaper solution to these problems, an application of artificial intelligence (AI) i.e. Machine learning can be applied, that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [2]. It is a study of algorithms and statistical models that computer systems use to progressively improve their performance on a specific task. It focuses on the development of computer programs that can access data and use it to learn for themselves. It can also be used in mental health management. By using natural language processing, it is easy to process the user's problem and provide appropriate solution. According to conservative estimates in medical books, 50 to 80% of all physical diseases are caused by stress. Stress is believed to be the principal cause in cardiovascular diseases. Stress can place one at higher risk for diabetes, ulcers, asthma, migraine headaches, skin disorders, epilepsy, and sexual dysfunction. Each of these diseases,

and host of others, is psychosomatic (i.e. either caused or exaggerated by mental conditions such as stress) in nature. Stress has three prong effects:

- Subjective effects of stress include feelings of guilt, shame, anxiety, aggression or frustration. Individuals also feel tired, tense, nervous, irritable, moody, or lonely [5].
- Visible changes in a person's behaviour are represented by behavioural effects of stress. Effects of behavioural stress is seen such as increased accidents, use of drugs or alcohol, laughter out of context, outlandish or argumentative behaviour, very excitable moods, and/or eating or drinking to excess.
- Diminishing mental ability, impaired judgment, rash decisions, forgetfulness and/or hypersensitivity to criticism are some effects of Cognitive stress.

CHAPTER – 2

II. LITERATURE SURVEY

Mental health has become exceedingly important to maintain a healthy lifestyle and a positive outlook. The awareness regarding mental health issues has gained prevalence in recent times. The system described in this paper talks about the classification performance of various classifiers such as Random Forests (RF), Support Vector Machines (SVM), Linear Discriminant Analysis (LDA), and k-Nearest neighbour. LDA was considered the most efficient in terms of average generalization errors. One major drawback of LDA is that it only performs well on a smaller number of correlated features.

[2] This paper monitors the Twitter feed of 135 Mechanical Turk (MTurk) participants and was compared with validated reports of suicide risk using a machine learning algorithm. The algorithm was able to identify the suicidal cases with 92% accuracy. The algorithm falls short in terms of identifying linguistic characteristics which is extremely vital in differentiating between various mental health-related causes.

[3] A system that conducts emotion detection on Twitter feeds using natural language processing has been described in this paper. The tweets are classified as negative or positive using classifiers such as Support Vector Machine and Naïve Bayes. Multinomial Naïve Bayes model outperforms the Support Vector Machine model. A drawback of this system is the low accuracy caused by the use of short forms.

[4] This paper talks about a system that processes Facebook data based on emotional process, temporal process, linguistic style, and all the aforementioned features together using techniques such as decision tree, k-Nearest Neighbours, Support Vector Machine, and ensemble. The Support Vector Machine model has

better performance. However, the decision tree has better results for precision, recall, and f1 score.

[5] The system described in this paper talks about a system to classify 10 subreddit COVID-19–related comments using Natural Language Processing. Linear Discriminant Analysis was used for topic modelling and LSTM was used for sentiment classification of the text. The scope was limited as it was restricted to the English language only.

[6] In this paper, a system that makes use of a neural network for online character recognition for recognizing a physician’s handwriting. Natural Language Processing is used for formatting and classifying the prescription. This system outperforms an existing solution called MedEx.

[7] The “ngrams” feature was used in this paper, meaning that predictions were based upon a contiguous sequence of n words that were codified and used as an input to the machine learning algorithm. The system was able to predict the probabilities of suicidal ideation or heightened psychiatric symptoms of the patients. However, while deciding the size of the n-grams, there was a trade-off between higher specificity and lower sensitivity.

[8] In this paper, a system that classifies the symptom keyword as ‘negative symptoms’ or ‘catatonic syndrome’. Text Hunter, an information extraction suite, was used for the same. A major drawback of the system was that it was not comprehensive and missed aspects of psychosis presentation.

[9] This paper aimed to use natural language processing to perform prediction of psychiatric readmission to promote interventions for this purpose. The natural language processing model used here is a Latent Dirichlet Allocation (LDA) model. This model is used to determine the most relevant and frequent words on specific subjects presented in a text document. The topics considered here for the LDA model were depression, anxiety, trauma, etc. TF IDF (Term frequency-inverse document frequency) was also calculated to recognize the most frequently appearing words and limit their lexicon. SVM was further used for classification and, specificity and sensitivity were used as performance metrics.

CHAPTER – 3

III. AIM

3.1 OBJECTIVE

- To overcome the gap between mental health management system for users and cost of therapies required to diagnose these mental health problems.
- To build an application that can provide gateway to user to open with their problems.
- To provide costless therapy to user to overcome their problems.
- To reduce the time required to process the user's problems.

3.2 SCOPE OF THE PRESENT INVESTIGATION

Stress, fear, anger, anxiety, depression are the terms that are recognized to be the only cause due to which a person suffers from several physiological disorders. It has a major impact on our body resistance which unknowingly leads to severe problems. Despite its impact on our day-to-day life, it is still infeasible for a physician to monitor with the stress levels and make the user aware of the same throughout the day. As the gap between the availability of mental health professional and the cost of each therapy session keeps increasing, the demand for digitized healthcare solutions has increased steadfastly. Generally, people write to communicate with others.

In addition to describing simple factual information, people also used to express their activities, and convey their feelings, mental states, hopes, and desires. This written information can be used to recognize the feeling of person that whether he is feeling stressful, angry, depressed or anxious etc. and at the same time help them too. This project overcomes all these problems by predicting exact feelings of user using natural language processing and provides respective solution like if user feels angry then recommender system will suggest him tips for anger management etc. [7] It saves user's time needed to visit the therapist by giving treatment within a second as well as saves money needed for therapy sessions.

3.3 EXISTING SYSTEM:

In the conventional systems, the user's emotions are detected by using various stress management systems and the appropriate solutions are provided. Artificial Intelligence methods uses various statistical models in order to analyze the stress levels and provide smart solutions to the users. Human emotions like anger, happiness and sadness are identified and suitable solutions are provided. AI models are designed in order to identify various stress levels and provide recommendations to the concerned users. But, all this requires a large amount of human effort and time and it is not suitable in all situations.

Disadvantages:

- Less robust when compared to the proposed systems.
- Involves a lot of human effort and time.

3.4 PROPOSED SYSTEM:

In order to overcome the shortcomings of the existing system, a smarter system has been developed where a AI system has been designed in order to mimic the human brain and provide timely recommendations to the concerned users. The

users state or stress levels are captured with the help of a Chabot and then they are analyzed using various AI systems in order to analyze the human state and provide correct solutions. The human emotions are captured, they are analyzed using AI techniques and they provide the appropriate counselling solutions.

Advantage:

- Improving the Efficiency to classify the emotion.
- Implement content based filtering .

CHAPTER - 4

4.1 PYTHON OVERVIEW:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- ☐ **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- ☐ **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- ☐ **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features:

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- ❑ **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- ❑ **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- ❑ **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- ❑ **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- ❑ **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- ❑ **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- ❑ **Databases:** Python provides interfaces to all major commercial databases.
- ❑ **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- ❑ **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- ❑ IT supports functional and structured programming methods as well as OOP.
- ❑ It can be used as a scripting language or can be compiled to byte-code for building large applications.
- ❑ It provides very high-level dynamic data types and supports dynamic type checking.
- ❑ IT supports automatic garbage collection.
- ❑ It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

4.2 PYTHON ENVIRONMENT:

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

SOFTWARE REQUIREMENTS

- Python
- Anaconda Navigator
- Python built-in modules
- Numpy
- Pandas

- Matplotlib
- Sklearn
- Seaborn

PANDAS

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data munging/wrangling if not THE most used one. Pandas is an open source

What's cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

Installation and Getting Started

In order to “get” Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like NumPy) and has optional dependancies (like Matplotlib for plotting). Therefore, I think that the easiest way to get Pandas set up is to install it through a package like the Anaconda distribution , “a cross platform distribution for data analysis and scientific computing.”

In order to use Pandas in your Python IDE (Integrated Development Environment) like Jupyter Notebook or Spyder (both of them come with Anaconda by default), you need to import the Pandas library first. Importing a library means loading it into the memory and then it's there for you to work with. In order to import Pandas all you have to do is run the following code:

- **import pandas as pd**
- **import numpy as np**

Usually you would add the second part ('as pd') so you can access Pandas with 'pd.command' instead of needing to write 'pandas.command' every time you need to use it. Also, you would import numpy as well, because it is very useful library for scientific computing with Python. Now Pandas is ready for use! Remember, you would need to do it every time you start a new Jupyter Notebook, Spyder file etc.

Working with Pandas

Loading and Saving Data with Pandas

When you want to use Pandas for data analysis, you'll usually use it in one of three different ways:

- Convert a Python's list, dictionary or Numpy array to a Pandas data frame
- Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc
- Open a remote file or database like a CSV or a JSON on a website through a URL or read from a SQL table/database

There are different commands to each of these options, but when you open a file, they would look like this:

- **pd.read_filetype()**

As I mentioned before, there are different filetypes Pandas can work with, so you would replace “filetype” with the actual, well, filetype (like CSV). You would give the path, filename etc inside the parenthesis. Inside the parenthesis you can also pass different arguments that relate to how to open the file. There are numerous arguments and in order to know all you them, you would have to read the documentation (for example, the documentation for `pd.read_csv()` would contain all the arguments you can pass in this Pandas command).

In order to convert a certain Python object (dictionary, lists etc) the basic command is:

- **`pd.DataFrame()`**

Inside the parenthesis you would specify the object(s) you’re creating the data frame from. This command also has different arguments .

You can also save a data frame you’re working with/on to different kinds of files (like CSV, Excel, JSON and SQL tables). The general code for that is:

- **`df.to_filetype(filename)`**

Viewing and Inspecting Data

Now that you’ve loaded your data, it’s time to take a look. How does the data frame look? Running the name of the data frame would give you the entire table, but you can also get the first n rows with `df.head(n)` or the last n rows with `df.tail(n)`. `df.shape` would give you the number of rows and columns. `df.info()` would give you the index, datatype and memory information. The command `s.value_counts(dropna=False)` would allow you to view unique values and counts for a series (like a column or a few columns). A

very useful command is `df.describe()` which inputs summary statistics for numerical columns. It is also possible to get statistics on the entire data frame or a series (a column etc):

- `df.mean()` Returns the mean of all columns
- `df.corr()` Returns the correlation between columns in a data frame
- `df.count()` Returns the number of non-null values in each data frame column
- `df.max()` Returns the highest value in each column
- `df.min()` Returns the lowest value in each column
- `df.median()` Returns the median of each column
- `df.std()` Returns the standard deviation of each column

Selection of Data

One of the things that is so much easier in Pandas is selecting the data you want in comparison to selecting a value from a list or a dictionary. You can select a column (`df[col]`) and return column with label `col` as Series or a few columns (`df[[col1, col2]]`) and returns columns as a new DataFrame. You can select by position (`s.iloc[0]`), or by index (`s.loc['index_one']`) . In order to select the first row you can use `df.iloc[0,:]` and in order to select the first element of the first column you would run `df.iloc[0,0]` . These can also be used in different combinations, so I hope it gives you an idea of the different selection and indexing you can perform in Pandas.

Filter, Sort and Groupby

You can use different conditions to filter columns. For example, `df[df['year'] > 1984]` would give you only the column year is greater than 1984. You can use `&` (and) or `|` (or) to add different conditions to your filtering. This is also called boolean filtering. It is possible to sort values in a certain column in an ascending order using `df.sort_values(col1)` ; and also in a descending order using `df.sort_values(col2,ascending=False)`. Furthermore, it's possible to sort values

by col1 in ascending order then col2 in descending order by using `df.sort_values([col1,col2],ascending=[True,False])`.

The last command in this section is `groupby`. It involves splitting the data into groups based on some criteria, applying a function to each group independently and combining the results into a data structure. `df.groupby(col)` returns a `groupby` object for values from one column while `df.groupby([col1,col2])` returns a `groupby` object for values from multiple columns.

Data Cleaning

Data cleaning is a very important step in data analysis. For example, we always check for missing values in the data by running `pd.isnull()` which checks for null Values, and returns a boolean array (an array of true for missing values and false for non-missing values). In order to get a sum of null/missing values, run `pd.isnull().sum()`. `pd.notnull()` is the opposite of `pd.isnull()`. After you get a list of missing values you can get rid of them, or drop them by using `df.dropna()` to drop the rows or `df.dropna(axis=1)` to drop the columns. A different approach would be to fill the missing values with other values by using `df.fillna(x)` which fills the missing values with `x` (you can put there whatever you want) or `s.fillna(s.mean())` to replace all null values with the mean (mean can be replaced with almost any function from the statistics section).

It is sometimes necessary to replace values with different values. For example, `s.replace(1,'one')` would replace all values equal to 1 with 'one'. It's possible to do it for multiple values: `s.replace([1,3],['one','three'])` would replace all 1 with 'one' and 3 with 'three'. You can also rename specific columns by running: `df.rename(columns={'old_name': 'new_name'})` or use `df.set_index('column_one')` to change the index of the data frame.

Join/Combine

The last set of basic Pandas commands are for joining or combining data frames or rows/columns. The three commands are:

- `df1.append(df2)`— add the rows in df1 to the end of df2 (columns should be identical)
- `df.concat([df1, df2],axis=1)`—add the columns in df1 to the end of df2 (rows should be identical)
- `df1.join(df2,on=col1,how='inner')`—SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values. how can be equal to one of: 'left', 'right', 'outer', 'inner'

NUMPY

Numpy is one such powerful library for array processing along with a large collection of high-level mathematical functions to operate on these arrays. These functions fall into categories like Linear Algebra, Trigonometry, Statistics, Matrix manipulation, etc.

Getting NumPy

NumPy's main object is a homogeneous multidimensional array. Unlike python's array class which only handles one-dimensional array, NumPy's ndarray class can handle multidimensional array and provides more functionality. NumPy's dimensions are known as axes. For example, the array below has 2 dimensions or 2 axes namely rows and columns. Sometimes dimension is also known as a rank of that particular array or matrix.

Importing NumPy

NumPy is imported using the following command. Note here np is the convention followed for the alias so that we don't need to write numpy every time.

- `import numpy as np`

NumPy is the basic library for scientific computations in Python and this article illustrates some of its most frequently used functions. Understanding NumPy is the first major step in the journey of machine learning and deep learning.

Sklearn

In python, scikit-learn library has a pre-built functionality under sklearn. Pre processing.

Next thing is to do feature extraction Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally our models are trained using Classifier algorithm.. We use nltk . classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered . The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre processed data. The chosen classifiers were Decision tree , Support Vector Machines and Random forest. These algorithms are very popular in text classification tasks.

SEABORN

Data Visualization in Python

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has a excellent library for you.

To get a little overview here are a few popular plotting libraries:

- **Matplotlib:** low level, provides lots of freedom
- **Pandas Visualization:** easy to use interface, built on Matplotlib
- **Seaborn:** high-level interface, great default styles
- **ggplot:** based on R's ggplot2, uses [Grammar of Graphics](#)
- **Plotly:** can create interactive plots

In this article, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library. This article will focus on the syntax and not on interpreting the graphs.

Matplotlib

Matplotlib is the most popular python plotting library. It is a low level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

1. To install Matplotlib pip and conda can be used.
2. pip install matplotlib
3. conda install matplotlib

Matplotlib is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

- `import matplotlib.pyplot as plt`

Line Chart

In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want, and plotting each column on the same axis.

Anaconda navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find

the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

What Applications Can I Access Using Navigator?

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

Advanced conda users can also build your own Navigator applications

How Can I Run Code With Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What's new in 1.9?

- Add support for **Offline Mode** for all environment related actions.
- Add support for custom configuration of main windows links.
- Numerous bug fixes and performance enhancements.

4.3 METHODOLOGY

System consists of three modules:

Front end (Application): Chabot is used to take input from user. It provides two-way communications between user and system. And user's replies are supplied at the back end to train the system.

It includes two sub modules:

User Conversation: In this sub module user's conversation is used to recognize the feeling that whether he is feeling stressful, angry, depressed or anxious etc.

Recommender System:

Recommender system is used to display the corresponding output. It works as therapist by showing the recommended quotes, images, audios or videos from database pertaining to particular class of emotion to assist user.

Back end:

Natural Language Processing Panel: This panel contains pipelined model built through natural language processing & processes the text given by user

Recognition of Emotion & Natural Language Processing:

The user's description about his emotion is processed by using natural language processing (NLP).

NLP uses machine learning to reveal the structure and mean of text. It enables you to analyze text, and also integrate it with your system. By using NLP system detects how user is feeling now, and uses this information for further classification

Classification of emotion

To provide exact recommendation to user, system classifies the emotions in four classes, i.e. Anger, fear, depression and anxiety. To classify the emotions, system uses a Collaborative filtering algorithm and Naive Bayes algorithm.

Once the text has been pre-processed and vectorised, we then train different classifiers to predict the nature and seriousness of the mental health related problem and choose a model that works best for our use case. Thus we will be training our data on the following algorithms: 1) Naïve Bayes: Probabilistic

classifiers such as Naïve Bayes which is based on the Bayes' theorem. It's a simple yet extremely efficient and popular method used for natural language processing (NLP) applications. 2) Support-Vector Machines (SVMs): Models based on supervised learning such as support-vector machines (SVMs) which is widely implemented for textual analysis and classification based problems. 3) Logistic Regression: Logistic Regression which is another widely implemented supervised learning algorithm which classifies and makes predictions on probabilities of dependent variables.

Database (SQLite):

At the back end, database is present. After knowing the class of emotion, system will fire the query to database where all the quotes, images, blogs, audios, and videos are stored. According to class of emotion the data will be fetched from database. For example, if user is feeling angry then some jokes will be fetched or if user is feeling afraid, some brave stories will be fetched from database and this will be shown in the recommender system as an output

Block diagram:

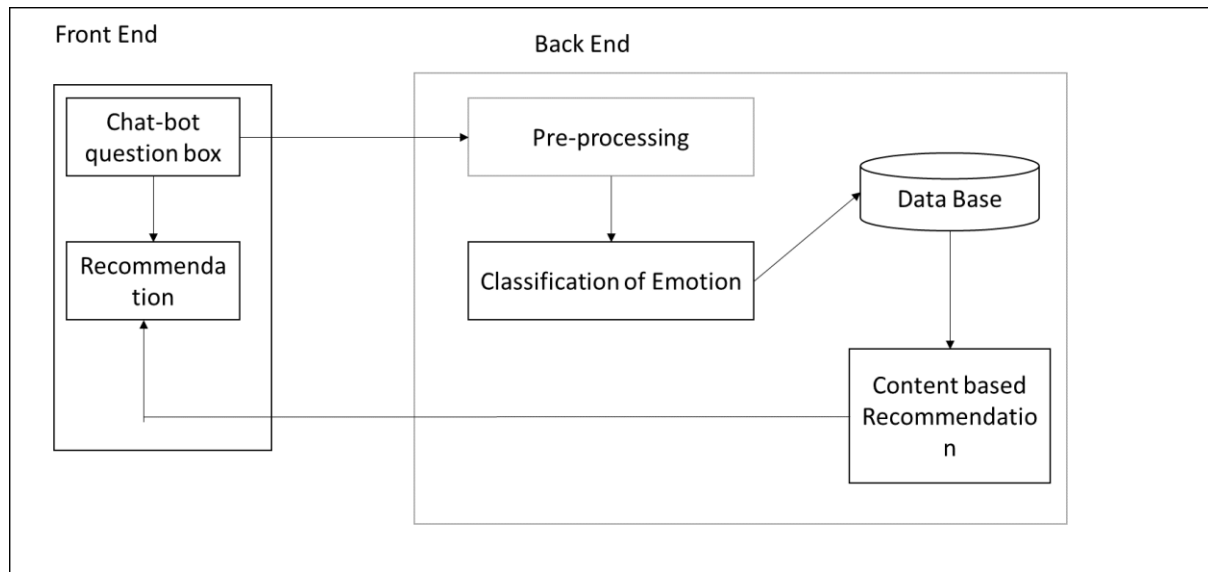
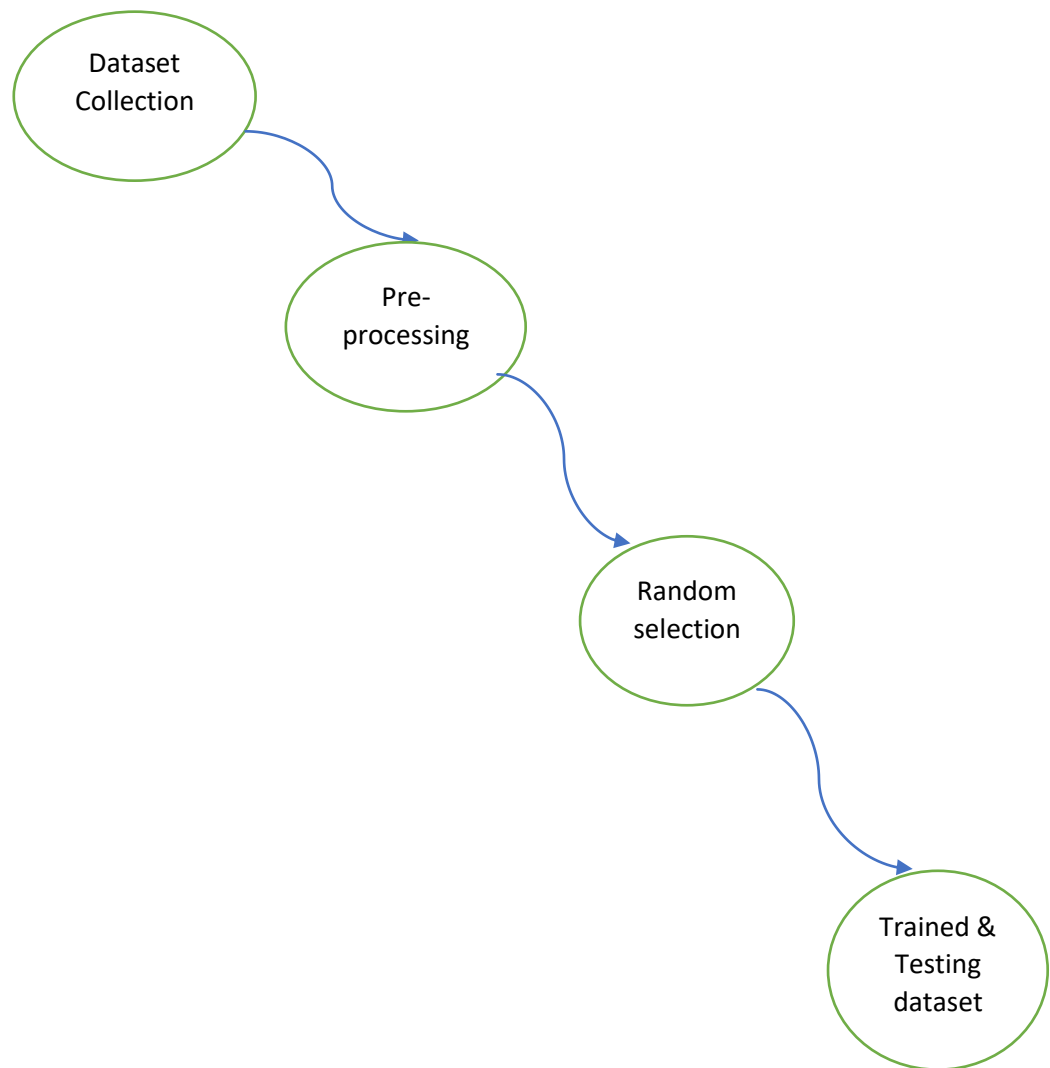
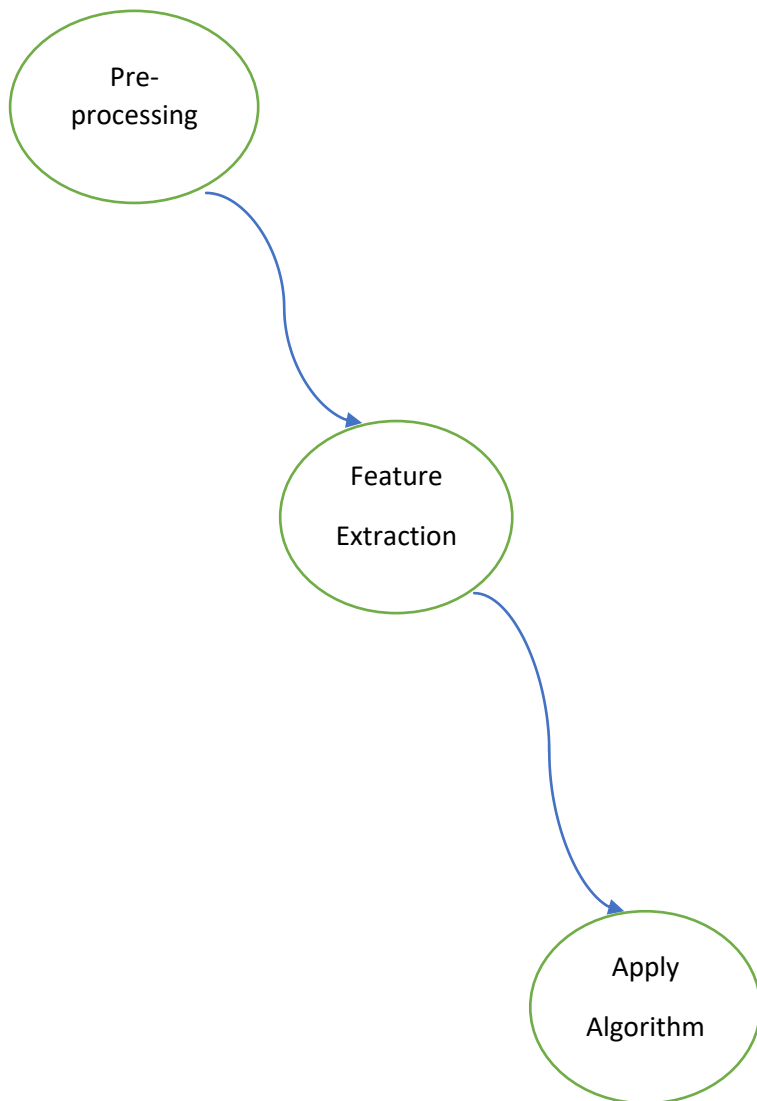


Fig 1.1 : System Architecture

DATA FLOW DIAGRAM:





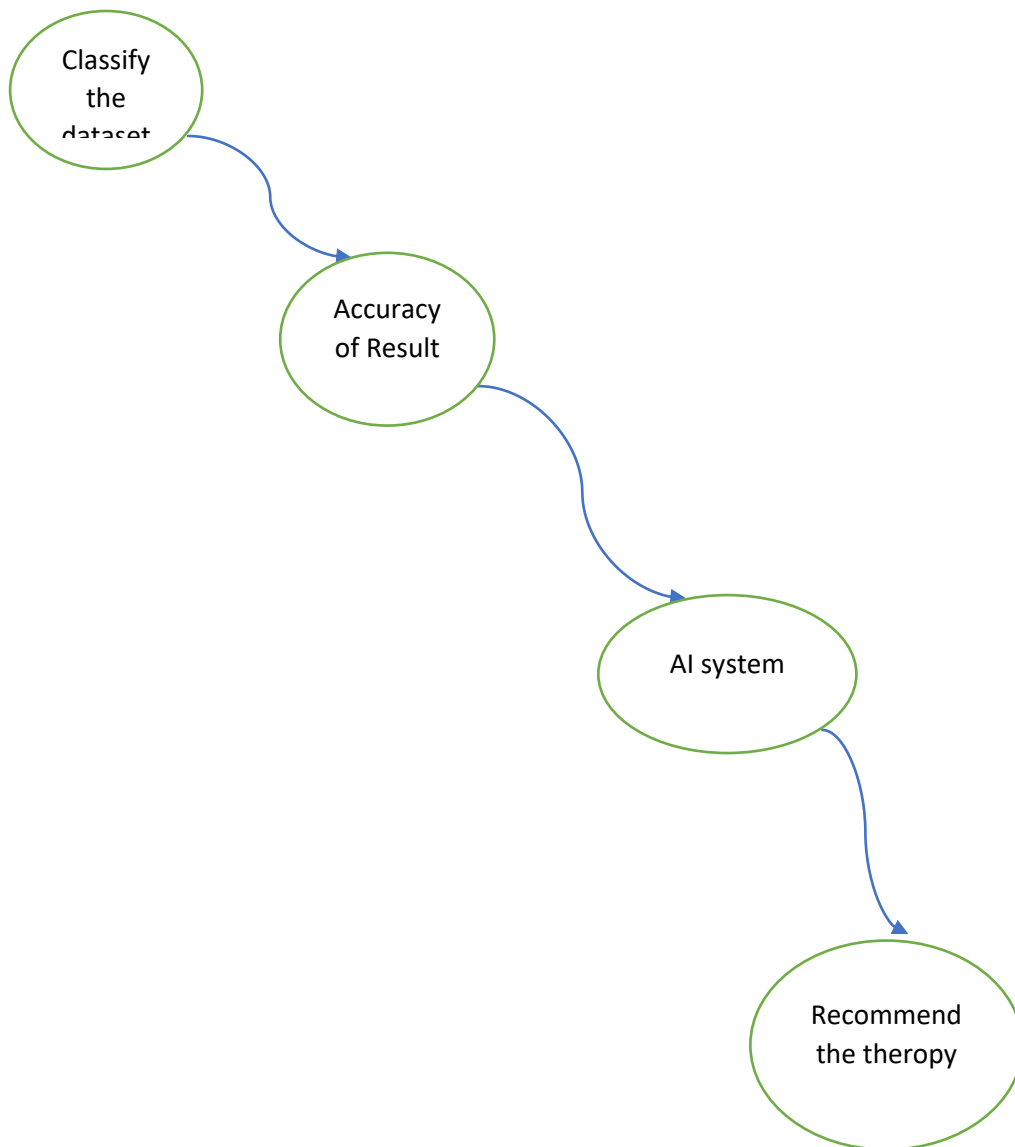


Fig 2: Data flow Diagram

UML Diagram - Use Case Diagram:

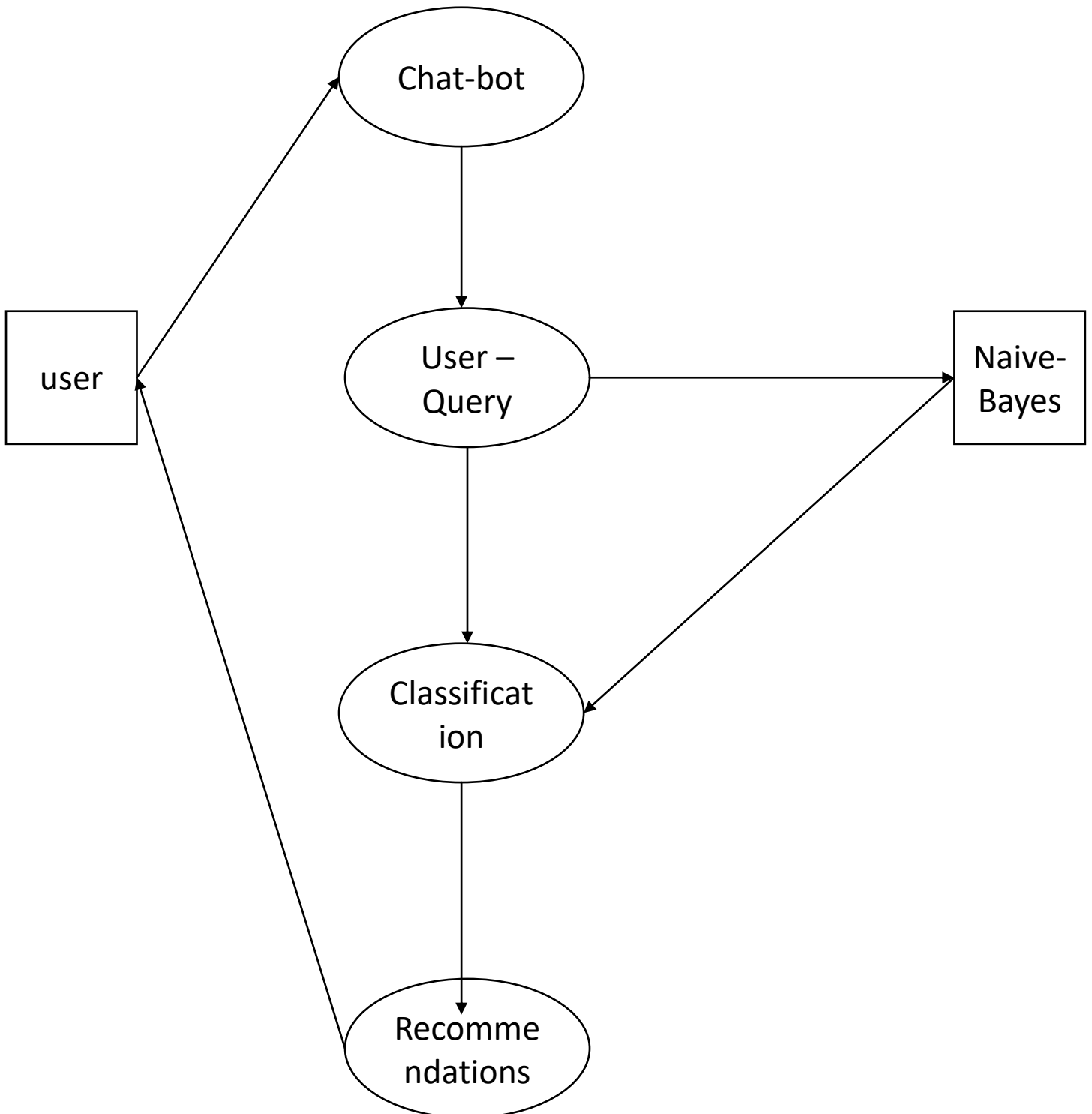


Fig 3: Use Case Diagram

Class Diagram:

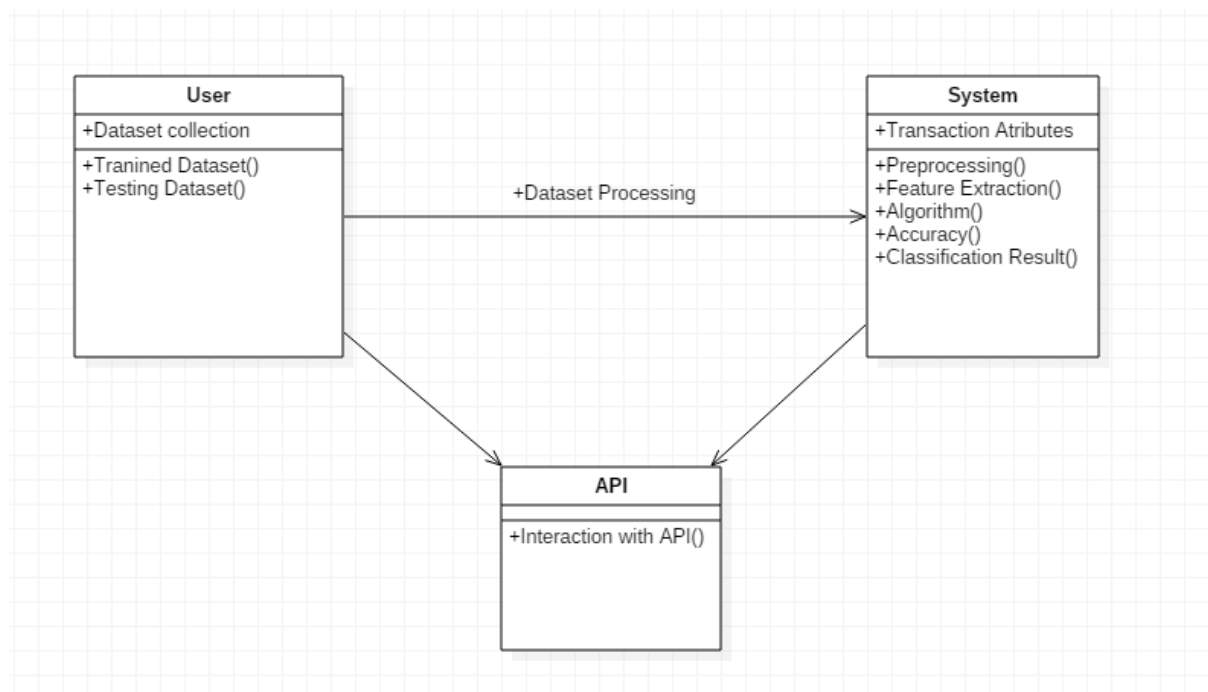


Fig 4: Class Diagram

Sequence Diagram:

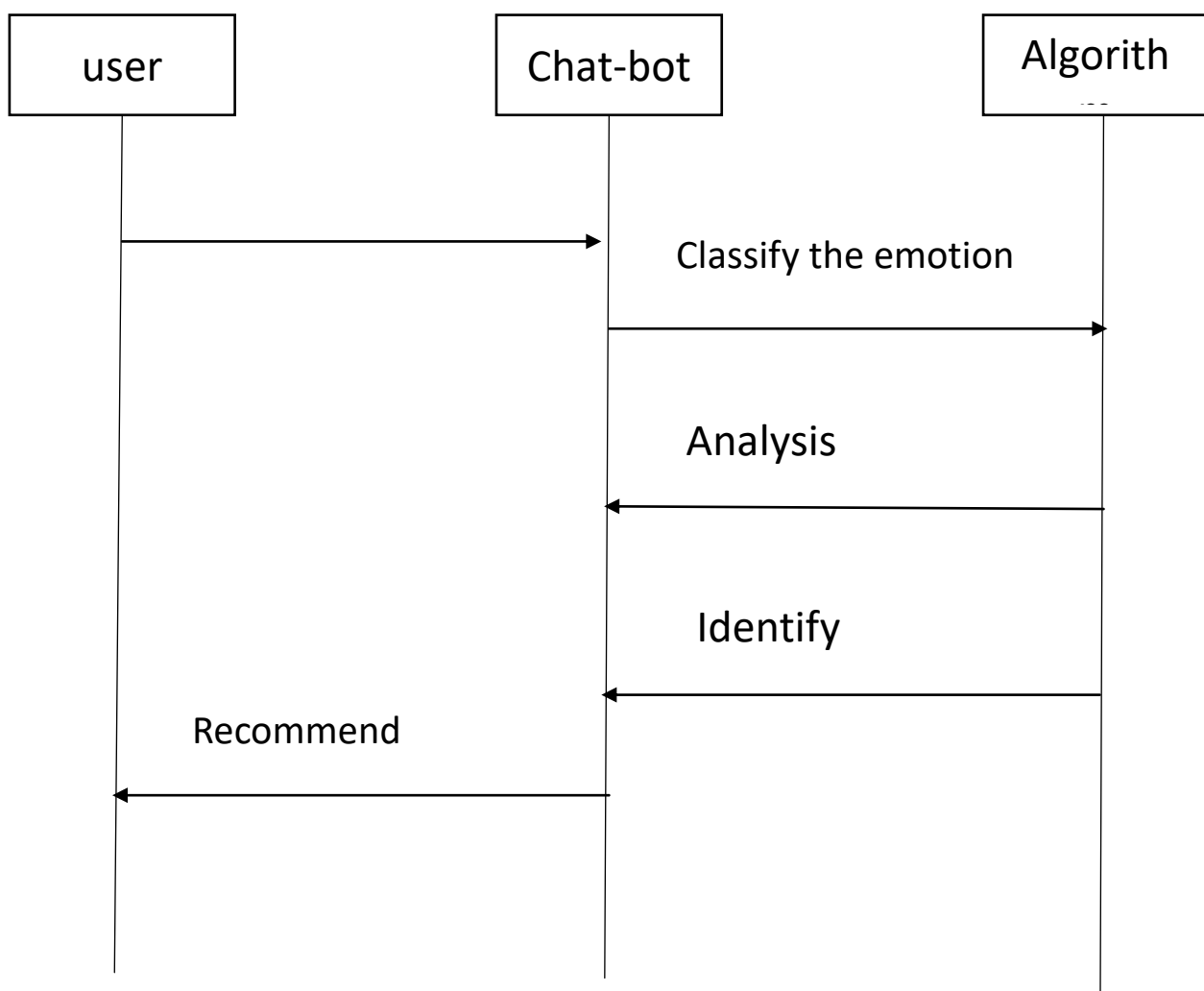


Fig 5: Sequence Diagram

Activity Diagram:

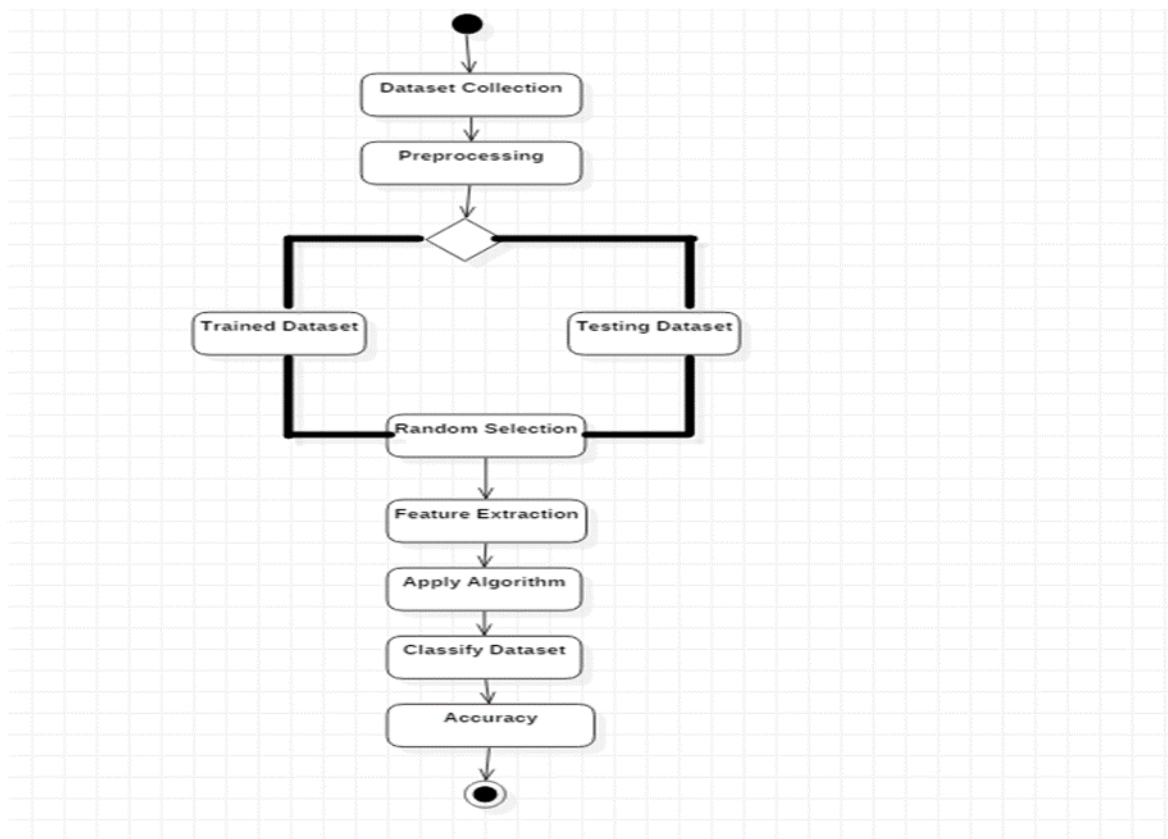


Fig :6 Activity Diagram

Collaboration Diagram:

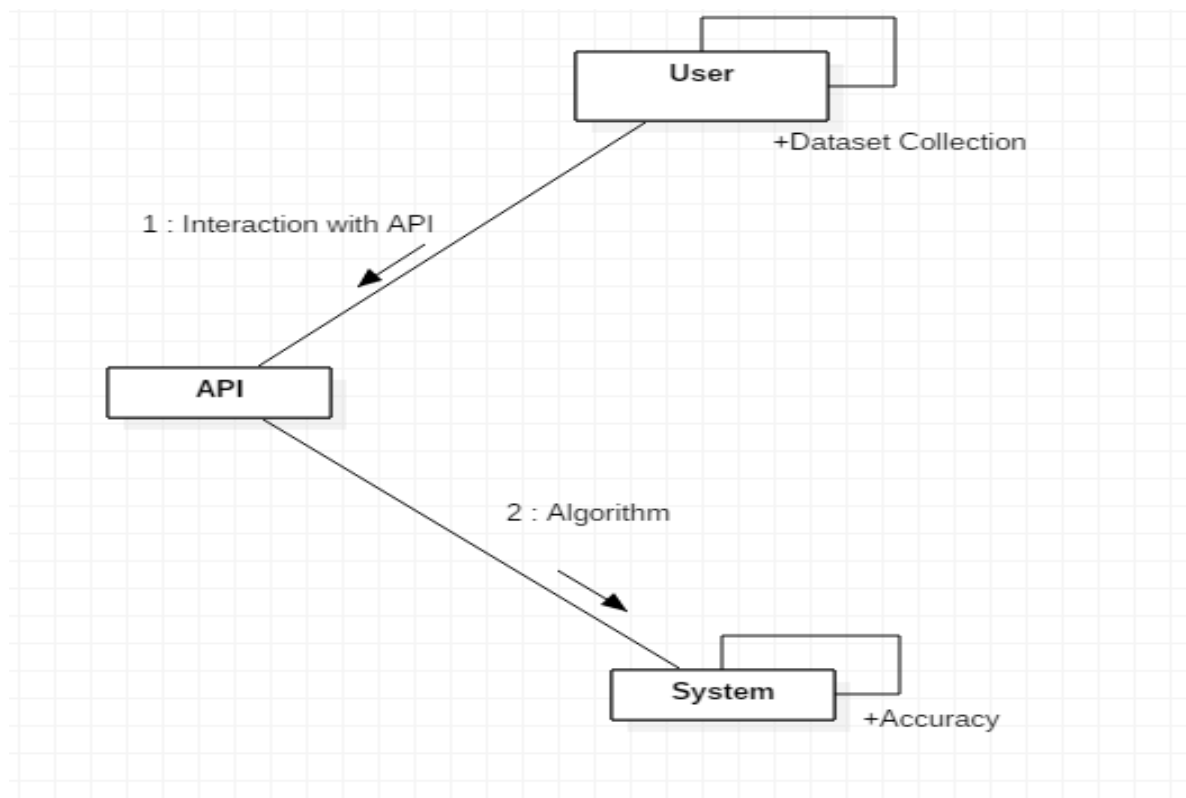


Fig :7 collaboration Diagram

CHAPTER-5

RESULT ANALYSIS

Artificial Intelligence has drastically changed the way businesses interact with their customers. Chabot's are important to improve customer experience and deployed across business functions to improve performance. Chabots are immediately available and respond quickly

Initially Chatbot only knows the greetings. The training is given to the Chatbot such as how to reply when someone is sad, which are the ingredients used to make a cake, who is the Prime Minister of India. Chatbot saves the responses given by trainer. [3] When we enter the same text again, Chatbot compares the responses matching to that text. Then Chatbot responses to user with appropriate text. The training is done by using Back propagation method.

The training is given to the Chatbot such as how to reply when someone is sad, which are the ingredients used to make a cake, who is the Prime Minister of India. Chatbot saves the responses given by trainer. [3] When we enter the same text again, Chatbot compares the responses matching to that text. Then Chatbot responses to user with appropriate text. The training is done by using Back propagation method.



Talk with an AI Therapist

While not a Therapist, the AI responds like one.
Our therapist won't ask to schedule appointments or charge you \$100 an hour.
Uses GPT3 to power the chat, and love to power the recommendations

Start a Chat



Hi! While I'm not a Therapist, I'm an AI that talks and responds like one.

What's on your mind?

Send

Generate Recommendations



Hi! While I'm not a Therapist, I'm an AI that talks and responds like one.

What's on your mind?

Hi Ramya... let's get started

I'm fine.

Send

Generate Recommendations



So why are you here on this platform?

Okay...So why don't you start by telling me what you're struggling with?

Okay. Does this stress happen often?

Send

Generate Recommendations

CHAPTER – 6

SUMMARY AND CONCLUSIONS

The web application AI Therapist will predict the emotions of user. It is designed for analysing the mental health issues faced by user and recommends user with respect to their problem. While interacting with a chatbot, user will as if he is talking with real person. Further work includes training of Chatbot to get accurate response. Training is going to be done using PyCharm for training the Chatbot, for emotion detection and for providing more accurate responses like images, audios, and videos for better therapy

Future scope

Future work could involve diagnosis and classification along with suggestions for treatment that could be recommended based on the transcripts of counselling sessions. The creation of such a platform by leveraging the use of technology and natural processing techniques would help treat mental health issues and create awareness about the same. Such a platform would not only help in identifying mental health issues at an early stage but also reduce the cost. Owing to the current COVID 19 pandemic, there is a lot of uncertainty which makes it all the more important to

address mental health. Such research could pave the way for improving counselling and therapy sessions and be an important analysis tool for therapist

APPENDIX

A. Code of the System

```
# Start with a basic flask app webpage.
from flask_socketio import SocketIO, emit
from flask import Flask, render_template, url_for, copy_current_request_context
from random import random
from time import sleep
from threading import Thread, Event
from transformers import GPT2Tokenizer
import openai
import os

flask_secret_key = "Secret_key"
openapikey = "sk-MfoHFbMTNDfJRyAnLm7pT3BIbkFJLlp5IYTADJ2mKZyHhku"
```

```

app = Flask(__name__)
app.config['SECRET_KEY'] = flask_secret_key
app.config['DEBUG'] = True

#turn the flask app into a socketio app
socketio = SocketIO(app, async_mode=None, logger=True, engineio_logger=True)

#random number Generator Thread
thread = Thread()
thread_stop_event = Event()

def randomNumberGenerator():
    """
    Generate a random number every 1 second and emit to a socketio instance (broadcast)
    Ideally to be run in a separate thread?
    """
    #infinite loop of magical random numbers
    print("Making random numbers")
    while not thread_stop_event.isSet():
        number = round(random()*10, 3)
        print(number)
        socketio.emit('newnumber', {'number': number}, namespace='/test')
        socketio.sleep(5)

@app.route('/')
def index():
    #only by sending this page first will the client be connected to the socketio instance

```

```

return render_template('index.html')

@socketio.on('connect', namespace='/test')
def test_connect():
    # need visibility of the global thread object
    global thread
    print('Client connected')

    #Start the random number generator thread only if the thread has not been started before.
    if not thread.is_alive():
        print("Starting Thread")
        thread = socketio.start_background_task(randomNumberGenerator)

@socketio.on('disconnect', namespace='/test')
def test_disconnect():
    print('Client disconnected')

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

summary_config = [
    {
        "Current Summary": "Cassie is in her first Therapy session. She is unsure if she should be here.",
        "Prompt": "Okay. Why don't we start by discussing what has been on your mind?",
        "Text": "Well, I am very frustrated lately. You see, I work as an executive assistant at SII, but it's really just a means to make money while I pursue my real goal--to be an actress. I have been taking classes since I was, like, 8 or 9 years old. Hollywood has always fascinated me. Can you imagine living in those times? The golden age of the cinema. Fred and Ginger waltzing across the big screen, Vivien Leigh and 'Gone With The Wind', Casablanca.",
        "New Summary": "Cassie is in her first Therapy session and is unsure if she should be here. Cassie has been frustrated lately. She works as an executive assistant at SII. She has always wanted to be an actress."
    },

```

```

{
  "Current Summary": "First Message",
  "Prompt": "What is your name, and what is on your mind?",
  "Text": "Hi, it's nice to meet you. I've been feeling like I've been losing control of my life lately. I've been trying to balance being great at my job with pursuing my passions, but I feel like I don't have enough time in the week.",
  "New Summary": "User feels like they are losing control of their life. They want to pursue their passions, but they don't feel like they have enough time."
},
{
  "Current Summary": "User feels like they've been in a rut lately because they didn't get much work done today.",
  "Prompt": "Why do you think you didn't get much work done today?",
  "Text": "I think its because I didn't go into the day with a plan. Like I didn't come into the day with goals I'd like to accomplish.",
  "New Summary": "User feels like they've been in a rut lately because they didn't get much work done today. This is because they didn't come into the day with a plan or goals."
},
{
  "Current Summary": "Jack feels like he's losing control of his life. He wants to pursue his passions but he doesn't feel like he has enough time. ",
  "Prompt": "I understand. What are your passions?",
  "Text": "Well, I want to start software companies in Silicon Valley. It's my dream to be a CEO, but right now I don't have the requisite skills to do so. Like I need to be able to code better, I need to be able to communicate better, and I need more courage. It's tough because I want to learn and grow and build these skills, but I also want people to respect me at my job. I just want to take a big risk sometimes.",
  "New Summary": "Jack feels like he is losing control of his life because he doesn't have the time to pursue his passion of starting a company. He doesn't think he currently have the skills to achieve his dreams and doesn't feel like he has enough time to learn them."
},
{
  "Current Summary": "Cassie has been frustrated lately. She works as an executive assistant at SII. She has always wanted to be an actress. She is in an improvisational troupe that she enjoys, but it's hard on her because her boyfriend doesn't like that she's acting. He wants her to build a real career

```

and be a productive bread winner, and get more serious about her relationship. She thinks he is jealous because she has less time for him, and because sometimes she does love scenes in the improv troupe.",

"Prompt": "How is your relationship with him besides the acting?",

"Text": "Well...good, I guess. We met in college, and have been dating ever since then. I do really like him, maybe even love him, but...he can be kind of demanding at times.",

"New Summary": "Cassie has been frustrated lately. She works as an executive assistant at SII. She has always wanted to be an actress. She is in an improvisational troupe that she enjoys, but it's hard on her because her boyfriend doesn't like that she's acting. He wants her to build a real career and be a productive bread winner, and get more serious about her relationship. She thinks he is jealous because she has less time for him, and because sometimes she does love scenes in the improv troupe. Her relationship with him is good, but he can be demanding at times."

},

{

"Current Summary": "First Message",

"Prompt": "Hi, I'm an AI Therapist. What's on your mind?",

"Text": "",

"New Summary": ""

}

]

response_config = [

{

'Current Summary': "User has a gambling problem. They took the highway on the way home, passed a casino, and went to it. She thought that she could just go for a few hours like a regular person, but ended up staying the whole night. She's in serious gambling debt, and lost a lot more money that night. Her partner also knows about her relapse.",

'Text': "Well, I was thinking that I hadn't gambled in a month and it'd been pretty easy to stay away for that long so I thought maybe I don't have a problem. So I wanted to gamble like a regular person, you know, just go in for a bit and then leave.",

'Response': "So you were wanting to test out personal control as well as test whether or not you really have a gambling problem?"

},

{

'Current Summary': "User is an Afghanistan War Veteran. They are suffering from PTSD from an improvised explosive device that killed their fellow servicemen. They feel guilty about their death because they waved their truck on ahead, after another truck broke. Shortly after being waved ahead, that truck hit the Improvised Explosive Device. She feels at fault for their death.",

'Text': "If I hadn't have waved them through and told them to carry on, this wouldn't have happened. It is my fault that they died. (Begins to cry)",

'Response': "It is certainly sad that they died. However, I want us to think through the idea that you should have had them wait and not had them go on, and consequently that it was your fault. If you think back about what you knew at the time — not what you know now 5 years after the outcome — did you see anything that looked like a possible explosive device when you were scanning the road as the original lead truck?"

},

{

'Current Summary': "Jack feels like he's losing control of his life. He wants to pursue his passions but he doesn't feel like he has enough time.",

'Text': "Hi, I'm Jack. It's nice to meet you. I've been feeling like I've been losing control of my life lately. I've been trying to balance being great at my job with pursuing my passions, but I feel like I don't have enough time in the week.",

'Response': "I think I understand. You mentioned pursuing your passions... What are they?"

},

{

'Current Summary': "Ms. Evans, also known as Cassie, is in her first Therapy session. Cassie has been frustrated lately. She works as an executive assistant at SII. She has always wanted to be an actress. She is in an improvisational troupe that she enjoys, but it's hard on her because her boyfriend doesn't like that she's acting. He thinks she should build a real career and be a productive bread winner. She thinks he is jealous.",

'Text': "Well to start off, my boyfriend is not very happy with me and the acting thing. He thinks it's ridiculous because it is so hard to make a living at it. And in the meantime, I could be building a 'real' career. He wants to get more serious with me, but wants to make sure that I can be a productive bread winner. And I think he is jealous.",

'Response': "Jealous? How so?"

},

{

'Current Summary': "Jack feels like he is losing control of his life because he doesn't have the time to pursue his passion of starting a company. He doesn't think he currently have the skills to achieve his dreams and doesn't feel like he has enough time to learn them.",

'Text': "Well, I want to start software companies in Silicon Valley. It's my dream to be a CEO, but right now I don't have the requisite skills to do so. Like I need to be able to code better, I need to be able to communicate better, and I need more courage. It's tough because I want to learn and grow and build these skills, but I also want people to respect me at my job. I just want to take a big risk sometimes.",

'Response': "Sometimes it can be hard to balance everything in your life. Why do you want to be a CEO for a software company?"

}

]

question_config = [

{

'Text': "Hi, I'm Derek... I've been going through a rough time lately with my girlfriend. She wants so much of my time and she can be demanding. It feels like she is controlling my life and not me. She's always planning things so far ahead, and I feel bad by saying no because they feel fun. But what ends up happening is that I don't actually do what I want to do by the time the plans come to fruition... Sometimes I'm afraid to say no.",

'Response': "I want you to understand, and let me emphasize this... Sometimes it's okay to say no.",

'Question': "How is your relationship with your girlfriend?"

},

{

'Text': "We were up late talking and then crashed on the living room floor. It was sweet and nice. And he didn't make any stupid comments about having sex. Or not having it, rather. It was the first time in a long while that we didn't argue a lot; and we actually talked about stuff. Meaningful things. It was like when we first started going out.",

'Response': "I'm glad you're starting to open up to your boyfriend.",

'Question': "How are you feeling right now?"

},

{

'Text': "Well, I want to start software companies in Silicon Valley. It's my dream to be a CEO, but right now I don't have the requisite skills to do so. Like I need to be able to code better, I need to be able to communicate better, and I need more courage. It's tough because I want to learn and grow and build these skills, but I also want people to respect me at my job. I just want to take a big risk sometimes.",

'Response': "Sometimes it can be hard to balance everything in your life.",

```

    'Question': "Have you ever thought that you're putting too much pressure on yourself?"
  },
  {
    'Text': "My boss thinks I'm doing great at work, but my peers aren't respecting my work. I spent an
entire month working on one review and the information wasn't even used. I just feel like I am not
being respected or looked up to.",
    'Response': "At the end of the day, you can only focus on what you can control.",
    'Question': "Let's switch the topic. Overall, how's everything in your life right now?"
  }
]

```

```

recommendation_config = [

```

```

  {
    "User": ["Hi, I'm Jack. I'm having a hard time getting out of my comfort zone", "I think it's
important to get out of my comfort zone so that I can continue to learn. For example, last night, I
wanted to talk to some new people, but was afraid to, so I just talked with people I already knew.",
"Because I'm afraid of being weird and leaving a bad first impression", "I'm not sure how often, but
usually in social situations I tend not to introduce myself or start new conversations with people",
"Because I think I'm afraid of embarrassing myself, or that they won't like me", "I'm not sure yet, I
think I need to step outside my comfort zone"],
    "AI": ["Why is it important to get out of your comfort zone? Can you give me a specific example of
how you stayed in your comfort zone?", "Why were you afraid to talk with new people?", "Do you
often feel like that?", "Why do you think that is?", "And how do you usually correct this?"],
    "Recommendation": "The only way to grow is to step outside your comfort zone. So why don't you
try something small like small talk with an acquaintance, and then become more comfortable talking
to new people. That way you can eventually build up the skills to talk with strangers."

```

```

  },
  {
    "User": ["I've been dealing with relationship issues recently", "It has been painful", "Well its hard
to communicate with my girlfriend because she can be very demanding.", "She doesn't fully listen to
my opinion, and assumes that we should always do what she wants to do", "I think it should be a
compromise, and/or we work towards finding the right solution together", "She usually just gets mad
at me, and then I have to be defensive or let her win the argument", "I feel like I'm not able to be
myself and have to let her run all over me", "That I'm able to have my opinions, do what I want, and
pursue my own passions without being burdened by the desires of someone else", "I would love to
build my own projects, hang out with friends, do stupid/weird things. I think I'm a naturally goofy type,
fun loving character and I wish I could explore that side of myself more"],

```

"AI": ["Okay, how has this affected you?", "How have these difficulties manifested?", "How does she come across to be demanding?", "What are your expectations for how your partner should behave towards you?", "What have past conflicts with this person been like?", "How does this situation make you feel?", "What would being yourself mean?", "What did you do on your own? What kinds of things do you like to do?"]],

"Recommendation": "Have you tried talking with her honestly about being able to voice your opinion? It's important to stand up for yourself as well as not bend to the will of others. If she's unwilling to let you voice or listen to your opinion, it may make sense to leave the relationship."

},

{

"User": ["I'm trying to become a better friend", "Why so that I can build stronger relationships with people. Its important for us humans to connect", "I need to be more comfortable reaching out to people", "I think it starts by reaching out! I think once I try it a few times, it will become easier", "I hope so", "I will try"],

"AI": ["Okay... Why? And who?", "I think that is an admirable goal. What skills do you need to become a better friend?", "What can you do to be more comfortable?", "Do you think you can fulfill this role?", "Should I be concerned that you hope so? Maybe you should try first?"]],

"Recommendation": "Reaching out to people isn't easy at first. Don't be discouraged, and lower your expectations on yourself. Try by saying 'hi' to someone new."

},

{

"User": [],

"AI": [],

"Recommendation": ""

}

]

```
def create_summary_dict ():
```

```
    summary_dict = {
```

```
        'Current Summary': "",
```

```
        'Prompt': "",
```

```

        'Text': "",
        'New Summary': ""
    }
    return summary_dict

def create_response_dict ():
    response_dict = {
        'Current Summary': "",
        'Text': "",
        'Response': ""
    }
    return response_dict

def create_question_dict ():
    question_dict = {
        'Text': "",
        'Response': "",
        'Question': ""
    }
    return question_dict

def generate_summary_prompt(config):
    end_token = "\n###\n\n"

    generated_prompt = "A summarizing tool for therapy.\n\n###\n\n"
    for r in config:
        current_summary = "Current Summary: " + r['Current Summary'] + "\n"
        prompt = "Prompt: " + r['Prompt'] + "\n"
        text = "Text: " + r['Text'] + "\n"

```

```

new_summary = "New Summary: " + r['New Summary'] + "\n"

response_prompt = current_summary + prompt + text + new_summary + end_token
generated_prompt = generated_prompt + response_prompt


generated_prompt = generated_prompt.rstrip().rstrip('###').rstrip()
return generated_prompt


def generate_response_prompt(config):
    end_token = "\n###\n\n"

    generated_prompt = "The following is a conversation with an AI therapist. The therapist is helpful,
creative, clever, and very friendly. At the beginning of the conversation, they ask questions to learn
more about the User's problems, but once the Current Summary gets long, they start to offer
recommendations.\n\n"

    for r in config:
        current_summary = "Current Summary: " + r['Current Summary'] + "\n"
        text = "Text: " + r['Text'] + "\n"
        response = "Response: " + r['Response'] + "\n"

        response_prompt = current_summary + text + response + end_token
        generated_prompt = generated_prompt + response_prompt

    generated_prompt = generated_prompt.rstrip().rstrip('###').rstrip()
    return generated_prompt


def generate_question_prompt(config):
    end_token = "\n###\n\n"

```

generated_prompt = "Therapist asking a question based on the past response and the previous users text. The goal is for the question to help the user open up about their problems.\n\n"

for r in config:

text = "Text: " + r['Text'] + "\n"

response = "Response: " + r['Response'] + "\n"

question = "Question: " + r['Question'] + "\n"

response_prompt = text + response + question + end_token

generated_prompt = generated_prompt + response_prompt

generated_prompt = generated_prompt.rstrip().rstrip('###').rstrip()

return generated_prompt

def generate_recommendation_prompt(config):

end_token = "\n###\n\n"

response_prompt = "

generated_prompt = "Generate recommended next steps based on a therapy conversation.\n\n###\n\n"

for r in config:

larger_num = max(len(r['User']), len(r['AI']))

gen_text = "

for i in range(0, larger_num):

if i+1 > len(r['User']):

msg = "

else:

msg = 'User: ' + r['User'][i]

```

        if i+1 > len(r['AI']):
            response = ""
        else:
            response = 'AI: ' + r['AI'][i]
        gen_text = gen_text + msg + '\n' + response + '\n'

    recommendation = 'Recommendation: ' + r['Recommendation'] + '\n'
    response_prompt = response_prompt + gen_text + recommendation + end_token

    generated_prompt = generated_prompt + response_prompt

    generated_prompt = generated_prompt.rstrip().rstrip('###').rstrip()
    return generated_prompt

#call summary api
def call_summary_api(the_prompt):
    #update values
    response = openai.Completion.create(
        engine="davinci",
        prompt = the_prompt,
        max_tokens=700,
        temperature=.5,
        #top_p=1, #Don't use both this and temp (according to OpenAI docs)
        frequency_penalty=0.2,
        presence_penalty=0.0,
        n=1,
        stream = None,

```

```

logprobs=None,
stop = ["\n"])

return (response)

#call api
def call_response_api(the_prompt):
    #update values
    response = openai.Completion.create(
    engine="davinci",
    prompt = the_prompt,
    max_tokens=400,
    temperature=.7,
    #top_p=1, #Don't use both this and temp (according to OpenAI docs)
    frequency_penalty=0.2,
    presence_penalty=0.0,
    n=1,
    stream = None,
    logprobs=None,
    logit_bias={30:1},
    stop = ["\n"])

    return (response)

#call api
def call_question_api(the_prompt):
    #update values
    response = openai.Completion.create(
    engine="davinci",

```

```

prompt = the_prompt,
max_tokens=400,
temperature=.7,
#top_p=1, #Don't use both this and temp (according to OpenAI docs)
frequency_penalty=0.2,
presence_penalty=0.0,
n=1,
stream = None,
logprobs=None,
stop = ["\n"])
return (response)

def call_recommendation_api(the_prompt):
    #update values
    response = openai.Completion.create(
        engine="davinci",
        prompt = the_prompt,
        max_tokens=500,
        temperature=1,
        #top_p=1, #Don't use both this and temp (according to OpenAI docs)
        frequency_penalty=0.2,
        presence_penalty=0.0,
        n=3,
        stream = None,
        logprobs=None,
        best_of = 3,
        stop = ["\n"])
    return (response)

```

```

openai.api_key = openapikey

@socketio.on('python_old', namespace='/test')
def call_therapist(msg, namespace):
    print(msg['the_text'])
    input_text = msg['the_text']

    #input text into the generated starter dialogue
    summary_config[-1]['Text'] = input_text
    #generate prompt
    generated_summary_prompt = generate_summary_prompt(summary_config)

    #Check to see if the token is too large
    summary_tokens = tokenizer(generated_summary_prompt)['input_ids']
    print(len(summary_tokens) + 700)
    if len(summary_tokens) + 700 > 2048:
        summary_config.pop(1)
        generated_summary_prompt = generate_summary_prompt(summary_config)

    #Call Summary API
    summary_response = call_summary_api(generated_summary_prompt)
    #Clean Result
    clean_summary_response = summary_response.choices[0].text.rstrip().lstrip()
    #Add New Summary to Last Summary Config
    summary_config[-1]['New Summary'] = clean_summary_response

    #Create new response config and add requisite info
    new_response_config = create_response_dict()

```

```

new_response_config['Text'] = input_text
new_response_config['Current Summary'] = clean_summary_response

#Create new summary config and add summary response
new_summary_config = create_summary_dict()
new_summary_config['Current Summary'] = clean_summary_response

#Add new response config to the old response config
response_config.append(new_response_config)

#Generate prompt for response config and call API
generated_response_prompt = generate_response_prompt(response_config)

#Check to see if response prompt is too large
response_tokens = tokenizer(generated_response_prompt)['input_ids']
print(len(response_tokens) + 400)
if len(response_tokens) + 400 > 2048:
    response_config.pop(1)
    generated_response_prompt = generate_summary_prompt(response_config)

#Call Response API
response_response = call_response_api(generated_response_prompt)
#Clean Result
clean_response_response = response_response.choices[0].text.rstrip().lstrip()

#Add new response to new response config
new_response_config['Response'] = clean_response_response

#Add new response to new summary config
new_summary_config['Prompt'] = clean_response_response
print(len(clean_response_response))

```

```

clean_question_response = ""

#test if the response ended in a question. If not, call the question api
if clean_response_response[len(clean_response_response) - 1] != '?':
    print('The forced question section!!!!')
    new_question_dict = create_question_dict()
    new_question_dict['Text'] = input_text
    new_question_dict['Response'] = clean_response_response

    question_config.append(new_question_dict)
    generated_question_prompt = generate_question_prompt(question_config)
    print(generated_question_prompt)

    question_response = call_question_api(generated_question_prompt)
    clean_question_response = question_response.choices[0].text.rstrip().lstrip()

    clean_response_response = clean_response_response + ' ' + clean_question_response

    question_config.pop(len(question_config) - 1)

#Add both new configs to the old configs
summary_config.append(new_summary_config)
summary_config.pop(3)
response_config.pop(2)

#Add relevant text to the recommendations config
recommendation_config[-1]['User'].append(input_text)
recommendation_config[-1]['AI'].append(clean_response_response)

```

```

print(generated_summary_prompt)

print(clean_summary_response)

socketio.emit('to_socket_string', {'string': clean_response_response}, namespace='/test')

conversation_config = [
    {
        "User": ["I've been feeling really sad over the loss of my boyfriend",
            "Well, I keep feeling like I just can't move on like ...I have this baggage that is going to follow me with whomever I meet or...or any path I go in life.",
            "No.. I guess it doesn't but.....I don't know.",
            "Well.....I think it has something to do with my dad.",
            "Well, I guess growing up my dad never made me feel like I was important.... always saying things to belittle me and stuff like that. And then...when...Eric would do that stuff to me, you know like, he would go riding and not pay any attention to me or...just play poker and...disregard whatever I said to him, it kind of made me feel the same way, you know?",
            "Yeah...well...I guess it's all my fault though, you know because I just can't control my attitude",
            "I guess if....I guess I can if I really tried.",
            "Well, I don't think so.",
            "Yeah.",
            "Umm....I just can't control my mouth sometimes. Like I think that I just provoke them by just things that I say, you know. Maybe to them it's stupidity, I don't know, but, I guess I say things to hurt them and then it just instigates a fight automatically.",
            "Mmmm....(silence) Like, that it hurts me?"],
        "AI": ["How are you handling the feelings of sadness?",
            "Does this baggage you're referring to have to follow you around?",
            "Where do you think this baggage really comes from?",
            "I see...Talk to me about that.",
            "So you're feeling like your boyfriend reminds you of when your father used to.....make you feel like you weren't important",
            "So you can't control your attitude huh",

```

"Are you saying you haven't really tried?",

"Earlier you were talking about how your boyfriend reminds you of your dad...",

"Tell me... How do you think that all ties into with your attitude?",

"It sounds like this attitude your referring to may be a way to protect your feelings.",

"Yeah, like that it hurts you.... You know, anger usually comes right after hurtful feelings and the attitude may be a way to cope with those feelings."

]

},

{

"User": ["Its been tough lately. My parents have been dealing with marital issues, and I just did poorly on my exams. It's just annoying because I know that I should have done better during my exams, and I just didn't. I guess I just beat myself up a lot",

"Maybe more than other people. My parents have always told me in a loving way that you should always do your best and get the best. I agree with that, but when I don't do that... I don't know... it annoys me a lot, and I'm afraid it annoys them",

"No...",

"I think it doesn't really sound good. I don't really talk to them a lot. I don't try to talk to them because it makes me feel worse, but I don't think their relationship is getting any better.",

"For a little bit",

"Not very fun... No... it was quite tense most of the time. It stressed me out more than it relaxed me."],

"AI": ["You say 'I know I should have done better.' I guess hearing it from perspective, I'm thinking you had a lot going on at the time. It'd be understandable absolutely that you wouldn't do as well as normal. I suppose I'm just wondering if you have very high standards for yourself.",

"It sounds like you feel you should always do your best. Are there any kind of exceptions to that?",

"Okay, so I can then that was a very difficult time for you. What's the situation with your parents at the moment?",

"Were you home for the Summer?",

"And how was that?",

"Okay, it sounds like a lot has been happening. You've been feeling very low, you've had University pressure on, and you've had problems with your parents. You can't really feel like you can reach out to your parents. So I suppose then, could you tell me a bit more about how you've been feeling this past week?"

```

    ]
    },
    {
        "User": ["I got frustrated on Friday when I had just implemented a new policy for staff members. I
had imagined that I would get a lot of phone calls about it because I always do but I ended up snapping
at people over the phone.",
            "I felt quite stressed and also annoyed at other staff members because they didn't
understand the policy.",
            "I guess I was thinking that no-one appreciates what I do."],
        "AI": ["And how were you feeling at that time?",
            "And what was going through your mind?",
            "Okay. You just identified what we call an automatic thought. Everyone has them. They are
thoughts that immediately pop to mind without any effort on your part. Most of the time the thought
occurs so quickly you don't notice it but it has an impact on your emotions. It's usually the emotion
that you notice, rather than the thought. Often these automatic thoughts are distorted in some way
but we usually don't stop to question the validity of the thought. Tell me, what is the effect of
believing that 'no-one appreciates you?'",]
    },
    {
        "User": [],
        "AI": []
    }
]

```

```
def generate_conversation_prompt(config):
```

```
    end_token = "\n###\n\n"
```

```
    response_prompt = "
```

```

    generated_prompt = "The following is a conversation with an AI therapist. The AI is helpful, clever,
and humorous. They want to offer recommendations to help the User deal with their
issues.\n\n###\n\n"

```

```
    for r in config:
```

```
        larger_num = max(len(r['User']), len(r['AI']))
```

```

gen_text = ""
for i in range(0, larger_num):

    if i+1 > len(r['User']):
        msg = ""
    else:
        msg = 'User: ' + r['User'][i]

    if i+1 > len(r['AI']):
        response = ""
    else:
        response = 'AI: ' + r['AI'][i]
    gen_text = gen_text + msg + '\n' + response + '\n'

response_prompt = response_prompt + gen_text + end_token

generated_prompt = generated_prompt + response_prompt

generated_prompt = generated_prompt.rstrip().rstrip('###').rstrip()
generated_prompt = generated_prompt + '\nAI:'
return generated_prompt

def call_conversation_api(the_prompt):
    #update values
    response = openai.Completion.create(
        engine="davinci",

```

```

prompt = the_prompt,
max_tokens=400,
temperature=.7,
#top_p=1, #Don't use both this and temp (according to OpenAI docs)
frequency_penalty=0,
presence_penalty=0,
n=1,
stream = None,
logprobs=None,
stop = ["\n"])
return(response)

@socketio.on('python', namespace='/test')
def call_therapist_responses(msg, namespace):
    print('We have lift off')

    input_text = msg['the_text']
    response_tokens = 400

    #Add user text to conversation config
    conversation_config[-1]['User'].append(input_text)

    #generate prompt
    generated_prompt = generate_conversation_prompt(conversation_config)
    print(generated_prompt)

    #Check to see if the token is too large
    conversation_tokens = tokenizer(generated_prompt)['input_ids']

```

```

print(len(conversation_tokens) + response_tokens)
if len(conversation_tokens) + response_tokens > 2048:
    conversation_config.pop(0)
    generated_prompt = generate_conversation_prompt(conversation_config)

#Call Response API
response_response = call_conversation_api(generated_prompt)

#Clean Result
clean_response_response = response_response.choices[0].text.rstrip().lstrip()
print(clean_response_response)

#Add AI Response to the conversation config
conversation_config[-1]['AI'].append(clean_response_response)

#Add relevant text to the recommendations config
recommendation_config[-1]['User'].append(input_text)
recommendation_config[-1]['AI'].append(clean_response_response)

socketio.emit('to_socket_string', {'string': clean_response_response}, namespace='/test')
return(clean_response_response)

@socketio.on('recommendation_python', namespace='/test')
def get_recommendations(msg):
    print(openapikey)
    print('Called Recommendation Python')
    generated_recommendation_prompt =
generate_recommendation_prompt(recommendation_config)
    print(generated_recommendation_prompt)
    recommendation_response = call_recommendation_api(generated_recommendation_prompt)

```

```

print(recommendation_response)

array_recommendations = []
for i in range(0, len(recommendation_response.choices)):
    l = recommendation_response.choices[i].text.lstrip().rstrip()
    array_recommendations.append(l)

    socketio.emit('recommendation_socket', {'recommendations_array': array_recommendations},
namespace='/test')

@app.route('/chat')
def load_chat():
    return render_template('chat.html')

@app.route('/recommendations')
def load_recommendations():
    return render_template('recommendations.html')

@app.route('/')
def load_home():
    return render_template('index.html')

if __name__ == '__main__':
    socketio.run(app)

```

Reference

- [1] Astim jan, Hongying Meng, YonaFalinie, A. Gaus, and Fan Zhang, "Artificial intelligence System for automatic Depression Level Analysis Through Visual and Vocal Expression", volume 10, issue 3, July 2017.
- [2] Mandar Deshpande, Vignesh Rao, "Depression Detection using Emotion Artificial Intelligence", November 2017.
- [3] Tatsuya Nomura, "Problems of Artificial Emotions in Mental Therapy", July 2003.
- [4] Pietro Morasso, Maura Cassadio, PsicheGiannoni, Lorenzo Masia, Vittorio Sanguineti, Valntina Squeri, Elena Vergaro, "Desirable Features of 'Humanoid' Robot-Therapist", September 2009.
- [5] Aisling Kelliher, Barbara Barry, "Designing Therapeutic Care Experiences with Artificial Intelligence in Mind", August 2017.
- [6] A. Gelbukh, "Natural Language Processing", 5 February 1998.
- [7] R. G. Jackson, R. Patel, N. Jayatilleke, A. Kolliakou, M. Ball, G. Gorrell et al., "Natural language processing to extract symptoms of severe mental illness from clinical text: the Clinical Record Interactive Search Comprehensive Data Extraction (CRIS-CODE) project," BMJ Open, vol. 7, no. 1, 2017, pp. e012 012–e012 012

