A MOBILE APP FOR TRACKING PSYCHOLOGICAL MOOD CHANGES AND PROVIDING E-THERAPY USING NATURAL LANGUAGE PROCESSING AND GPT-3

Xianghao Zhang[1], Austin Amakye Ansah[2]

[1]Harvard-Westlake School, 3700 Coldwater Canyon Ave, Los Angeles, CA 91604
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Five percent of individuals aged 18 and over reported regularly having feelings of depression in 2022 [6].The app developed uses Generative Pre-trained Transformer (GPT) to analyze sentiment values and provide appropriate human-like responses much like speaking to a therapist [7]. The application provides the user with a means of journaling and analyzing their psychological fluctuations daily. An application of this is finding a cause for negative mood changes through frequent journaling and mood check ins. The evaluation of the app shows that it is good enough for journaling and basic AI interaction with respect to mental health guidance. In the end, we delivered a ready to use journaling app with GPT integration included to provide an experience akin to speaking to a real therapist.

KEYWORDS

Gpt, Mental Health, AI, Natural Language Processing

1. INTRODUCTION

The issue at hand is the prominence of mental welfare related problems in individuals. In 2022, five percent of individuals aged 18 and over report regularly having feelings of depression [1]. 12.5% of that age group reported regularly having feelings of worry, nervousness, or anxiety in the same year [6]. Often, it is difficult to identify the cause of mental health fluctuations, and while traditional therapy offers help, it does not always solve the problem. Therapists are human and are therefore prone to bias [2]. Another issue is that not all therapists are experts. Some are prone to misdiagnosing or downright dismissing the cause of root concern of their patients. Indeed, many people who self-diagnose their psychological disorders may be told that their ailment is not real, even in the case of a true disorder. The problem isn't so much that people are depressed, but rather that they are not able to find the exact root cause of their mental health problems. We propose a psychological journaling app that analyzes daily moods and journal entries and provides AI assisted recommendations. The app works by asking daily for a mood check in and storing that data in firebase. Users have the ability to create journal entries like a diary and chat with a GPT powered chatbot [12]. Journal entries have a sentiment score tallied to them when they are edited based on the content of the entry. The chatbot is set up to provide responses aligning with those of a mental health assistant. This is an efficient solution for tracking mental health changes due to the collection and analysis of user data by the AI. The calendar system also plays a role in showing

David C. Wyld et al. (Eds): SIPR, NCWC, BIBC, SOFEA, DSML, ARIA, NLP, CSEN -2023
pp. 321-332, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131925

mood history through time. This gives the user the authority to monitor their mental health fluctuations throughout the week and figure out the root cause of their emotional distress.

2. CHALLENGES

Throughout the development of the app, a few issues were encountered in the backend internals and UX design process.

2.1. Fetching Valid Mood Check in Data from the Database

One issue was the fetching of mood check-in data from firebase. The mood check-in has a timeStamp field called 'checkedIn' which stores information about the time of the user check in. The timeStamp includes both the date and time of the checkIn. When the user checks in and opens the app one day later, the mood check-in for the previous day is still shown. A solution to this issue would be removing the time aspect from the data type and only using the current day's date information. This would make it easier to filter the database for entries created adjacent to each other in time.

2.2. Choosing a Good Calendar Library

There were two visual options when building the mood calendar view. Either a custom-built calendar or a third-party library could be used to show a calendar with emojis on check-in dates [13]. The library/package option was chosen out of convenience. One common issue with the external packages was the inability to place custom widgets on calendar tiles.

2.3. Chat Messages not Being Loaded From the Last Entry in the UI

Chat messages were being loaded in full of firebase, and the listview used to view the message bubbles did not scroll to the bottom when loaded [11].

A fix for the bulk message loading would be to load the messages from firebase in batches. Regarding the issue of the listview not scrolling to the bottom of the chat, the way the message list is being generated has to be examined in order to solve that problem.

3. SOLUTION

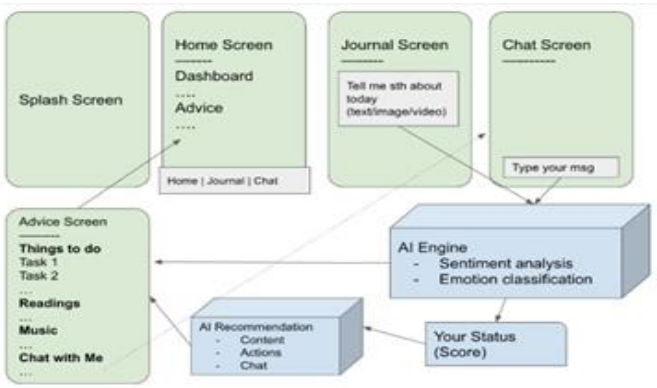

Figure 1. App UI Flowchart

The chat, Journal page, and recommendation page all work together to provide sentiment context

for the GPT API. All information (user data, and journal entries are stored within Firebase. All information fetched from the bot in the chat side is stored as conversational data. Most data types in the app contain a variable called ‘sentiment Score’ that is updated when the data type is created within a variable like a journal entry or a message. The average sentimentScore for all journal entries is computed and fed as prompt input for GPT. GPT then responds accordingly with output catered based on the base prompt, and the sentiment score provided. The mood checkIn feature of the app allows the user to log their wellbeing every day. This is data that is used to further feed the GPT prompt with context. The GPT model is constructed with a GPT library class that is fed the last 4 messages of the chat as preliminary context. This gives GPT a bit of memory about the last few responses it gave and the questions the user asked. Finally, that is sent back to the openAI API for a response.

The tech stack used is Flutter (on top of dart), Firebase due to its ease of access, and openAI GPT API for its affordable models.

An activity recommendation page also uses the GPT API and sentiment context to provide the best possible activities or tasks for the user. The base prompt takes the average journal sentiment score into account and uses this value to gauge GPT’s responses.

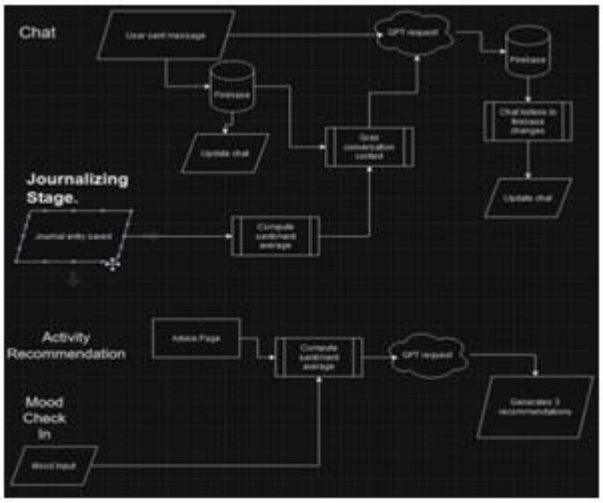

Figure 2. Overview of System

Component 1. AI Chat and Recommendation System

Two of the most important parts of the app are the AI chat component (Fig.3), and the AI recommendation component (Fig 4). Both of these pages use GPT requests to leverage data. GPT’s transformer architecture makes natural language processing easy and therefore it was a suitable choice for a natural language processing API, compared to NLTK (in python) [15]. When a user sends a message, the message is first sent to firebase and added to the main conversation record. Right after this, the last 4 messages of the chat are compiled and fed as data to the gpt class. The chat awaits a response from Firebase and updates automatically when the ChatBot receives data from the GPT API.

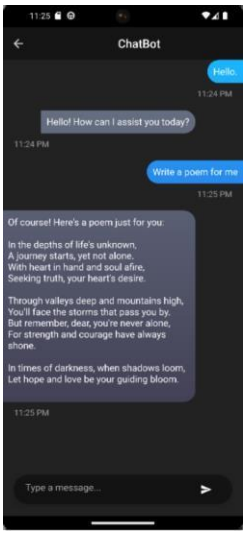

Figure 3. Chat UI

Figure 4. Recommendation

```
Future<gpt.ChatCTResponse?> sendRequest(List<Message> messages) async {
  gpt.OpenAI openai = gpt.OpenAI.instance;
  String gptKey = await AuthHandler().getGPTKey();
  openai = openai.build(token: gptKey);

  final request = gpt.ChatCompleteText(
      messages: constructMessagesFromMessageClass(messages),
      model: gpt.GptTurboChatModel(),
      temperature: 0.5); // gpt.ChatCompleteText

  final response = await openai.onChatCompletion(request: request);

  return response;
}
```

Figure 5. SendRequest function

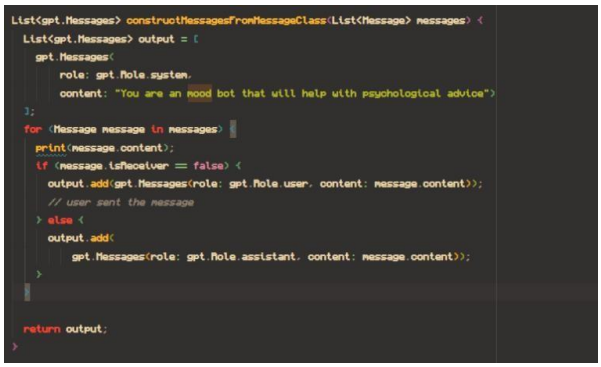

Figure 6. ConstructMessagesFromMessageClass function

In Figure 1, with the function titled "sendRequest," the function creates an openAI class instance. Next, the gpt API key is fetched from firebase (not stored locally) and added to the openAI instance [10]. Next, a GPT request is constructed using the message context function to give the API some preliminary conversation history and to serve as short term memory for the bot. Next, the function awaits the api response. The constructMessagesFromMessageClass function takes one parameter, which is a list of 'Message'. Message is a custom datatype created to store information about a message, like whether the bot sent the message, or the user did, and the timestamp for the message. This is converted into a GPT Message and categorized into a user, or assistant (bot) role. The main purpose of this function is to provide some language context to the gpt API. The System role tells GPT what its purpose is and how it should think when responding.

Component 2. Journal Entry Feature

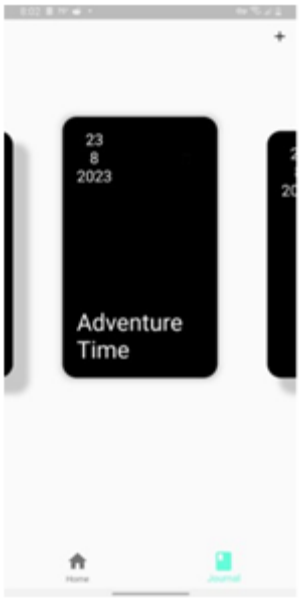

Figure 7. Journal view

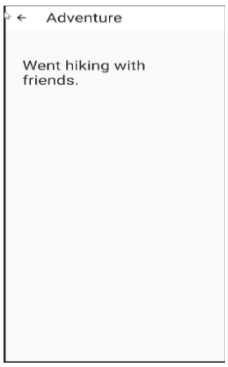

Figure 8. Journal editor

The journal entry feature allows users to journal throughout the day. When a journal entry is saved, its sentiment score is calculated, and its data object is sent to firebase [9]. The Journal page features basic cards with a sliding animation to view all journal entries.The journal view page features a simple UI to view and edit both the title and content of the entry. The data is saved to Firebase in real-time as it is edited, and a sentiment score is calculated instantly as well.

Figure 9. Pageview builder for the Journal slides

```dart
class Journal {
  late String uid;
  String title;
  String body;
  bool isFavorite = false;
  double sentimentScore = 0.0;

  final DateTime createdAt;

  Journal(
      {this.uid = "",
      required this.title,
      required this.body,
      required this.createdAt,
      this.sentimentScore = 0.0,
      isFavorite});

  factory Journal.fromJson(Map<String, dynamic> json) {
    return Journal(
      uid: json["uid"].toString(),
      title: json['title'],
      body: json['body'],
      sentimentScore: json['sentimentScore'],
      createdAt: DateTime.parse(json['createdAt']),
      isFavorite: false,
    );
```

Figure 10. Data model for a journal entry

Figure 11. GetJournal method to fetch a list of all journal entries

The code in Fig 10 renders the journal entry cards and builds a filter card. The journal page works by taking the user to the journal form when a button is pressed. In the form, the user can proceed to type his/her journal. Once the user is done with writing, a function takes in what was typed and calculates a sentiment score. The Journal class in Figure 11 contains some internal functions to convert the data types into types appropriate for storage in Firebase. This includes the Json format, which is a map of strings to dynamic data types. The getJournal function contains optional parameters to define whether the query should be sorted by date or not.

Component 3. Mood check-in system

The mood check-in is also a substantial feature of the app. The user has the option to do one mood check-in per day. This data is fed to GPT for context. The mood check in screen features two pages. One for the mood, and the other to detect the possible cause of said mood. The user is offered eight possible mood choices (Fig 12) , and 4 reason choices to start. Once the user is done completing both pages, they are allowed to click on a button that completes the mood check in for that day. Additionally they are able to view a calendar of all of their previous mood check-ins.

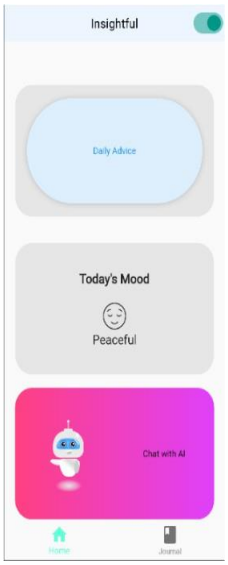

Figure 12. Homepage

Figure 13. Mood selection screen

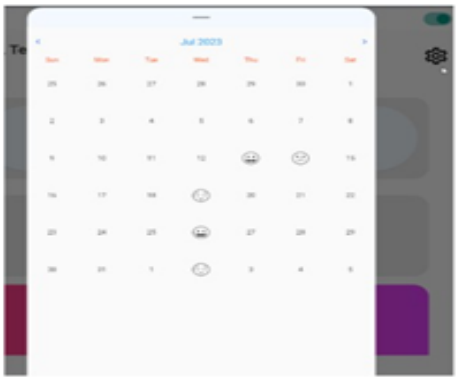

Figure 14. Calendar Syste

```
Map<String, String> reasons = {
  "school": "School",
  "briefcase": "Work",
  "dna": "Life",
  "family": "Family",
};

class Mood {
  Mood({this.mood, this.reason, this.checkedIn});

  String? mood;
  String? reason;
  Timestamp? checkedIn;

  Map<String, dynamic> toJson() {
    return {
      "mood": mood,
      "reason": reason,
      "checkedIn": checkedIn,
    };
  }

  factory Mood.fromJson(Map<String, dynamic> json) {
    return Mood(
```

Figure 15. Mood Data Model

Figure 16. _saveStateToFireStore() function to save mood data

In the first image, a list of moods and responses is created. The key for a mood is its corresponding emoji name. The mood class stores the mood, reason, and check-in date for a mood check in.

The saveStateToFirestore function removes the time from the date and leaves only the day, month, and year values to make daily check-in querying easier. The function then converts the mood object to a json object and uploads that data to Firebase Firestore.

The last image (Fig 16) is a snippet for the code that creates a grid view of moods with their rendered emojis. The code also creates a page for the mood reasons grid.

MoodPrism is a self-monitoring mood-tracking app with user alerts [3]. It has most of the features of the app developed in this paper, except for a GPT based chatbot section and a recommendation system. Insightful also uses the web to store data, while MoodPrism stores information offline. This can have benefits; for instance, being able to write a journal entry offline, or on a roadtrip with little to no service. The benefit of having insightful work online, is that all data is synced at entry-time, and access to the AI engine is consistent. CopeSmart is an app developed by Rachel Kenny and her team of researchers to assess the feasibility of telemental health apps as mediums for promoting positive mental health [5]. Although Copesmart exhibits all of the features of a telemental health app, its main problem is that it does not look appealing to users. Copesmart was created as a research tool and not much consideration was put into the UI and user experience. Pelin Karaturhan, author of Combining Momentary and Retrospective Self-Reflection in a Mobile Photo-Based Journaling Application and her team of researchers, developed a mobile photo based journaling application [4]. Unlike Insightful or any of the other apps mentioned, this app is solely photo based. Users create journal entries using photos and add descriptive text to them. Users are also asked to score the photos based on intellectual and emotional significance. Insightful does not make use of videos or images, and only text as a medium of storage.

4. EXPERIMENT

Throughout the app design process, it was apparent that the GPT model needed a good prompt to generate good data.

One thing we considered testing was the API's ability to generate a unique selection of options for the recommendation system.

4.1. Experiment 1

The first change to the code with respect to the experiment was to change the prompt that GPT used to generate recommendations and provide more unique outputs. A random seed was provided in the prompt to "trick" GPT into having a different input statement. The goal of this was to generate as unique an activity recommendation as possible using the emotional average score, and the randomization seed.

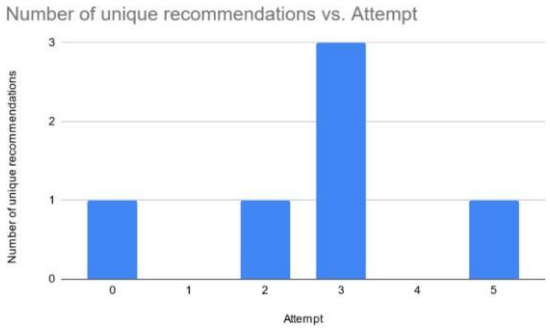

Figure 17. Number of unique recommendations to attempt number

The mean was calculated to be 1 unique recommendation and the median was calculated to be 1 unique recommendation. The highest unique recommendation value was 3, out of 3 possible recommendation cards. The number of recommendation cards generated has the largest effect on the number of unique recommendations generated by the AI. Insightful generates at most 3 recommendations by default before prompting for a new set of data.

4.2. Experiment 2

One main concern with the AI was its ability to understand that sentiment scores could be rated from good to bad and how well it would return the right response in accordance to those sentiment scores. The Sentiment is calculated by either taking an average of all journal entry scores, or summing their respective scores to obtain a balance. This is fed to gpt for decision making. To calculate the sentiment values, we used a library called dart_sentiment, based on the AFINN-165 dataset.

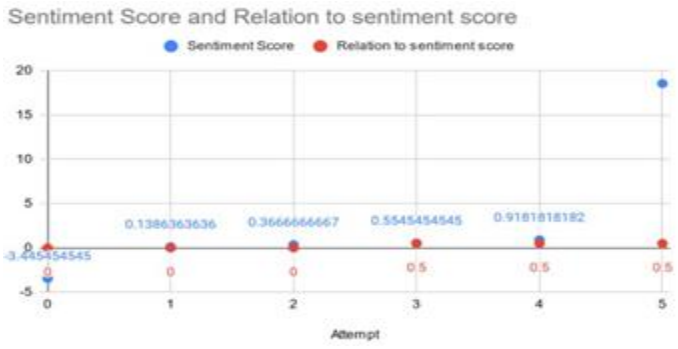

Figure 18. Sentiment score and relation to sentiment score

The red line indicates one of three relation values: low, medium, or high.

We use these three values to rate the relevance of GPT's output to the average sentiment score. Much like rating something on a scale of 1-10, it was more convenient to use a relevance scale of zero to one. 0 corresponds to low, 0.5 to medium, and 1 to high. Relation refers to how appropriate a response was according to a sentiment value. Negative sentiment values usually indicate negative life occurrences. One would expect the bot to respond with The data point with the highest sentiment score was an extremely positive journal entry. All data points appeared an equal number of times. The average sentiment score from the experimental dataset was 2.85, and the average relational score was 0.25, which shows the non-uniqueness of the data returned. The results of this experiment show that while the GPT pipeline shows some understanding of its context, it does not generate data consistent enough to that context, regardless of the sentiment score. A solution could be to switch the model being used, or generate the recommendations at a slower pace.

In order to test the efficiency of the GPT API in generating meaningful responses, it was necessary to observe how changes to the prompt used for the request affected the output of GPT's web response. The findings of that experiment were not so helpful because the API generated mostly the same information. This was not the case with the actual chat feature of the app, since unique responses were generated in that case. Out of 3 possible advice cards, the highest number obtained in a continuous sequence of 5 experimental attempts was three unique responses, but this only occurred once. In the second experiment, we wanted to see if the sentiment score had an effect on the topic of the responses generated by the recommendation system.

5. RELATED WORK

Of the three mental health apps mentioned in Bakker's paper A randomized controlled trial of three smartphone apps for enhancing public mental health, MoodPrism was the first. MoodPrism is a self-monitoring mood-tracking app that prompts users to report their emotional state daily via a short survey [3]. Unlike the app being developed in this paper, MoodPrism features preventative and stepped-care support. However, the number of publicly available MHapps that make claims of mental health improvements far outweighs the number of MHapps that have supporting evidence and robust experimental research studies [3]. Nevertheless, according to Bakker, the use of the app, among other different MH apps, greatly decreased depression and anxiety, and improved mental wellbeing.

The apps described in the article make no use of AI for mental health analysis. however. In Insightful, daily emotional state surveys are fed to a sentiment analysis engine for AI feedback Rachel Kenny author of, Feasibility of 'CopeSmart': A Telemental Health App for Adolescents. [5], and her team, developed an app named CopeSmart which creates check in alerts at 8 pm each evening, an ideal time for adolescents and young people to engage with the app [5]. CopeSmart has a mood check-in feature to track mood fluctuation daily, and a graph to display that data. It also implements a calendar to view mood check-in history. Insightful is much like CopeSmart, with the only feature difference being sentiment analysis and an AI Chatbot.

The following section references Combining Momentary and Retrospective Self-Reflection in a Mobile Photo-Based Journaling Application by Pelin Karaturhan.

In Pelin's article, researchers developed a mobile photo based journaling application. Users add photos daily and write reflections in relation to those photos. Users are also asked to score the photos based on intellectual and emotional significance. Their findings indicate that users may not be able to motivate themselves and see the value of recording daily experiences until they look at their input retrospectively [4]. Unlike Insightful, the researchers proposed solution did not utilize artificial intelligence. Insightful also makes use of daily mood check ins to allow users to track

their mental health progression through a calendar.

6. CONCLUSIONS

Although the app was a success in that it functioned well as a self-help therapist and a mental health tracker, it was missing some vital features that would have improved the user experience [14]. Firstly, it is missing a statistics page to view information like all mood check-ins and the number of journal entries. The app is also missing a sorting feature needed to arrange journal entries. Overall, the journal entry page UI could be better. It is currently not possible to add videos or images to journal entries yet, and the journal entry edit page could have been more polished. Larger buttons and animations could've been used to provide a more interactive user experience as well.

The prompts used to generate advice and recommendations in the app could've been better. The ability to save multiple AI chats could also have been included [8]. Although the app has a number of limitations, it still accomplishes its goal of being able to provide users with a psychological outlet, a way to track their moods over time, and a journal that takes their mood into account.

REFERENCES

[1] Boud, David. "Using journal writing to enhance reflective practice." New directions for adult and continuing education 2001.90 (2001): 9-18.
[2] Minerva, Francesca, and Alberto Giubilini. "Is AI the Future of Mental Healthcare?." Topoi (2023): 1-9.
[3] Bakker, David, et al. "A randomized controlled trial of three smartphone apps for enhancing public mental health." Behaviour Research and Therapy 109 (2018): 75-83.
[4] Karaturhan, Pelin, et al. "Combining Momentary and Retrospective Self-Reflection in a Mobile Photo-Based Journaling Application." Nordic Human-Computer Interaction Conference. 2022.
[5] Kenny, Rachel, Barbara Dooley, and Amanda Fitzgerald. "Feasibility of" CopeSmart": a telemental health app for adolescents." JMIR mental health 2.3 (2015): e4370.
[6] Leventhal, Allan M. "Sadness, depression, and avoidance behavior." Behavior modification 32.6 (2008): 759-779.
[7] Transformer, ChatGPT Generative Pre-trained, and Alex Zhavoronkov. "Rapamycin in the context of Pascal' s Wager: generative pre-trained transformer perspective." Oncoscience 9 (2022): 82.
[8] Argyle, Lisa P., et al. "AI Chat Assistants can Improve Conversations about Divisive Topics." arXiv preprint arXiv:2302.07268 (2023).
[9] Moroney, Laurence, and Laurence Moroney. "The firebase realtime database." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform (2017): 51-71.
[10] Aydın, Ömer, and Enis Karaarslan. "OpenAI ChatGPT generated literature review: Digital twin in healthcare." Available at SSRN 4308687 (2022).
[11] Jiang, Fan, and Shaoping Ku. "How to display the data from database by ListView on Android." 2010 2nd International Workshop on Intelligent Systems and Applications. IEEE, 2010.
[12] Rivas, Pablo, and Liang Zhao. "Marketing with chatgpt: Navigating the ethical terrain of gpt-based chatbot technology." AI 4.2 (2023): 375-384.
[13] Backes, Michael, Sven Bugiel, and Erik Derr. "Reliable third-party library detection in android and its security applications." Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016.
[14] Mataix-Cols, David, and Isaac M. Marks. "Self-help with minimal therapist contact for obsessive –
[15] compulsive disorder: a review." European Psychiatry 21.2 (2006): 75-80.
[16] Millstein, Frank. Natural language processing with python: natural language processing using NLTK. Frank Millstein, 2020.