CS5346 ADVANCED ARTIFICIAL INTELLIGENCE
PROJECT – 1 FALL 2023

INTELLIGENT EXPERT SYSTEM FOR DIAGNOSING MENTAL ILLNESSES AND TREATMENT

Submitted by
Kavitha Lodagala (A05252367)

Team members:
Roopika Ganesh
Sakshi Tripathi

Under the guidance of
Dr. Moonis Ali

CONTENTS

LIST OF FIGURES

LIST OF SCREENS

1. PROBLEM DESCRIPTION

1.1 Description of the problem:

Mental health is an important element of general well-being, and persons with mental health difficulties require correct diagnosis and effective treatment. To meet this important need, we proposed an intelligent expert system for a mental health clinic.

Mental health problems affect a large proportion of the people in today's globe. Timely diagnosis and appropriate treatment are critical for improving the lives of those suffering from various mental diseases. The aim is to develop a comprehensive expert system that can diagnose a variety of mental diseases and offer appropriate therapies depending on the diagnosis.

The major goal is to create and deploy an intelligent expert system capable of diagnosing 19 illnesses. Following an accurate diagnosis of a mental condition, the system should suggest relevant treatment alternatives. These recommendations should be built on reliable medical information.

To obtain thorough information regarding the symptoms, diagnostic criteria, and treatment modalities related to the specified mental diseases, extensive research will be conducted using web resources and other trusted sources. The system will allow mental health clinic workers to easily input patient symptoms, easing the diagnosing procedure. The expert system will analyze patient symptoms using Backward Chaining to diagnose mental diseases. A Backward Chaining decision tree will be created, which will capture the logical flow for diagnosis and translate it into a collection of rules. Forward Chaining will be used to recommend relevant treatments for the diagnosed disease. Then a decision tree will be created, which will capture the logical flow for treatment recommendations. These decision trees will subsequently be converted into rules. Variables describing symptoms, diagnostic criteria, and treatment alternatives will be included in these rules.

Once an AI expert system is put into place, it should be carefully tested with patient cases to verify its accuracy in making diagnoses and recommending treatments. The eventual result of this research will be an intelligent expert system that can help mental health clinics diagnose a range of mental diseases and suggest suitable therapies. The accuracy and effectiveness of mental health diagnoses might be greatly enhanced by this system, which would also improve patient outcomes.

1.2 Objective of Project:

Our main objective is to create an expert system that relies on a knowledge base containing rules and information. When a user interacts with the system, they will be initially asked about their health status. If the user reports being unwell, they will be prompted with a series of questions regarding their symptoms. By processing the user's responses using a backward chaining algorithm, the system will use its knowledge base to identify the specific type of disorder the user might have. Subsequently, this detected disorder will serve as input for a

forward chaining algorithm, which will then determine an appropriate course of treatment. Ultimately, this approach aims to enhance the quality of healthcare by providing more accurate and tailored treatment recommendations for the patient.

2. Domain

2.1 The significance of diagnosing a mental illness:

The importance of diagnosing a mental disorder is critical to both individual and societal well-being. Accurate diagnosis is the foundation for appropriate treatment and support, which ultimately relieves pain, prevents escalation, and promotes better mental health outcomes.

Because mental health issues are complex and can show in a wide range of symptoms, accurate diagnosis is critical. Early detection and diagnosis allow patients to receive timely interventions and treatment regimens that are tailored to their personal requirements. This not only lessens the severity and duration of discomfort, but it also has the potential to prevent illnesses from worsening and becoming more complex.

Furthermore, mental disorder diagnosis is critical for suicide prevention. Individuals at danger of self-harm or suicide can be identified using mental health diagnoses, allowing for early intervention and potentially life-saving support.

Accurate diagnosis provides patients and their families with information about the nature of the ailment, its causes, and the potential consequences. This understanding can help to alleviate fear and stigma, paving the path for informed decisions and helpful relationships.

AI expert systems are already providing significant benefits in the diagnosis and treatment of mental diseases. Backward chaining is utilized to detect the condition that the patient is suffering from; the diagnosed disorder is then sent as input to forward chaining, which provides therapy recommendations to the patient.

Mental disorders used in Project are –
Our project can diagnose 19 disorders and their symptoms. The disorder list and their symptoms are -

1. **Bipolar Disorder**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, hyperactivity
2. **Schizophrenia**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, delusion disorganized thinking, lack of motivation, amnesia, incoherent speech, excitability.
3. **Schizoaffective Disorder**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, delusion disorganized thinking, lack of motivation, hopelessness, grandiosity.
4. **Major Depressive Disorder**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, sleeplessness, slowed thinking

5. **Panic Disorder with Agoraphobia**
   Symptoms- irritability, unfocused, restlessness, fatigue, chest pain, dizziness, sweating, nausea, helplessness, fear being alone
6. **Dissociative Identity Disorder**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, delusion disorganized thinking, lack of motivation, amnesia, identity confusion, blackout
7. **Dysthymia**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, poor appetite, or overeating
8. **Generalized Anxiety Disorder**
   Symptoms- irritability, unfocused, restlessness, fatigue
9. **Dementia**
   Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, mental confusion, paranoia, mental disorientation, mental decline, lack of restraint, nervousness, jumbled speech
10. **Post-Traumatic Stress Disorder**
    Symptoms- irritability, unfocused, restlessness, nightmares, intense memories of trauma
11. **Obsessive-Compulsive Disorder**
    Symptoms- irritability, unfocused, restlessness, mood swings, social isolation, agitation, impulsivity, compulsive behavior, hypervigilance, ritualistic behavior, repetitive behavior
12. **Psychosis**
    Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, delusion disorganized thinking, lack of motivation, hostility
13. **Body Dismorphic Disorder**
    Symptoms- irritability, unfocused, restlessness, fatigue, weight fluctuation, lack of confidence, thinking about body, changing appearance often, constantly picking skin
14. **Insomnia**
    Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, sleeplessness, headache.
15. **Narcolepsy**
    Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, suicidal thoughts, sleeplessness, loss of muscle, cataplexy,
    Sleep paralysis
16. **Borderline Personality Disorder**
    Symptoms- irritability, unfocused, restlessness, mood swings, social isolation, boredom, distorted self-image, emptiness, loss of interest in daily activities
17. **Alzheimer's**
    Symptoms- irritability, unfocused, restlessness, sadness, anger, hallucination, mental confusion, paranoia, mental disorientation, mental decline, lack of restraint, nervousness, suspicious
18. **Bulimia**
    Symptoms- irritability, unfocused, restlessness, fatigue, weight fluctuation, lack of confidence, thinking about body, repeatedly eating large amount of food, vomit after eating, extreme fasting, laxative after eating

19. Kleptomania

Symptoms- irritability, unfocused, restlessness, sadness, anger, urge to steal items, pleasure after stealing items, guilt

2.2 AI Expert System

An Artificial Intelligence Expert system is a computer program or software application that simulates a human expert's decision-making and problem-solving abilities in a certain topic or field. These expert systems are designed to capture and duplicate the knowledge, reasoning, and expertise of human specialists in a certain field.

Expert systems play an important role in a variety of industries and applications, improving decision-making, increasing efficiency, lowering costs, and providing access to specialized expertise. Their significance is especially clear in domains where problem complexity and the necessity for dependable and consistent decision assistance are critical.

The following are the components of AI expert systems: 1. Knowledge base
2. Inference Engine 3. Interface

2.2.1 Knowledge Base

The knowledge base is an important component that greatly adds to the system's ability to reason, make decisions, and provide expert-like responses. The knowledge base is a repository for domain-specific information, facts, rules, and heuristics used by the system to replicate the knowledge and competence of a human expert in a particular field. Here are some of the most important components of AI expert systems' knowledge bases.

The expert system's core is the knowledge base, which contains a huge repository of domain-specific information, facts, rules, and heuristics. This knowledge is frequently collected from human expertise in the topic and is organized methodically.

2.2.2 Inference Engine

The inference engine is a critical component that does reasoning and draws conclusions based on knowledge and rules in the system's knowledge base. It is crucial in imitating the problem-solving abilities and decision-making processes of a human expert.

The inference engine relies heavily on rule-based reasoning to process the knowledge and rules in the knowledge base. These rules are often written as “if-then” statements, where the “if” section (antecedent) explains a condition or set of conditions and the “then” section (consequent) specifies an action or conclusion. The inference engine analyses these rules based on the available information to determine which actions or conclusions to take.

2.2.3 Interface

The user interface (UI) or the means through which users interact with the system is sometimes referred to as an interface. The interface facilitates communication between users and the expert system by allowing users to input queries, get results, and interact with the system's features.

3. Methodologies

3.1 Backward Chaining

Backward chaining is a method used in artificial intelligence expert systems to achieve a specific goal or conclusion by working backward from the desired outcome to find the facts that must be met in order to achieve that goal. This strategy is commonly utilized in expert diagnostic systems and goal-driven problem-solving scenarios.

The backward chaining process begins with the definition of a specific goal or desired outcome. This goal represents what the expert system is trying to achieve or determine. For example, in a medical diagnosis expert system, the goal might be to identify the underlying cause of a patient's symptoms. Once the goal is defined, the system works backward from the goal to identify the conditions or facts that must be true for the goal to be met. These conditions are often represented as "if-then" rules or relationships in the knowledge base of the expert system. The system evaluates the rules and knowledge in its knowledge base to determine which rules are relevant to the current goal. It identifies rules whose consequents (the "then" part) match the goal.

The data structures used in Backward chaining are:

1.Conclusion List:

In this list all the variables which are included in then part of rules and the corresponding rule numbers are stored. In backward chaining the goal variable is searched in this list and corresponding rule is fetched.

2.Variable List:

All variables that appear in the if condition but not in the then section are stored in this list, along with their corresponding user-supplied value. However, at first, no variable is instantiated.

3.Clause Variable list:

All variables in the if condition is saved, and a clause number is assigned. The size of this list is determined by the maximum number of variables included in the IF condition from all rules. This list contains the variables of a rule, with any remaining spaces left blank.

The formula can be used to compute the clause number for a given rule number:

If rule numbers are in the pattern 1,2,3,4...
Clause number = 10 * (Rulenumber-1) + 1

If the rule numbers are in the following format: 10, 20, 30, 40...
Clause number = 10 * ((Rule number/10)-1) + 1

Here, each rule in the clause variable list is allocated 10 spaces.

4.Conclusion Stack:

This stack is important in the backward chaining algorithm. This variable maintains the specifics of the currently executing rule, such as the rule number and clause number tells which variable must be instantiated in order to execute the related rule.

3.2 Forward Chaining

Forward chaining is an inference approach used in AI expert systems to draw conclusions, make decisions, and achieve goals by beginning with available facts and applying rules or knowledge repeatedly to derive additional information. The primary goal of the system is to continuously acquire knowledge and achieve conclusions.

The procedure starts with a set of known facts, data, or conditions. These facts are usually offered as inputs or as the result of backward chaining. The expert system then evaluates rules that are consistent with the existing state of facts. Once satisfied, the system executes then part to generate new conclusions or facts. These conclusions are saved in the system's memory. After adding new facts or conclusions to the knowledge base, the system reevaluates the rules to determine whether any more rules are satisfied or not. This method is repeated until no more rules can be implemented or a certain goal is reached. Forward chaining is frequently used to attain certain aims or reach desired conclusions.

The data structures used in Backward chaining are:

1.Clause Variable list:

All variables in the if condition is stored, and a clause number is assigned. The size of this list is determined by the maximum number of variables included in the IF condition from all rules. This list contains the variables of a rule, with any remaining spaces left blank.

The formula can be used to compute the clause number for a given rule number:
If rule numbers are in the pattern 1,2,3,4...
Rule number = {(Quotient (clause number/3))} +1)
If the rule numbers are in the following format: 10, 20, 30, 40...
Rule number = {(Quotient (clause number/3))} +1) * 10

Here, each rule in the clause variable list is allocated 4 spaces.

2.Conclusion Variable queue:

The most significant part of the forward chain implementation is this data structure. This specifies which IF-THEN sentence produces the intended result, as well as which clause in the IF section is being examined.

3.Variable list:

All variables that appear in the if condition but not in the then section are stored in this list, along with their corresponding user-supplied value. However, at first, no variable is instantiated.

4.Clause Variable pointer:

This keeps track of the rule number and clause number of the current IF-THEN condition under consideration.

3.3 Existing System

The use of conventional or established ways of decision-making and problem-solving. Human expertise, manual analysis, and defined rules or processes are frequently used in these approaches. Without the assistance of automated technologies, human specialists make decisions based on their knowledge, experience, and judgment. While these traditional approaches have been used successfully for many years, they do have some drawbacks and limits.

3.3.1 Disadvantages of Existing System

Traditional techniques to decision-making and problem-solving have limits, such as subjectivity, scalability concerns, difficulty in dealing with complicated cases, and communication and expertise availability challenges.

3.4 Proposed System

AI expert systems use their knowledge base, inference engine, and reasoning ability to solve complicated issues and decision-making scenarios. It has a knowledge base that is structured and comprises domain-specific information, facts, rules, and heuristics. This knowledge is gathered from field experts and arranged for easy retrieval and processing.

When a user enters a query, problem, or group of observations into the system, the process begins. To arrive at conclusions or suggestions, the inference engine evaluates user input and knowledge contained in the knowledge base. AI expert systems frequently employ rule-based reasoning, in which rules are stated in an “if-then” manner. The inference engine compares the input to applicable rules and performs the actions defined in those rules.

4. DECISION TREE

4.1 Introduction to decision tree

A decision tree is a graphical representation of decision rules and their potential outcomes. A tree-like concept is used to structure decision trees. There are various types of nodes in this tree.

1. Decision node: These are shaped like an oval. This is a decision node in which the user is posed a question and must respond with either a 'yes' or a 'no'.
2. Intermediate and final nodes: These are represented by rectangles. We can only have one incoming branch that is either yes or no, but not both. We can have any number of yes outgoing branches for the intermediate node.

4.2 Backward Chaining

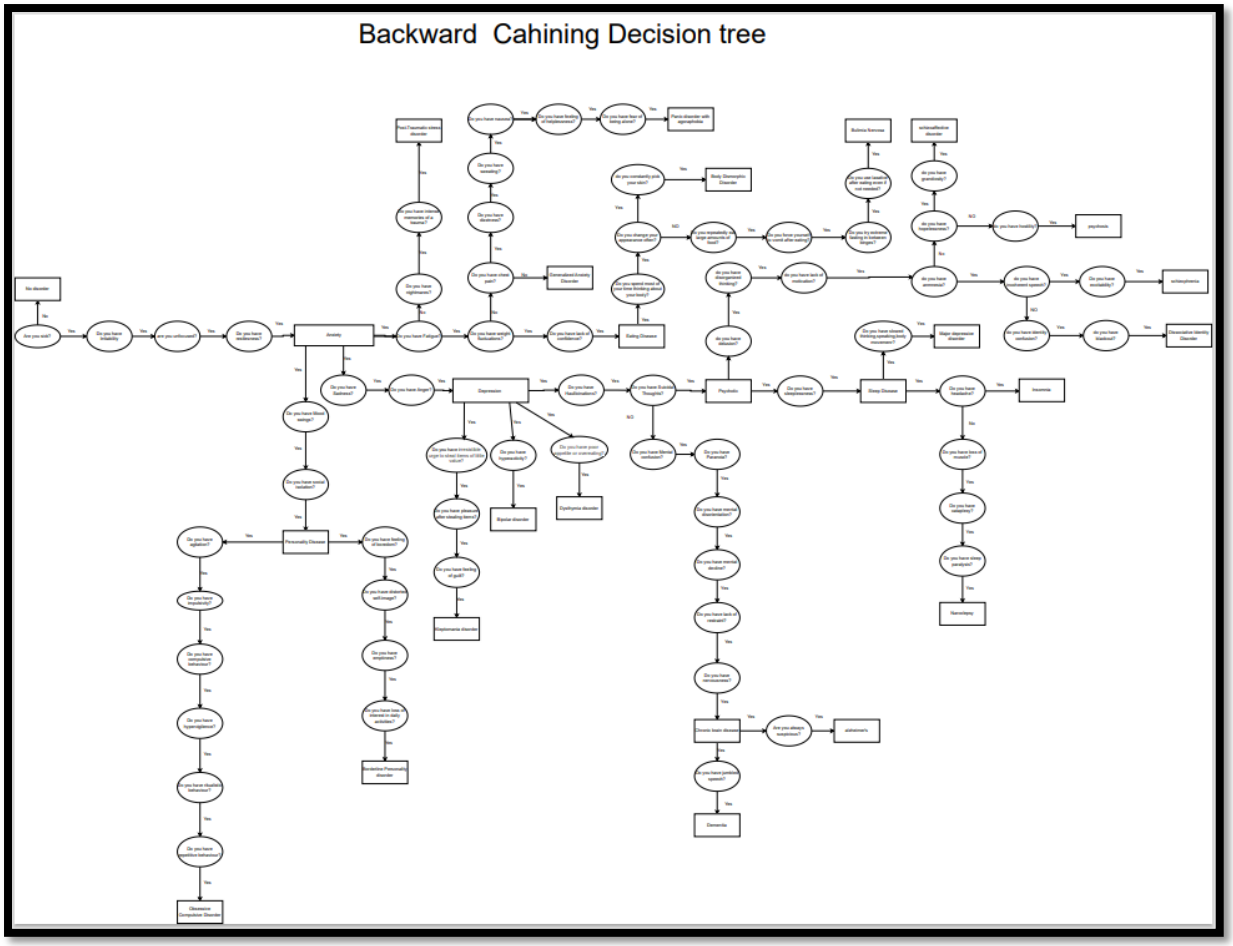


Fig. 1 Backward chaining decision tree

4.3 Forward Chaining

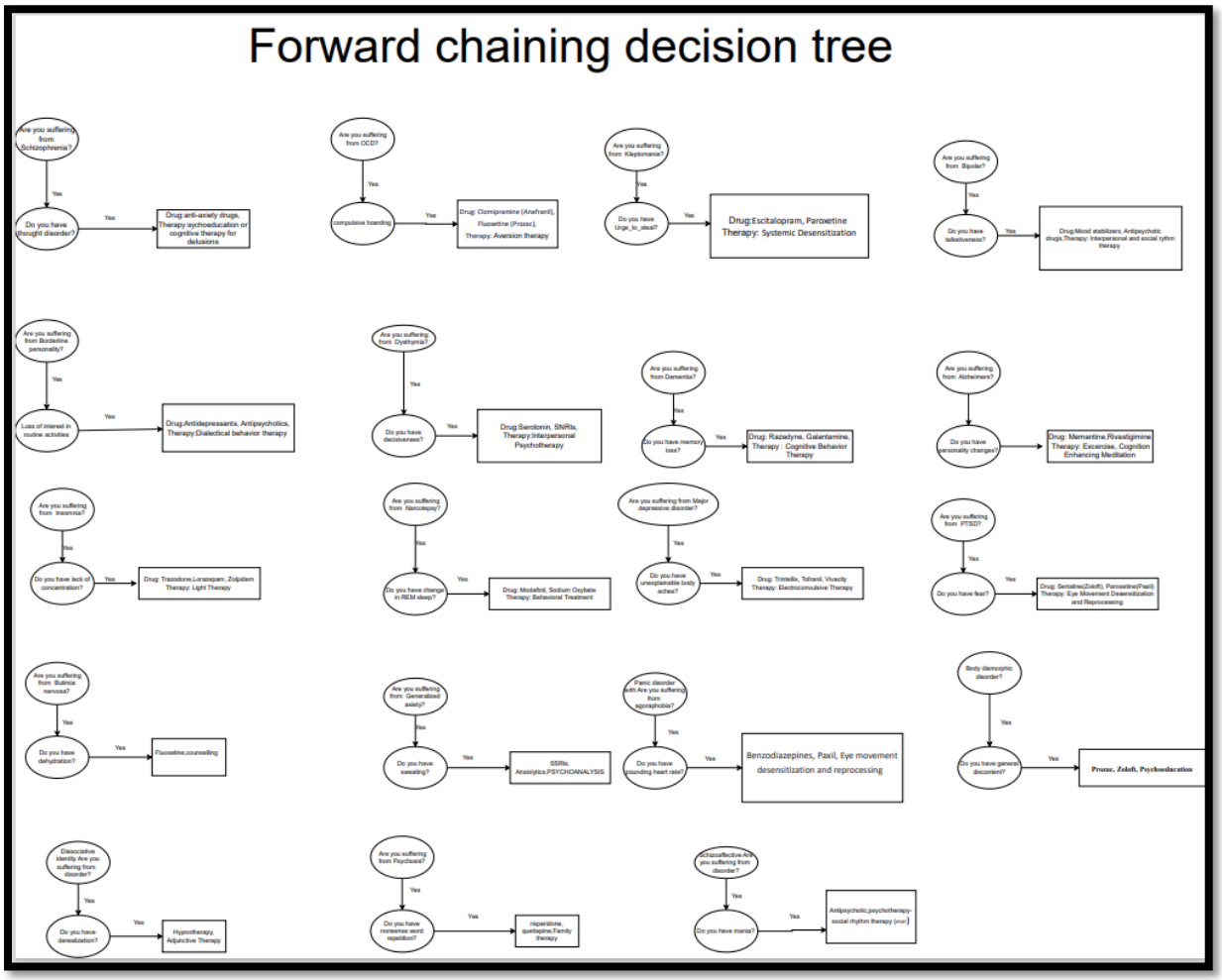


Fig. 2 Forward chaining decision tree

5. RULES

A Rule is a fundamental component used to represent knowledge and encode the decision-making or problem-solving logic of the system. Rules are structured as "if-then" statements and are a critical part of the knowledge base.

5.1 Backward Chaining Rules:

Inference rules (BACKWARD CHAINING RULES)

1. IF SICK=NO
   THEN DISORDER=NO
2. IF SICK=YES AND IRRITABILITY=YES AND UNFOCUSED=YES AND RESTLESSNESS=YES
   THEN ANXIETY=YES
3. IF ANXIETY=YES AND MOODSWING=YES AND SOCIAL_ISOLATION=YES
   THEN PERSONALITY_DISEASE=YES

4. IF PERSONALITY_DISEASE=YES AND AGITATION=YES AND IMPULSIVITY=YES AND COMPULSIVE_BEHAVIOUR=YES AND HYPERVIGILANCE=YES AND RITUALISTIC_BEHAVIOUR =YES AND REPETITIVE_BEHAVIOUR=YES
THEN DISORDER=OCD

5. IF PERSONALITY_DISEASE=YES AND BOREDOM=YES AND DISTORTED_SELF_IMG=YES AND EMPTINESS=YES AND LOSS_OF_INTEREST=YES
THEN DISORDER= BORDERLINE_PERSONALITY_DISORDER

6. IF ANXIETY=YES AND SADNESS=YES AND ANGRY=YES
THEN DEPRESSION=YES

7. IF DEPRESSION=YES AND STEALING=YES AND STEALING_PLEASURE =YES AND GUILT=YES
THEN DISORDER=KLEPTOMANIA

8. IF DEPRESSION=YES AND HYPERACTIVITY=YES
THEN DISORDER= BIPOLAR_DISORDER

9. IF DEPRESSION=YES AND POOR_APPETITE=YES
THEN DISORDER=DYSTHYMIA

10. IF DEPRESSION=YES AND HALLUCINATION=YES AND SUICIDAL_THOUGHT=YES
THEN PYSCHOTIC=YES

11. IF DEPRESSION=YES AND HALLUCINATION=YES AND SUICIDAL_THOUGHT=NO AND MENTAL_CONFUSION=YES AND PARANOIA=YES AND MENTAL_DISORIENTATION=YES AND MENTAL_DECLINE=YES AND LACK_OF_RESTRAINT=YES AND NERVOUSNESS=YES
THEN CHRONIC_BRAIN_DISEASE=YES

12. IF CHRONIC_BRAIN_DISEASE=YES AND JUMBLED_SPEECH=YES
THEN DISORDER=DEMENTIA

13. IF CHRONIC_BRAIN_DISEASE=YES AND SUSPICIOUS=YES
THEN DISORDER=ALZHEIMER

14. IF PSYCHOTIC=YES AND SLEEPLESSNESS=YES
THEN SLEEP_DISEASE=YES

15. IF SLEEP_DISEASE=YES AND HEADACHE=YES
THEN DISORDER=INSOMNIA

16. IF SLEEP_DISEASE=YES AND HEADACHE=NO AND LOSS_OF_MUSCLE=YES AND CATAPLEXY=YES AND SLEEP_PARALYSIS=YES
THEN DISORDER=NARCOLEPSY

17. IF SLEEP_DISEASE=YES AND SLOWTHINKING=YES
THEN DISORDER=MAJOR_DEPRESSIVE_DISORDER

18. IF ANXIETY=YES AND FATIGUE=YES AND WEIGHTFLUCTUATION=YES AND LACK_OF_CONFIDENCE=YES
THEN EATING_DISEASE=YES

19. IF ANXIETY=YES AND FATIGUE=NO AND NIGHTMARES=YES AND TRAUMA_MEMORIES=YES
THEN DISORDER=PTSD

20. IF ANXIETY=YES AND FATIGUE=YES AND WEIGHTFLUCTUATION=NO AND CHESTPAIN=NO
THEN DISORDER=GENERALIZED_ANXIETY

21. IF ANXIETY=YES AND FATIGUE=YES AND WEIGHTFLUCTUATION=NO AND CHESTPAIN=YES AND DIZZINESS=YES AND SWEATING=YES AND NAUSEA=YES AND HELPLESSNESS=YES AND FEAR_OF_BEING_ALONE=YES
THEN DISORDER=PANIC_DISORDER_WITH_AGORAPHOBIA

22. IF EATING_DISEASE=YES AND BODY_THINKING=YES AND APPEARANCE_CHANGE=YES AND PICKY_SKIN=YES
THEN DISORDER=BODY_DISMORPHIC_DISORDER

23. IF EATING_DISEASE=YES AND BODY_THINKING=YES AND APPEARANCE_CHANGE=NO AND BINGE_EATING=YES AND VOMITING=YES AND FASTING=YES AND LAXATIVE_USE=YES
THEN DISORDER=BULIMIA_NERVOSA
24. IF PSYCHOTIC=YES AND DELUSION=YES AND DISORGANIZED_THINKING=YES AND LACK_OF_MOTIVATION=YES AND AMNESIA=YES AND INHCOHERENT_SPEECH=YES AND EXCITABILITY=YES
THEN DISORDER=SCHIZOPHRENIA
25. IF PSYCHOTIC=YES AND DELUSION=YES AND DISORGANIZED_THINKING=YES AND LACK_OF_MOTIVATION=YES AND AMNESIA=YES AND INCOHERENT_SPEECH=NO AND COMPULSION=YES AND BLACKOUT=YES
THEN DISORDER=DISSOCIATIVE_IDENTITY_DISORDER
26. IF PSYCHOTIC=YES AND DELUSION=YES AND DISORGANIZED_THINKING=YES AND LACK_OF_MOTIVATION=YES AND AMNESIA=NO AND HOPELESSNESS=NO AND HOSTILITY=YES
THEN DISORDER=PSYCHOSIS
27. IF PSYCHOTIC=YES AND DELUSION=YES AND DISORGANIZED_THINKING=YES AND LACK_OF_MOTIVATION=YES AND AMNESIA=NO AND HOPELESSNESS=YES AND GRANDIOSITY=YES
THEN DISORDER=SCHIZOAFFECTIVE_DISORDER

5.2 Forward Chaining Rules

Inference Rules for Forward Chaining:

1. IF DISORDER = SCHIZOPHRENIA AND THOUGHT_DISORDER=YES
THEN TREATMENT=ANTI_ANXIETY_DRUG, PSYCHOEDUCATION
2. IF DISORDER=OCD AND COMPULSIVE_H OARDING=YES
THEN TREATMENT=CLOMIPRAMINE, FLUOXETINE, AVERSION_THERAPY
3. IF DISORDER=KLEPTOMANIA AND URGE_TO_STEAL=YES
THEN TREATMENT=ESCITALOPRAM, PAROXETINE, SYSTEMIC_DESENSITIZATION
4. IF DISORDER=BIPOLAR AND TALKATIVENESS=YES
THEN TREATMENT = MOOD_STABILIZERS, ANTIPSYCHOTIC_DRUGS, SOCIAL_RYTHM_THERAPY
5. IF DISORDER=BORDERLINE_PERSONALITY AND LOSS_OF_INTEREST=YES
THEN TREATMENT=ANTIPRESSANTS, ANTIPSYCHOTICS, DIALECTICAL_BEHAVIOR_THERAPY
6. IF DISORDER=DYSTHYMIA AND DECISIVENESS=YES
THEN TREATMENT=SEROTONIN, SNRIs, INTERPERSONAL_PSYCHOTHERAPY
7. IF DISORDER = DEMENTIA AND MEMORY_LOSS = YES
THEN TREATMENT = RAZADYNE, GALANAMINE, COGNITIVE BEHAVIOUR THERAPY
8. IF DISORDER = ALZHEIMERS AND PERSONALITY_CHANGES = YES
THEN TREATMENT = MEMANTINE, RIVASTIGIMINE, EXCERCISE, COGNITION ENHANCING MEDITATION
9. IF DISORDER = INSOMNIA AND LACK_OF_CONCENTRATION = YES
THEN TREATMENT = TRAZODONE, LORAZEPEN, ZOLPIDEIM, LIGHT THERAPY
10. IF DISORDER = NARCOLEPSY AND CHANGE_IN_REM_SLEEP = YES
THEN TREATMENT = MODAFINIL, SODIUM OXYBATE, BEHAVIOURAL TREATMENT
11. IF DISORDER = MAJOR_DEPRESSIVE_SIDORDER AND UNEXPLAINABLE_BODY_ACHES = YES
THEN TREATMENT = TRINTELLIX, TOFRANIL, VIVACITY, ELECTROCONVULSIVE THERAPY
12. IF DISORDER = POST_TRAUMATIC_STRESS_DISORDER AND FEAR = YES
THEN TREATMENT = SERATINE (ZOLOFT, PAROXENTINE(PAXIL), EYE MOVEMENT D ESENSITIZATION AND REPROCESSING
13. IF DISORDER= BULLIMIA_NERVOSA AND DEHYDRATION=YES

THEN TREATMENT= FLUOXETINE, COUNSELLING THERAPY

14. IF DISORDER= GENERALIZED_ANXIETY_DISORDER AND SWEATING=YES
THEN TREATMENT= SSRIs, ANXIOLYTICS, PSYCHOANALYSIS

15. IF DISORDER= PANIC_DISORDER_WITH_AGORAPHOBIA AND POUNDING_HEARTRATE=YES
THEN TREATMENT= BENZODIAZEPINES, PAXIL, EYE MOVEMENT DESENSITIZATION AND REPROCESSING

16. IF DISORDER= BODY_DISMORPHIC_DISORDER AND GENERAL_DISCONTENT=YES
THEN TREATMENT=PROZAC, ZOLOFT, PSYCHOEDUCATION

17. IF DISORDER= DISSOCIATIVE_IDENTITY_DISORDER AND DEREALIZATION=YES
THEN TREATMENT=HYPNOTHERAPY, ADJUNCTIVE THERAPY

18. IF DISORDER= PSYCHOSIS AND NONSENSE_WORD_REPETITION=YES
THEN TREATMENT= RISPERIDONE, QUETIAPINE, FAMILY THERAPY

19. IF DISORDER= SCHIZOAFFECTIVE_DISORDER AND MANIA=YES
THEN TREATMENT= ANTIPSYCHOTIC, PYSCHOTHERAPY SOCIAL RYTHM THERAPY

6. PROGRAM IMPLEMENTATION

The program is implemented using python language. In this project we have implemented three python files that is Project1_A05252367_Disorder_BW.py (backward chaining algorithm), Project1_A05252367_Drug_FW.py (forward chaining algorithm) and Project1_A05252367_main.py file which calls backward chaining and forward chaining algorithms.

6.1 Backward Chaining

The backward chaining algorithm is implemented in a separate python file where this file imports the main file. Conclusion list, clause variable list is stored in main file and variable list, knowledge base, derived variable lists are stored as JSON file. JSON is a structured dataset and is suitable for representing hierarchical data like trees or nested properties. Once main calls the process(goal) function of backward chaining with the goal as input “DISORDER” which needs to be determined. This goal variable will be passed input to the search_con(goal) function where this finds the rule number (ri) for the corresponding goal. Now the rule number is passed as input to the rule_to_clause(rule_num) where this calculates the clause number by substituting the rule number in the formula. Next the computed clause number will be passed as input to the update_VL(clause_number) this will ask the user multiple questions regarding the symptoms and based on user response it will update the variable list and derived variable list. If the update_VL function encounters any intermediate node first it checks whether the system generated any output for this variable. If it has some value then it will continue with other variables else it will call process(intermediate_node). Once variables are instantiated for that rule(ri) and next function will be called is validate_ri(rule_num) this function checks rules present in knowledge base which is stored in JSON and if any rule is satisfied with user input, then the conclusion is returned otherwise none value is returned.

This process continues until some disorder is returned by the validate_ri function. This can be checked by storing all the disorders in the DISORDER LIST variable and continue loop if the conclusion is not in DISORDER LIST. There are three scenarios where loop might break

1. If the user response satisfies any rule and conclusion present DISORDER LIST

2. If the user is not sick that is sick = no then it will break from the loop

3.If the user is suffering from some other disorder which is not included in our project, then all the intermediate nodes will be having some value, but conclusion will not be returned. So, in this case there is a check where if all intermediate nodes filled with some values yes or no then loop breaks.

6.2 Main

In Project1_A05252367_main.py we have main function which is a special function as it is a starting point of program execution and in this file variables that are required for backward chaining and forward chaining are declared. In this file both backward chaining algorithm and forward chaining algorithm are imported. Once the main function is called, the goal variable =" DISORDER" is initialized and then the process function is called. Now the backward chaining algorithm completes the execution and returns the conclusion. Based on the value of the conclusion, the forward chaining algorithm is called.

1. If the conclusion is some disorder which is present in the DISORDER LIST, then the process function of the forward chaining algorithm is called. Once the forward chaining algorithm is executed, it will return treatment to the main.

2. If the conclusion is None or NO DISORDER then the user is not having any disorder. So, the treatment is not needed for that user.

Finally, the time and space consumption are calculated and outputs displayed

6.3 Forward Chaining

The forward chaining algorithm is implemented in a separate python file where this file is imported by the main file. Clause variable list is stored in main file and variable list, knowledge base, derived variable lists are stored as JSON file. Forward chaining algorithm is called only after completion of backward chaining algorithm and returned conclusion should be some valid disorder that is conclusion should be present in DISORDER LIST. Once main calls the **process(disorder)** function in forward chaining algorithm where input disorder is the output generated by backward chaining algorithm. This disorder variable will be stored in the derived variable list and "DISORDER" is passed as input to the **search_con(goal)** function where this finds the clause number (ri) for the corresponding goal. Now the clause number is passed as input to the **update_VL(clause_number)** this function will ask the user multiple questions regarding the symptoms and based on user response it will update the variable list and derived variable list. Next, the **clause_to_rule(clause_num)** function is called where this calculates the rule number by substituting the clause number in the formula. Once variables are instantiated for that rule(ri) and the next function that will be called is **validate_ri(rule_num)**.

This function checks rules present in the knowledge base which is stored in JSON and if any rule is satisfied with user input, then the conclusion is returned otherwise none value is returned.

This process continues until some treatment is returned by the validate_ri function.

7. SOURCE CODE

Program is implemented using python language.

7.1 Backward Chaining

Project1_A05252367_Disorder_BW.py:

```
"""
we have implemented by using option#1 that is we used algorithm for building the AI expert system

********** File name:Project1_A05252367_Disorder_BW.py **********

# This file is implementation Backward chaining algorithm.
# This is called once Project1_A05252367_main.py execution starts.
# it import Project1_A05252367_main.py to use all the variables declared
# knowledge base and variable list json files stored dictionary
# The execution starts from process function where "DISORDER" is passed from main

**************************************************************************
"""


import Project1_A05252367_main as M
import json
import logging

# Creating an object
LOG = logging.getLogger()
LOG.setLevel(logging.DEBUG)


#READING AND LOADING ALL THE NECESSARY JSON FILES NEEDED FOR PROCESSING
json_file1 = open('Project1_A05252367_BW_KNOWLEDGE_BASE.json', "r")
kb_rules = json.load(json_file1)
json_file2 = open('Project1_A05252367_BW_VARIABLE_LIST.json', "r")
variable_list = json.load(json_file2)
json_file3 = open('Project1_A05252367_BW_INTERMEDIATE_NODE.json', "r")
intermediate_node = json.load(json_file3)
json_file1.close()
json_file2.close()
json_file3.close()

# This variable stores all the rules which are executed by validate function
visited_rules=[]


"""
**************************************************************************
# search_con(): This function searches goal_variable in CONCLUSION_LIST.
# If the matching goal is found then it checks corresponding rule
# whether it is already visited or not by using visited_rules.
# If not visited then it return rule_number
# else checks for another matching goal and process repeats again
# goal_variable: It is the string(goal) where it needs to be searched in CONCLUSION_LIST
```

```
# Rule number is returned to process function
*****************************************************************************
"""

def search_con(goal_variable):
    LOG.info("INSIDE SEARCH_CON FUCNTION WITH GOAL VARIABLE :%s" % goal_variable)
    # assigning some invalid rule number if no rule is found this will be returned
    rule_num=-1

    # In CONCLUSION_LIST every value is checked for matching goal
    for ri,con in M.CONCLUSION_LIST.items():

        # checking goal variable with conclusions in conclusion_list and the corresponding rule should not be visited
        if(con==goal_variable and ri not in visited_rules):

            #store the rule and breaks from loop
            rule_num=ri
            break

    LOG.info("visited rules: %s" %visited_rules)
    LOG.info("THIS IS THE RULE TO BE PASSED TO NEXT FUNCTION:%s" % rule_num)

    return rule_num

"""
*****************************************************************************
# rule_to_clause(): This function converts rule number to clause number
# Formula used is 10 * ( rule number -1)+1. Rules number are in the form of 1,2,3,..
# rule number: This is an integer which is calculated by search_con function and
# sent as input to rule_to_clause() function
# Clause number is calculated by using the formula and it is returned to process function
*****************************************************************************
"""
def rule_to_clause(rule_number:int):
    LOG.info("INSIDE THE RULE_TO_CLAUSE FUNCTION WITH RULE NUMBER :%s " % rule_number)

    #formula for calculating clause number
    clause_number=10*(rule_number-1)+1
    LOG.info("THIS IS THE CALCULATED CLAUSE NUMBER :%s "% clause_number)

    return clause_number

"""
*****************************************************************************
# update_VL: This function asks the user several questions
# It asks questions to get the values for the variables present in CLAUSE_VARIABLE_LIST for that clause number.
# The users input is stored in VARIABLE_LIST. But if the variable already instantiated then it just skips to ask
# that question. If the variable occured is not present in VARIABLE_LIST then it calls process function as that
# variable is a intermediate node or if the intermediate node is already processed then we just skip.
# Every time we get answer from user then it is stored in DERIVED_VARIABLE_LIST
# clause number that is calculated by rule_to_clause function is passed as input to this function update_VL
*****************************************************************************
"""

def update_VL(clause_number:int):
    LOG.info("INSIDE UPDATE VL FUNCTION AT WITH CLAUSE NUMBER :%s "% clause_number)

    #stores clause variable list corresponding to given clause number
    temp_clause_list=M.CLAUSE_VARIABLE_LIST[clause_number]

    # If there is intermediate node in CLAUSE_VARIABLE_LIST for the given clause_number and it is not processed
    # then it just call process function which recursive call and execution continues once the intermediate node is processed
```

```
    for clause_var in temp_clause_list:

        #If variable is intermediate then condition will be true
        if(clause_var in intermediate_node.keys() ):

            #checking intermediate node value if this is empty then if condition will be true
            if(intermediate_node[clause_var]['SystemOutput']==""):

                #calls process function with intermediate node as input
                process(clause_var)

    # checks every variable in CLAUSE_VARIABLE_LIST for the give clause_number and takes input from user
    for i in range(len(temp_clause_list)):

            # If there is intermediate node  which is processed and generated value is "no" then condition
            # becomes false and exists the funtion as there is no need ask further question to the user
            if(temp_clause_list[i] in intermediate_node.keys() and
intermediate_node[temp_clause_list[i]]['SystemOutput']=="no"):
                return "done"

            # If the variable is present in variable_list then condition will be true
            if temp_clause_list[i] in variable_list:

                # if the variable is not instantiated then condition will be true
                if(variable_list[temp_clause_list[i]]['Userinput']==""):

                    # loop continues until every variable is instantiated for that clause number in clause_variable_list
                    while(1):

                        # Asking the user questions regarding the  symptoms which will be "yes" or "no"
                        inputvariable = input(variable_list[temp_clause_list[i]]["Question"]+""+temp_clause_list[i]+"? ")

                        # checking if the user input is "yes" or "no" if he enters other than these.
                        # Same question will be asked again
                        if inputvariable.lower() in  ["yes","no"]:

                            # user response is valid and is stored in variable_list
                            variable_list[temp_clause_list[i]]['Userinput'] = inputvariable.lower()
                            break

                    # stores user response in this variable
                    M.DERIVED_VARIABLE_LIST[temp_clause_list[i]] = variable_list[temp_clause_list[i]]['Userinput']

    LOG.info("UPDATING THE DERIVED VARIABLE LIST AS :%s "% M.DERIVED_VARIABLE_LIST)
    return "done"

"""
*****************************************************************************
# validate_ri(): This function checks the ri rule in kb_rules with the user input present in VARIABLE_LIST.
# Once it satisfies the kb rules then it will return corresponding conclusion else it will return None.
# It also adds every rule that is validated into visited_rules listwhich will later used in search_con to
# track rules which are executed
# FUNCTION INPUTS:
# ri : Rule number that we need to validate
# conclusion : it is just None value
# FUNCTION RETURN:
# It will return conclusion variable. If the rule is satisfied it will return conclusion in kb_rules
# else returns None
*****************************************************************************
"""

def validate_ri(ri:int,conclusion:str):
```

```
    LOG.info("INSIDE UPDATE VALIDATE_RULE FUCNTION WITH RULE NUMNER :%s "% ri)

    #converting to string as rule numbers are stored as string in kb_rule
    rule_num = str(ri)

    # A local variable which is created to store the variables used in the rule ri
    symptoms_list=list(kb_rules[rule_num]['SYMPTOMS'].keys())

    LOG.info("PRINTING THE SYMPTOM PRESENT IN KNOWLEGE BASE :%s "% symptoms_list)

    # A flag to track whether the rule is satisfied or not
    # flag=1 rule not satisfied and flag=0 rule satisfied
    flag=0

    #append that rule into visited
    visited_rules.append(ri)

    # checks each variable in kb rule(ri) with userInput if there is any mismatch loop breaks and returns None
    # else assigns conclusion variable with conclusion in kb_rule(ri) and then return conclusion
    for symptom in symptoms_list:

        # check whether the user input in variable_list is matching with kb_rules or not
        # check if the intermediate node is satisfied or not
        if((symptom  in variable_list and kb_rules[rule_num]['SYMPTOMS'][symptom] ==
variable_list[symptom]['Userinput'])
          or (symptom in intermediate_node and kb_rules[rule_num]['SYMPTOMS'][symptom] ==
intermediate_node[symptom]['SystemOutput'])):
            continue
        else:
            #Rule is not satisfied and assigns flag to 1 and breaks from the loop
            flag=1
            break

    # flag =0 means the rule(ri) is satisfied and returns the conclusion
    if(flag == 0):
        conclusion=kb_rules[rule_num]['CONCLUSION']
        LOG.info("RULE IS SATISFIED AND THE CONCLUSION RETUREND IS :%s"% conclusion)

         # if the conclusion is intermediate variable then assign yes and add same in the derived_variable_list
        if(conclusion in intermediate_node.keys()):
            intermediate_node[conclusion]["SystemOutput"]="yes"
            M.DERIVED_VARIABLE_LIST[conclusion]=intermediate_node[conclusion]["SystemOutput"]

    else:
        #rule is not satisfied if the conclusion is intermediate node then assign no and add same in the derived_variable_list
        if(kb_rules[rule_num]['CONCLUSION'] in intermediate_node.keys()):
            intermediate_node[kb_rules[rule_num]['CONCLUSION']]["SystemOutput"]="no"

M.DERIVED_VARIABLE_LIST[kb_rules[rule_num]['CONCLUSION']]=intermediate_node[kb_rules[rule_num]['CONCL
USION']]["SystemOutput"]

    LOG.info("This is validate return value:%s"%conclusion)
    return conclusion

"""
*****************************************************************************
# process(): This function just process the goal by calling search_con,rule_to_clause, update_VL and validate_ri.
# The execution continues until goal is reached or if the user is not suffering then loop breaks.
# goal: Initially goal will be "DISORDER" and the process starts from this.
# returns conclusion that is some disorder which is in DISORDER_LIST or it can be none or no disorder
# if the user is not sick
```

```
**************************************************************************
"""
def process(goal):

    LOG.info("INSIDE THE PROCESS FUNCTION WITH GLOBAL VARIABLE :%s" % goal)

    #if the goal is intermediate node and is not processed then condition will be true
    # or if the goal is not occured and it is "no disorder" then condition will be true
    while((goal in intermediate_node.keys() and intermediate_node[goal]["SystemOutput"]=="") or
     (goal not in intermediate_node.keys() and  M.backward_conclusions not in M.DISORDER_LIST and
M.backward_conclusions != "NO DISORDER")):

        # calling search_con function to find the rule number to the goal variable
        rule_num = search_con(goal)

        # The rules is passed as input to rule_to_clause and clause number is returned
        clause_num = rule_to_clause(rule_num)

        # calling update_vl function to ask the userinput for that rule
        d = update_VL(clause_num)

        # once input taken from user validation of that rule  is done by calling validate_ri with rule ri and conclusion None as
input
        M.backward_conclusions = validate_ri(rule_num,M.backward_conclusions)

        # if the ANXIETY is not satisfied then loop breaks
        # this is because ANXIETY is the mandatory symptom for all disorders if the user is not
        # feeling ANXIETY he is not having any disorder
        if intermediate_node["ANXIETY"]["SystemOutput"] == "no":
            break

        # flag1 to check if all the intermediate nodes are processed and still the conclusion is not occured
        # then flag1 will be 0 else if any intermediate node is not processed then flag will be 1 and for loop breaks
        flag1=0
        for key in intermediate_node.keys():
            if intermediate_node[key]["SystemOutput"]=="":
                flag1=1
                break

        # while loop breaks as user is not suffering from any disorder
        if flag1==0:
            break

    LOG.info("YOUR VALUE IS :%s" % M.backward_conclusions)

    # loading the conclusion value into DERIVED_VARIABLE_LIST
    M.DERIVED_VARIABLE_LIST["DISORDER"] = M.backward_conclusions

    # loading the DERIVED_VARIABLE_LIST into json file
    json_file = open('Project1_A05252367_BW_DERIVED_VARIABLE_LIST.json', "w")
    json.dump(M.DERIVED_VARIABLE_LIST, json_file, indent=6)
    json_file.close()

    #returning conclusion of backward_chaining that is goal value
    return M.backward_conclusions
```

Project1_A05252367_BW_DERIVED_VARIABLE_LIST.json:
This file contains user response and system generated output

```
{
    "SICK": "yes",
```

```
    "IRRITABILITY": "yes",
    "UNFOCUSED": "yes",
    "RESTLESNESS": "yes",
    "ANXIETY": "yes",
    "DISORDER": "KLEPTOMANIA",
    "MOOD SWINGS": "no",
    "SOCIAL ISOLATION": "no",
    "PERSONALITY DISEASE": "no",
    "SADNESS": "yes",
    "ANGRY": "yes",
    "DEPRESSION": "yes",
    "STEALING": "yes",
    "STEALING PLEASURE": "yes",
    "GUILT": "yes"
}
```


Project1_A05252367_BW_INTERMEDIATE_NODE.json
It contains all the intermediate nodes in decision tree


```
{
  "ANXIETY":{
        "SystemOutput":""
    },
  "PERSONALITY DISEASE":{
        "SystemOutput":""
    },
  "DEPRESSION":{
        "SystemOutput":""
    },
  "PSYCHOTIC":{
        "SystemOutput":""
    },
  "CHRONIC BRAIN DISEASE":{
        "SystemOutput":""
    },
  "SLEEP DISEASE":{
        "SystemOutput":""
    },
  "EATING DISEASE":{
        "SystemOutput":""
    }
}
```


Project1_A05252367_BW_KNOWLEDGE_BASE.json
This file is knowledge base which contains rules


```
{
  "1":{
    "CONCLUSION":"NO DISORDER",
    "SYMPTOMS":{
      "SICK":"no"
    }
  },
  "2":{
    "CONCLUSION":"ANXIETY",
    "SYMPTOMS":{
      "SICK":"yes",
      "IRRITABILITY":"yes",
      "UNFOCUSED":"yes",
      "RESTLESNESS":"yes"
    }
  },
  "3":{
    "CONCLUSION":"PERSONALITY DISEASE",
```

```
  "SYMPTOMS":{
    "ANXIETY":"yes",
    "MOOD SWINGS":"yes",
    "SOCIAL ISOLATION":"yes"
  }
},
"4":{
  "CONCLUSION":"OBSESSIVE COMPULSIVE DISORDER",
  "SYMPTOMS":{
    "PERSONALITY DISEASE":"yes",
    "AGITATION":"yes",
    "IMPULSIVITY":"yes",
    "COMPULSIVE BEHAVIOUR":"yes",
    "HYPERVIGILENCE":"yes",
    "RITUALISTIC BEHAVIOUR":"yes",
    "REPETITIVE BEHAVIOUR":"yes"
  }
},
"5":{
  "CONCLUSION":"BORDERLINE PERSONALITY DISORDER",
  "SYMPTOMS":{
    "PERSONALITY DISEASE":"yes",
    "BOREDOM":"yes",
    "DISTORTED SELF IMAGE":"yes",
    "EMPTYNESS":"yes",
    "LOSS OF INTEREST":"yes"
  }
},
"6":{
  "CONCLUSION":"DEPRESSION",
  "SYMPTOMS":{
    "ANXIETY":"yes",
    "SADNESS":"yes",
    "ANGRY":"yes"
  }
},
"7":{
  "CONCLUSION":"KLEPTOMANIA",
  "SYMPTOMS":{
    "DEPRESSION":"yes",
    "STEALING":"yes",
    "STEALING PLEASURE":"yes",
    "GUILT":"yes"
  }
},
"8":{
  "CONCLUSION":"BIPOLAR DISORDER",
  "SYMPTOMS":{
    "DEPRESSION":"yes",
    "HYPERACTIVITY":"yes"
  }
},
"9":{
  "CONCLUSION":"DYSTHYMIA",
  "SYMPTOMS":{
    "DEPRESSION":"yes",
    "POOR APPETITE":"yes"
  }
},
"10":{
  "CONCLUSION":"PSYCHOTIC",
  "SYMPTOMS":{
```

```
    "DEPRESSION":"yes",
    "HALLUCINATION":"yes",
    "SUICIDAL THOUGHTS":"yes"
  }
},
"11":{
  "CONCLUSION":"CHRONIC BRAIN DISEASE",
  "SYMPTOMS":{
    "DEPRESSION":"yes",
    "HALLUCINATION":"yes",
    "SUICIDAL THOUGHTS":"no",
    "MENTAL DECLINE":"yes",
    "LACK OF RESTRAINT":"yes",
    "NERVOUSNESS":"yes"
  }
},
"12":{
  "CONCLUSION":"DEMENTIA",
  "SYMPTOMS":{
    "CHRONIC BRAIN DISEASE":"yes",
    "JUMBLED SPEECH":"yes"
  }
},
"13":{
  "CONCLUSION":"ALZHEIMERS DISEASE",
  "SYMPTOMS":{
    "CHRONIC BRAIN DISEASE":"yes",
    "SUSPICIOUS":"yes"
  }
},
"14":{
  "CONCLUSION":"SLEEP DISEASE",
  "SYMPTOMS":{
    "PSYCHOTIC":"yes",
    "SLEEPLESSNESS":"yes"
  }
},
"15":{
  "CONCLUSION":"INSOMNIA",
  "SYMPTOMS":{
    "SLEEP DISEASE":"yes",
    "HEADACHE":"yes"
  }
},
"16":{
  "CONCLUSION":"NARCOLEPSY",
  "SYMPTOMS":{
    "SLEEP DISEASE":"yes",
    "HEADACHE":"no",
    "LOSS OF MUSCLE":"yes",
    "CATAPLEXY":"yes",
    "SLEEP PARALYSIS":"yes"
  }
},
"17":{
  "CONCLUSION":"MAJOR DEPRESSIVE DISORDER",
  "SYMPTOMS":{
    "SLEEP DISEASE":"yes",
    "SLOW THINKING":"yes"
  }
},
"18":{
```

```
  "CONCLUSION":"EATING DISEASE",
  "SYMPTOMS":{
    "ANXIETY":"yes",
    "FATIGUE":"yes",
    "WEIGHT FLUCTUATION":"yes",
    "LACK OF CONFIDENCE":"yes"
  }
},
"19":{
  "CONCLUSION":"POST TRAUMATIC STRESS DISORDER",
  "SYMPTOMS":{
    "ANXIETY":"yes",
    "FATIGUE":"no",
    "NIGHTMARES":"yes",
    "TRAUMA MEMORIES":"yes"
  }
},
"20":{
  "CONCLUSION":"GENERALIZED ANXIETY DISORDER",
  "SYMPTOMS":{
    "ANXIETY":"yes",
    "FATIGUE":"yes",
    "WEIGHT FLUCTUATION":"no",
    "CHEST PAIN":"no"
  }
},
"21":{
  "CONCLUSION":"PANIC DISORDER WITH AGORAPHOBIA",
  "SYMPTOMS":{
    "ANXIETY":"yes",
    "FATIGUE":"yes",
    "WEIGHT FLUCTUATION":"no",
    "CHEST PAIN":"yes",
    "DIZZINESS":"yes",
    "SWEATING":"yes",
    "NAUSEA":"yes",
    "HELPLESSNESS":"yes",
    "FEAR OF BEING ALONE":"yes"
  }
},
"22":{
  "CONCLUSION":"BODY DISMORPHIC DISORDER",
  "SYMPTOMS":{
    "EATING DISEASE":"yes",
    "BODY THINKING":"yes",
    "APPEARANCE CHANGE":"yes",
    "PICKY SKIN":"yes"
  }
},
"23":{
  "CONCLUSION":"BULIMIA NERVOSA",
  "SYMPTOMS":{
    "EATING DISEASE":"yes",
    "BODY THINKING":"yes",
    "APPEARANCE CHANGE":"no",
    "BINGE EATING":"yes",
    "VOMITING":"yes",
    "FASTING":"yes",
    "LAXATIVE USE":"yes"
  }
},
"24":{
```

```
    "CONCLUSION":"SCHIZOPHRENIA",
    "SYMPTOMS":{
      "PSYCHOTIC":"yes",
      "DELUSION":"yes",
      "DISORGANIZED THINKING":"yes",
      "LACK OF MOTIVATION":"yes",
      "AMNESIA":"yes",
      "INCOHERENT SPEECH":"yes",
      "EXCITABILITY":"yes"
    }
  },
  "25":{
    "CONCLUSION":"DISSOCIATIVE IDENTITY DISORDER",
    "SYMPTOMS":{
      "PSYCHOTIC":"yes",
      "DELUSION":"yes",
      "DISORGANIZED THINKING":"yes",
      "LACK OF MOTIVATION":"yes",
      "AMNESIA":"yes",
      "INCOHERENT SPEECH":"no",
      "IDENTITY CONFUSION":"yes",
      "BLACKOUT":"yes"
    }
  },
  "26":{
    "CONCLUSION":"PSYCHOSIS",
    "SYMPTOMS":{
      "PSYCHOTIC":"yes",
      "DELUSION":"yes",
      "DISORGANIZED THINKING":"yes",
      "LACK OF MOTIVATION":"yes",
      "AMNESIA":"no",
      "HOPELESSNESS":"no",
      "HOSTILITY":"yes"
    }
  },
  "27":{
    "CONCLUSION":"SCHIZOAFFECTIVE DISORDER",
    "SYMPTOMS":{
      "PSYCHOTIC":"yes",
      "DELUSION":"yes",
      "DISORGANIZED THINKING":"yes",
      "LACK OF MOTIVATION":"yes",
      "AMNESIA":"no",
      "HOPELESSNESS":"yes",
      "GRANDIOSITY":"yes"
    }
  }
}
```


Project1_A05252367_BW_VARIABLE_LIST.json

This file contains all the variable which is nothing but symptoms and the user response is stored in this file


```
{
  "SICK":{
    "Question":"Are you ",
    "Userinput":""
  },
  "IRRITABILITY":{
    "Question":"Do you have ",
```

```
   "Userinput":""
},
"UNFOCUSED":{
  "Question":"Do you have ",
   "Userinput":""
},
"RESTLESNESS":{
  "Question":"Do you have ",
   "Userinput":""
},
"MOOD SWINGS":{
  "Question":"Do you have ",
   "Userinput":""
},
"SOCIAL ISOLATION":{
  "Question":"Do you have ",
   "Userinput":""
},
"AGITATION":{
  "Question":"Do you have ",
   "Userinput":""
},
"IMPULSIVITY":{
  "Question":"Do you have ",
   "Userinput":""
},
"COMPULSIVE BEHAVIOUR":{
  "Question":"Do you have ",
   "Userinput":""
},
"HYPERVIGILENCE":{
  "Question":"Do you have ",
   "Userinput":""
},
"RITUALISTIC BEHAVIOUR":{
  "Question":"Do you have ",
   "Userinput":""
},
"REPETITIVE BEHAVIOUR":{
  "Question":"Do you have ",
   "Userinput":""
},
"BOREDOM":{
  "Question":"Do you have ",
   "Userinput":""
},
"DISTORTED SELF IMAGE":{
  "Question":"Do you have ",
   "Userinput":""
},
"EMPTYNESS":{
  "Question":"Do you have feeling of ",
   "Userinput":""
},
"LOSS OF INTEREST":{
  "Question":"Do you have ",
   "Userinput":""
},
"SADNESS":{
  "Question":"Do you have ",
   "Userinput":""
},
```

```
"ANGRY":{
  "Question":"Are you ",
   "Userinput":""
},
"STEALING":{
  "Question":"Do you have habit of ",
   "Userinput":""
},
"STEALING PLEASURE":{
 "Question":"Do you get ",
 "Userinput":""
 },
"GUILT":{
  "Question":"Do you have feeling of ",
   "Userinput":""
},
"HYPERACTIVITY":{
  "Question":"Do you have ",
   "Userinput":""
},
"POOR APPETITE":{
  "Question":"Do you have ",
   "Userinput":""
},
"HALLUCINATION":{
  "Question":"Do you have ",
   "Userinput":""
},
"SUICIDAL THOUGHTS":{
  "Question":"Do you have ",
   "Userinput":""
},
"MENTAL CONFUSION":{
  "Question":"Do you have ",
   "Userinput":""
},
"PARANOIA":{
  "Question":"Do you have ",
   "Userinput":""
},
"MENTAL DISORIENTATION":{
  "Question":"Do you have ",
   "Userinput":""
},
"MENTAL DECLINE":{
  "Question":"Do you have ",
   "Userinput":""
},
"LACK OF RESTRAINT":{
  "Question":"Do you have ",
   "Userinput":""
},
"NERVOUSNESS":{
  "Question":"Do you have ",
   "Userinput":""
},
"JUMBLED SPEECH":{
  "Question":"Do you have ",
   "Userinput":""
},
"SUSPICIOUS":{
  "Question":"are you ",
```

```
  "Userinput":""
},
"SLEEPLESSNESS":{
  "Question":"Do you have ",
  "Userinput":""
},
"HEADACHE":{
  "Question":"Do you have ",
  "Userinput":""
},
"LOSS OF MUSCLE":{
  "Question":"Do you have ",
  "Userinput":""
},
"CATAPLEXY":{
  "Question":"Do you have ",
  "Userinput":""
},
"SLEEP PARALYSIS":{
  "Question":"Do you have ",
  "Userinput":""
},
"SLOW THINKING":{
  "Question":"Do you have ",
  "Userinput":""
},
"FATIGUE":{
  "Question":"Do you have ",
  "Userinput":""
},
"WEIGHT FLUCTUATION":{
  "Question":"Do you have ",
  "Userinput":""
},
"LACK OF CONFIDENCE":{
  "Question":"Do you have ",
  "Userinput":""
},
"NIGHTMARES":{
  "Question":"Do you have ",
  "Userinput":""
},
"TRAUMA MEMORIES":{
  "Question":"Do you have ",
  "Userinput":""
},
"CHEST PAIN":{
  "Question":"Do you have ",
  "Userinput":""
},
"DIZZINESS":{
  "Question":"Do you have ",
  "Userinput":""
},
"SWEATING":{
  "Question":"Do you have ",
  "Userinput":""
},
"NAUSEA":{
  "Question":"Do you have ",
  "Userinput":""
},
```

```
"HELPLESSNESS":{
  "Question":"Do you have ",
   "Userinput":""
},
"FEAR OF BEING ALONE":{
  "Question":"Do you have ",
   "Userinput":""
},
"BODY THINKING":{
  "Question":"Do you have ",
   "Userinput":""
},
"APPEARANCE CHANGE":{
  "Question":"Do you have ",
   "Userinput":""
},
"PICKY SKIN":{
  "Question":"Do you have ",
   "Userinput":""
},
"BINGE EATING":{
  "Question":"Do you have habit of ",
   "Userinput":""
},
"VOMITING":{
  "Question":"Do you have ",
   "Userinput":""
},
"FASTING":{
  "Question":"Are you ",
   "Userinput":""
},
"LAXATIVE USE":{
  "Question":"Are you doing ",
   "Userinput":""
},
"DELUSION":{
  "Question":"Do you have ",
   "Userinput":""
},
"DISORGANIZED THINKING":{
  "Question":"Do you have ",
   "Userinput":""
},
"LACK OF MOTIVATION":{
  "Question":"Do you have ",
  "Userinput":""
},
"AMNESIA":{
  "Question":"Do you have ",
   "Userinput":""
},
"INCOHERENT SPEECH":{
  "Question":"Do you have ",
   "Userinput":""
},
"EXCITABILITY":{
  "Question":"Do you have ",
   "Userinput":""
},
"IDENTITY CONFUSION":{
  "Question":"Do you have ",
```

```
    "Userinput":""
  },
  "BLACKOUT":{
    "Question":"Do you have ",
    "Userinput":""
  },
  "HOSTILITY":{
    "Question":"Do you have ",
    "Userinput":""
  },
  "HOPELESSNESS":{
    "Question":"Do you have ",
    "Userinput":""
  },
  "GRANDIOSITY":{
    "Question":"Do you have ",
    "Userinput":""
  }
}
```

7.2 Main

Project1_A05252367_main.py

```
"""
we have implemented by using option#1 that is we used algorithm for building the AI expert system


******** File Name: Project1_A05252367_main.py **********
# This file contains the declaration and initilaization of variables
# It imports Project1_A05252367_Disorder_BW (BW) and Project1_A05252367_Drug_FW (FW) which are
backward and forward chaining alogorithms respectively.
# In this file all the variables required for BW and FW are declared
# Backward (BW) : CONCLUSION_LIST, CLAUSE_VARIABLE_LIST, DISORDER_LIST, backward_conclusion
# Forward (FW) : FORWARD_CLAUSE_VARIABLE_LIST, forward_conclusion
************************************************************
"""

import logging
import Project1_A05252367_Disorder_BW as BW
import Project1_A05252367_Drug_FW as FW
import time
import psutil

# CONCLUSION_LIST- It is a dictionary where it stores the rule number and conclusion of each rule
# this is used in backward chaining algorithm
CONCLUSION_LIST = {
   1 : "DISORDER",
   2 : "ANXIETY" ,
   3 : "PERSONALITY DISEASE" ,
   4 : "DISORDER" ,
   5 : "DISORDER" ,
   6 : "DEPRESSION" ,
   7 : "DISORDER" ,
   8 : "DISORDER" ,
   9 : "DISORDER" ,
   10 : "PSYCHOTIC" ,
   11 : "CHRONIC BRAIN DISEASE" ,
   12 : "DISORDER" ,
   13 : "DISORDER" ,
   14 : "SLEEP DISEASE" ,
   15 : "DISORDER" ,
```

```
    16 : "DISORDER" ,
    17 : "DISORDER" ,
    18 : "EATING DISEASE" ,
    19 : "DISORDER" ,
    20 : "DISORDER" ,
    21 : "DISORDER" ,
    22 : "DISORDER" ,
    23 : "DISORDER" ,
    24 : "DISORDER" ,
    25 : "DISORDER" ,
    26 : "DISORDER" ,
    27 : "DISORDER"
}

#CLAUSE_VARIABLE_LIST is a dictionary with clause number as key and variables of if clause are stored in a list as value to that key(clause number)
# This is used backward chaining algorithm and 10 slots for each rule

CLAUSE_VARIABLE_LIST= {
    1 : ["SICK"],
    11 : ["SICK", "IRRITABILITY","UNFOCUSED","RESTLESNESS"],
    21 : ["ANXIETY", "MOOD SWINGS" , "SOCIAL ISOLATION"],
    31 : ["PERSONALITY DISEASE", "AGITATION", "IMPULSIVITY", "COMPULSIVE BEHAVIOUR",
"HYPERVIGILENCE", "RITUALISTIC BEHAVIOUR", "REPETITIVE BEHAVIOUR"],
    41 : ["PERSONALITY DISEASE", "BOREDOM", "DISTORTED SELF IMAGE", "EMPTYNESS", "LOSS OF
INTEREST"],
    51 : ["ANXIETY", "SADNESS","ANGRY"],
    61 : ["DEPRESSION", "STEALING", "STEALING PLEASURE", "GUILT"],
    71 : ["DEPRESSION", "HYPERACTIVITY"],
    81 : ["DEPRESSION", "POOR APPETITE"],
    91 : ["DEPRESSION", "HALLUCINATION", "SUICIDAL THOUGHTS"],
    101 : ["DEPRESSION", "HALLUCINATION" , "SUICIDAL THOUGHTS", "MENTAL CONFUSION", "PARANOIA",
"MENTAL DISORIENTATION", "MENTAL DECLINE", "LACK OF RESTRAINT", "NERVOUSNESS"],
    111 : ["CHRONIC BRAIN DISEASE", "JUMBLED SPEECH"],
    121 : ["CHRONIC BRAIN DISEASE", "SUSPICIOUS"],
    131 : ["PSYCHOTIC", "SLEEPLESSNESS"],
    141 : ["SLEEP DISEASE", "HEADACHE"],
    151 : ["SLEEP DISEASE", "HEADACHE", "LOSS OF MUSCLE", "CATAPLEXY", "SLEEP PARALYSIS"],
    161 : ["SLEEP DISEASE", "SLOW THINKING"],
    171 : ["ANXIETY", "FATIGUE", "WEIGHT FLUCTUATIONS", "LACK OF CONFIDENCE"],
    181 : ["ANXIETY", "FATIGUE", "NIGHTMARES", "TRAUMA MEMORIES"],
    191 : ["ANXIETY", "FATIGUE", "WEIGHT FLUCTUATION", "CHEST PAIN"],
    201 : ["ANXIETY", "FATIGUE", "WEIGHT FLUCTUATION", "CHEST PAIN", "DIZZINESS", "SWEATING",
"NAUSEA", "HELPLESSNESS", "FEAR OF BEING ALONE"],
    211 : ["EATING DISEASE", "BODY THINKING", "APPEARANCE CHANGE", "PICKY SKIN"],
    221 : ["EATING DISEASE", "BODY THINKING", "APPEARANCE CHANGE", "BINGE EATING", "VOMITING",
"FASTING", "LAXATIVE USE"],
    231 : ["PSYCHOTIC", "DELUSION", "DISORGANIZED THINKING", "LACK OF MOTIVATION", "AMNESIA",
"INCOHERENT SPEECH", "EXCITABILITY"],
    241 : ["PSYCHOTIC", "DELUSION", "DISORGANIZED THINKING", "LACK OF MOTIVATION", "AMNESIA",
"INCOHERENT SPEECH", "IDENTITY CONFUSION", "BLACKOUT"],
    251 : ["PSYCHOTIC", "DELUSION", "DISORGANIZED THINKING", "LACK OF MOTIVATION", "AMNESIA",
"HOPELESSNESS", "HOSTILITY"],
    261 : ["PSYCHOTIC", "DELUSION", "DISORGANIZED THINKING", "LACK OF MOTIVATION", "AMNESIA",
"HOPELESSNESS", "GRANDIOSITY"]
}

# FORWARD_CLAUSE_VARIABLE_LIST  this is dictionary with clause number as key and variables of if clause are stored in a list as a value to the key(clause number)
# This is used by forwarding chaining algorithm and 3 slots for each rule
FORWARD_CLAUSE_VARIABLE_LIST = {
    1 : ["DISORDER", "THOUGHT DISORDER"],
```

```
    4 : ["DISORDER", "COMPULSIVE HOARDING"],
    7 : ["DISORDER", "URGE TO STEAL"],
    10 : ["DISORDER", "TALKATIVENESS"],
    13 : ["DISORDER", "LOSS OF INTEREST"],
    16 : ["DISORDER", "INDECISIVENESS"],
    19 : ["DISORDER", "MEMORY LOSS"],
    22 : ["DISORDER", "PERSONALITY CHANGES"],
    25 : ["DISORDER", "LACK OF CONCENTRATION"],
    28 : ["DISORDER", "CHANGE IN REM SLEEP"],
    31 : ["DISORDER", "UNEXPLAINABLE BODY ACHES"],
    34 : ["DISORDER", "FEAR"],
    37 : ["DISORDER", "DEHYDRATION"],
    40 : ["DISORDER", "SWEATING"],
    43 : ["DISORDER", "POUNDING HEART RATE"],
    46 : ["DISORDER", "GENERAL DISCONTENT"],
    49 : ["DISORDER", "DEREALIZATION"],
    52 : ["DISORDER", "NONSENSE WORD REPETITION"],
    55 : ["DISORDER", "MANIA"]
}


#DISORDER_LIST is a list that will contain all disorders that needs to be diagnosed by the Backward chaining algorithm
DISORDER_LIST = ["BIPOLAR DISORDER", "SCHIZOPHRENIA", "SCHIZOAFFECTIVE DISORDER", "MAJOR
DEPRESSIVE DISORDER",
 "PANIC DISORDER WITH AGORAPHOBIA", "DISSOCIATIVE IDENTITY DISORDER", "DYSTHYMIA",
"GENERALIZED ANXIETY DISORDER",
 "DEMENTIA", "POST TRAUMATIC STRESS DISORDER", "OBSESSIVE COMPULSIVE DISORDER",
"PSYCHOSIS", "BODY DISMORPHIC DISORDER",
 "INSOMNIA", "NARCOLEPSY", "BORDERLINE PERSONALITY DISORDER", "ALZHEIMERS DISEASE",
"BULIMIA NERVOSA", "KLEPTOMANIA"]

# KNOWLEDGE BASE & VARIABLE LIST are created as seperate JSON files for better readability and access

# The output of backward chaining algorithm(i.e, disorder) is stored in backward_conclusions
# The output of forward chaining algorithm(i.e, Treatment ) is stored in forward_conclusions
backward_conclusions = None
forward_conclusions=None

# derived variable of backward and forward chaining
DERIVED_VARIABLE_LIST={}
DERIVED_FORWARD_VARIABLE_LIST ={}


"""
********** Function Name: main() ********************************
# This function first configures the logging where we can store the logs in
Project1_A05252367_ITERATION_DETAILS.log file
# Then it calls the Backward chaining algorithm's process function and process returns the output(disorder is returned after
process BW algorithm)
# The output of backward chaining(BW) is sent as input to the forwarding chaining algorithm's process function.
# The process function of FW returns the output(treatment is returned after processing the FW algorithm)
# Time and space calculated and displayed.
**************************************************************************
"""

def main():

    # Create and configure logger
    logging.basicConfig(filename="Project1_A05252367_ITERATION_DETAILS.log",format='%(asctime)s
%(message)s',filemode='w')
    # Creating an object
    LOG = logging.getLogger()
```

```
    # Setting the threshold of logger to DEBUG
    LOG.setLevel(logging.DEBUG)

    LOG.info("PROGRAM START")
    LOG.info("DEFINING THE GOAL VARIABLE AS DISORDER FOR BACKWARD CHAINING")
    LOG.info("CALLING BACKWARD CHAINING PROCESS FUNCTION")


    print("\n**********Kindly input YES or NO for each question**********\n")
    goal_variable = "DISORDER"

    #recording the start time to calculate the time taken by the backward chaining
    start_time_bw = time.perf_counter()

    # calling the backward chaining algorithm and output is stored disorder
    disorder = BW.process(goal_variable)

    #ending the timer as the backward chaining algorithm execution is completed
    end_time_bw = time.perf_counter()

    LOG.info("THIS IS THE GOAL IDENTIFIED BY BACKWARD CHAINING ALGORITHM :%s "% disorder)
    LOG.info("PASSING THE DISORDER TO FORWARD CHAINING AND CALLING THE FORWARD CHAINING
PROCESS FUNCTION")


    # calculating time for BW by subtracting end time and start time
    print(f"\nTime Elapsed for Backward chaining : {end_time_bw - start_time_bw:0.2f} Secs")

    #checking whether person is having any disorder or not condition will be true if he is having some disorder else no
disorder
    if disorder in DISORDER_LIST:
        print("You are suffering from  : ", disorder ,"\n")

        # starting another timer to calculate the time taken by the forward chaining
        start_time_fw = time.perf_counter()

        # calling the forward chaining algorithm and output is stored treatment
        treatment = FW.process(disorder)

        #ending the timer as the forward chaining algorithm execution is completed
        end_time_fw = time.perf_counter()

        print("\nTreatment for" , disorder, " is : ",treatment)
        LOG.info("THIS IS THE GOAL IDENTIFIED BY FORWARD CHAINING ALGORITHM :%s "% treatment)

        # calculating time for FW by subtracting end time and start time
        print(f"\nTime Elapsed for Forward chaining :  {end_time_fw - start_time_fw:0.2f} Secs")
    else:
        print("You are not suffering from any disorder so no treatment required")
        LOG.info("No treatment required")

    # calculating the memory consumed by using psutil library in python
    memory = psutil.Process().memory_info().rss / (1024 * 1024)
    print("Memory consumed : ", memory, "MB\n")

# This calls the main function
if __name__ == "__main__":
    main()
```

7.3 Forward Chaining

Project1_A05252367_Drug_FW.py:

```
"""
we have implemented by using option#1 that is we used algorithm for building the AI expert system

********** File name:Project1_A05252367_Drug_FW.py **********

# This file is implementation forward chaining algorithm.
# This is called once Project1_A05252367_main.py  execution starts and Project1_A05252367_Disorder_BW.py is
completed.
# it import Project1_A05252367_main.py to use all the variables declared
# knowledge base and variable list json files stored as dictionary
# The execution starts from process function where diagnosed disorder which is
# returned by process function in backward chaining algorithm is passed as input forward chaining process function from
main

"""

import Project1_A05252367_main  as M
import json
import logging

# Creating an object
LOG = logging.getLogger()
LOG.setLevel(logging.DEBUG)

# goal variable is stored
global_disorder_value : None

# conclusions and its value is stored
global_conclusion_variable_queue=[]

#ALL THE JSON FILES ARE LOADED HERE
json_file1 = open('Project1_A05252367_FW_KNOWLEDGE_BASE.json', "r")
forward_kb_rules = json.load(json_file1)
json_file1.close()
json_file2 = open('Project1_A05252367_FW_VARIABLE_LIST.json', "r")
forward_variable_list = json.load(json_file2)
json_file2.close()

# This variable stores all the clause numbers which are executed by validate function
visited_clause =[]

"""
*****************************************************************************
# search_con(): This function searches goal_variable in FORWARD_CLAUSE_VARIABLE_LIST.
# If the matching goal is found then it checks corresponding clause_number
# whether it is already visited or not by using visited_clause.
# If not visited then it return clause_number
# else checks for another matching goal and process repeats again
# goal_variable: It is the string(goal) where it needs to be searched in FORWARD_CLAUSE_VARIABLE_LIST.
# clause number is returned to process function
*****************************************************************************
"""
def search_cvl(goal_variable): #get's the goal variable from the main function

    LOG.info("INSIDE SEARCH_CVL FUCNTION WITH GOAL VARIABLE :%s" % goal_variable)
    # assigning some invalid rule number if no rule is found this will be returned
    clause_num=-1

    # In FORWARD_CLAUSE_VARIABLE_LIST every value is checked for matching goal
    for key, value in M.FORWARD_CLAUSE_VARIABLE_LIST.items():
```

```
        # checking goal variable with variable in FORWARD_CLAUSE_VARIABLE_LIST and the corresponding
clause_num should not be visited
        if goal_variable in value and key not in visited_clause:
            #store the clause number and breaks from loop
            clause_num=key
            break


    LOG.info("visited clause: %s" %visited_clause)
    LOG.info("THIS IS THE CLAUSE NUMBER TO BE PASSED TO NEXT FUNCTION:%s" % clause_num)

    #returns clause number
    return clause_num


"""
*****************************************************************************
# update_VL: This function asks the user several questions
# It asks questions to get the values for the variables present in CLAUSE_VARIABLE_LIST for that clause number.
# The users input is stored in VARIABLE_LIST. But if the variable already instantiated then it just skips to ask that question.
# It also adds every clause number that is visited into visited_rules list which will later used in search_con to
# track clause_numbers which are already executed.
# Every time we get answer from user then it is stored in DERIVED_VARIABLE_LIST
# clause_number : It is output returned by search_con function and it is passed as input to the update_VL
*****************************************************************************
"""

def update_VL(clause_number:int):
    LOG.info("INSIDE THE UPDATE_VL FUNCTION WITH CLAUSE NUMBER :%s " % clause_number)

    #stores clause variable list corresponding to given clause number
    fw_temp_clause_list=M.FORWARD_CLAUSE_VARIABLE_LIST[clause_number]

    #append that clause numer into visited_clause list
    visited_clause.append(clause_number)

    #checking if the variable is instantiated in the variable list or not. If not, it will ask the user to provide the values of
variables and instantiate them.
    for i in range(len(fw_temp_clause_list)):
        if fw_temp_clause_list[i] in forward_variable_list and forward_variable_list[fw_temp_clause_list[i]]["Userinput"]=="":
            while(1):
                # Asking the user questions regarding the  symptoms which will be "yes" or "no"
                inputvariable = input(forward_variable_list[fw_temp_clause_list[i]]['Question'] +" "+fw_temp_clause_list[i]+"? ")

                # checking if the user input is "yes" or "no" if he enters other than these.
                # Same question will be asked again
                if inputvariable.lower() in  ["yes","no"]:
                    forward_variable_list[fw_temp_clause_list[i]]["Userinput"]  = inputvariable.lower()
                    break

            #forward_variable_list[fw_temp_clause_list[i]]["Userinput"] =
input(forward_variable_list[fw_temp_clause_list[i]]['Question'] +" "+fw_temp_clause_list[i]+"? ")
            M.DERIVED_FORWARD_VARIABLE_LIST[fw_temp_clause_list[i]] =
forward_variable_list[fw_temp_clause_list[i]]["Userinput"]
    LOG.info("UPDATING THE FORWARD DERIVED VARIABLE LIST AS :%s "%
M.DERIVED_FORWARD_VARIABLE_LIST)



"""
*****************************************************************************
# rule_to_clause(): This function converts clause number to rule number
```

```
# If rule numbers are in the pattern 1,2,3,4... then Rule number = {(Quotient (clause number/3))} +1)
# clause number: This is an integer which is calculated by search_con function and
# sent as input to clause_to_rule() function
# rule number is calculated by using the formula and it is returned to process function
**************************************************************************
"""
def clause_to_rule(clause_number:int):
   LOG.info("INSIDE THE CLAUSE_TO_RULE FUNCTION WITH CLAUSE NUMBER :%s " % clause_number)

   #formula for calculating rule number
   rule_number = int(clause_number//3)+1

   LOG.info("THIS IS THE CALCULATED RULE NUMBER :%s "% rule_number)

   # rule number is returned
   return rule_number

"""
**************************************************************************
# validate_ri(): This function checks the ri rule in kb_rules with the user input present in VARIABLE_LIST.
# Once it satisfies the kb rules then it will return corresponding conclusion else it will return None.
# FUNCTION INPUTS:
# ri : Rule number that we need to validate
# conclusion : it is just None value
# FUNCTION RETURN:
# It will return conclusion variable. If the rule is satisfied it will return conclusion in kb_rules
# else returns None
**************************************************************************
"""

def validate_ri(ri:int):

   LOG.info("INSIDE UPDATE VALIDATE_RULE FUCNTION WITH RULE NUMNER :%s "% ri)
   rule_num = str(ri)

   # A local variable which is created to store the variables used in the rule
   symptoms_list=list(forward_kb_rules[rule_num]['SYMPTOMS'].keys())

   LOG.info("PRINTING THE SYMPTOM PRESENT IN FORWARD KNOWLEGE BASE :%s "% symptoms_list)

   # A flag to track whether the rule is satisfied or not
   flag=0

   # checks each variable in kb rule(ri) with userInput if there is any mismatch loop breaks and returns None
   # else assigns conclusion with conclusion in kb_rule(ri) and then return conclusion
   for symptom in symptoms_list:

      if((symptom  in forward_variable_list and forward_kb_rules[rule_num]['SYMPTOMS'][symptom] ==
forward_variable_list[symptom]['Userinput'])):
         continue
      else:

         #Rule is not satisfied
         flag=1
         break

   # flag =0 means the rule(ri) is satisfied and returns the conclusion
   if(flag == 0):

      LOG.info("RULE IS SATISFIED AND THE TREATMENT RETUREND IS :%s"%
forward_kb_rules[rule_num]['TREATMENT'])
```

```
        # rule is satisfied and appends the conclusion into global_conclusion_variable
        global_conclusion_variable_queue.append(forward_kb_rules[rule_num]['TREATMENT'])

        return forward_kb_rules[rule_num]['TREATMENT']

    LOG.info("This is validate return value:%s"%forward_kb_rules[rule_num]['TREATMENT'])
    return None



"""
*****************************************************************************
# process(): This function just process the goal by calling search_con, update_VL, clause_to_rule and validate_ri.
# The execution continues until goal is reached or if the user is not suffering then loop breaks.
# "DISORDER" in Forward_variable_list is initialized with backward chaining algorithm output
# goal: Initially goal will be "DISORDER" and the process starts from this.
# returns conclusion that is some treatment to the disorder the user is suffering from.
*****************************************************************************
"""
def process(variable:str):
    LOG.info("INSIDE THE FORWARD PROCESS FUNCTION WITH GLOBAL VARIABLE :%s" % variable)

    # loop continues until conclusion is determined
    while(M.forward_conclusions==None):
        # assigning the backward chaining output to the variable_list
        forward_variable_list["DISORDER"]["Userinput"]=variable
        global_disorder_value = "DISORDER"

        # calling search_con function to find the clause number to the goal variable
        clause_num = search_cvl(global_disorder_value)

        # calling update_vl function to ask the userinput for that clause number
        update_VL(clause_num)

        # The clause is passed as input to clause_to_rule and rule number is returned
        rule_num=clause_to_rule(clause_num)

        # once input taken from user validation of that rule  is done by calling validate_ri with rule ri and conclusion None as
input
        M.forward_conclusions = validate_ri(rule_num)


    # loading the conclusion value into DERIVED_VARIABLE_LIST
    M.DERIVED_FORWARD_VARIABLE_LIST["TREATMENT"] = M.forward_conclusions

    # loading the DERIVED_VARIABLE_LIST into json file
    json_file = open('Project1_A05252367_FW_DERIVED_VARIABLE_LIST.json', "w")
    json.dump(M.DERIVED_FORWARD_VARIABLE_LIST, json_file, indent=6)
    json_file.close()

    #returning conclusion of backward_chaining that is goal value
    return M.forward_conclusions
```

Project1_A05252367_FW_VARIABLE_LIST.json

This file contains all the variable which is nothing but symptoms and the user response is stored in this file

```
{

  "DISORDER":{
     "Userinput":""
  },
```

```
"THOUGHT DISORDER":{
   "Question":"Do you have ",
   "Userinput":""
},
"COMPULSIVE HOARDING":{
   "Question":"Do you have ",
   "Userinput":""
},
"URGE TO STEAL":{
   "Question":"Do you have ",
   "Userinput":""
},
"TALKATIVENESS":{
   "Question":"Do you have ",
   "Userinput":""
},
"LOSS OF INTEREST":{
   "Question":"Do you have ",
   "Userinput":""
},
"INDECISIVENESS":{
   "Question":"Do you have ",
   "Userinput":""
},
"MEMORY LOSS":{
   "Question":"Do you have ",
   "Userinput":""
},
"PERSONALITY CHANGES":{
   "Question":"Do you have ",
   "Userinput":""
},
"LACK OF CONCENTRATION":{
   "Question":"Do you have ",
   "Userinput":""
},
"CHANGE IN REM SLEEP":{
   "Question":"Do you have ",
   "Userinput":""
},
"UNEXPLAINABLE BODY ACHES":{
   "Question":"Do you have ",
   "Userinput":""
},
"FEAR":{
   "Question":"Do you have ",
   "Userinput":""
},
"DEHYDRATION":{
   "Question":"Do you have ",
   "Userinput":""
},
"SWEATING":{
   "Question":"Do you have ",
   "Userinput":""
},
"POUNDING HEART RATE":{
   "Question":"Do you have ",
   "Userinput":""
},
"GENERAL DISCONTENT":{
   "Question":"Do you have ",
```

```
      "Userinput":""
    },
    "DEREALIZATION":{
      "Question":"Do you have ",
      "Userinput":""
    },
    "NONSENSE WORD REPETITION":{
      "Question":"Do you have ",
      "Userinput":""
    },
    "MANIA":{
      "Question":"Do you have ",
      "Userinput":""
    }
}
```

Project1_A05252367_FW_DERIVED_VARIABLE_LIST.json
This file contains user response and system generated output

```
{
    "THOUGHT DISORDER": "yes",
    "COMPULSIVE HOARDING": "yes",
    "TREATMENT": [
        "CLOMIPRAMINE",
        "FLUOXETINE",
        "AVERSION_THERAPY"
    ]
}
```

Project1_A05252367_FW_KNOWLEDGE_BASE.json
This file is knowledge base which contains rules

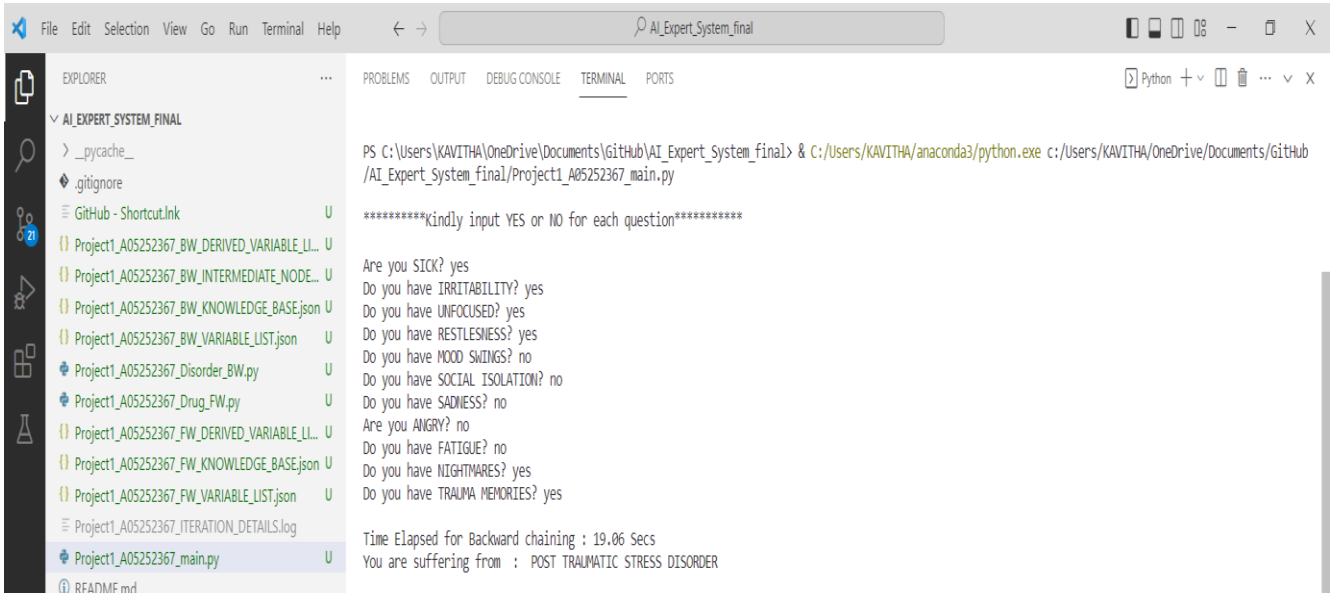
```
{
  "1":{
    "TREATMENT": ["ANTI_ANXIETY_DRUG", "PSYCHOEDUCATION"],
    "SYMPTOMS":{
      "DISORDER":"SCHIZOPHRENIA",
      "THOUGHT DISORDER" :"yes"
    }
  },
  "2":{
    "TREATMENT": ["CLOMIPRAMINE", "FLUOXETINE", "AVERSION_THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"OBSESSIVE COMPULSIVE DISORDER",
      "COMPULSIVE HOARDING" :"yes"
    }
  },
  "3":{
    "TREATMENT": ["ESCITALOPRAM", "PAROXETINE", "SYSTEMIC_DESENSITIZATION"],
    "SYMPTOMS":{
      "DISORDER":"KLEPTOMANIA",
      "URGE TO STEAL" :"yes"
    }
  },
  "4":{
    "TREATMENT": ["MOOD_STABILIZERS", "ANTIPSYCHOTIC_DRUGS", "SOCIAL_RYTHM_THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"BIPOLAR DISORDER",
      "TALKATIVENESS" :"yes"
    }
  },
  "5":{
```

```
    "TREATMENT": ["ANTIPRESSANTS", "ANTIPSYCHOTICS", "DIALECTICAL_BEHAVIOR_THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"BORDERLINE PERSONALITY DISORDER",
      "LOSS OF INTEREST" :"yes"
    }
  },
  "6":{
    "TREATMENT": ["SEROTONIN", "SNRIs", "INTERPERSONAL_PSYCHOTHERAPY"],
    "SYMPTOMS":{
      "DISORDER":"DYSTHYMIA",
      "INDECISIVENESS" :"yes"
    }
  },
  "7":{
    "TREATMENT": ["RAZADYNE", "GALANAMINE", "COGNITIVE BEHAVIOUR THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"DEMENTIA",
      "MEMORY LOSS" :"yes"
    }
  },
  "8":{
    "TREATMENT": ["MEMANTINE", "RIVASTIGIMINE", "EXCERCISE", "COGNITION ENHANCING
MEDITATION"],
    "SYMPTOMS":{
      "DISORDER":"ALZHEIMERS DISEASE",
      "PERSONALITY CHANGES" :"yes"
    }
  },
  "9":{
    "TREATMENT": ["TRAZODONE", "LORAZEPEN", "ZOLPIDEIM", "LIGHT THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"INSOMNIA",
      "LACK OF CONCENTRATION" :"yes"
    }
  },
  "10":{
    "TREATMENT": ["MODAFINIL", "SODIUM OXYBATE", "BEHAVIOURAL TREATMENT"],
    "SYMPTOMS":{
      "DISORDER":"NARCOLEPSY",
      "CHANGE IN REM SLEEP" :"yes"
    }
  },
  "11":{
    "TREATMENT": ["TRINTELLIX", "TOFRANIL", "VIVACITY", "ELECTROCONVULSIVE THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"MAJOR DEPRESSIVE DISORDER",
      "UNEXPLAINABLE BODY ACHES" :"yes"
    }
  },
  "12":{
    "TREATMENT": ["SERATINE", "AROXENTINE", "EYE MOVEMENT DESENSITIZATION AND
REPROCESSING"],
    "SYMPTOMS":{
      "DISORDER":"POST TRAUMATIC STRESS DISORDER",
      "FEAR" :"yes"
    }
  },
  "13":{
    "TREATMENT": ["FLUOXETINE", "COUNSELLING THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"BULIMIA NERVOSA",
      "DEHYDRATION" :"yes"
```

```
    }
  },
  "14":{
    "TREATMENT": ["SSRIs", "ANXIOLYTICS", "PSYCHOANALYSIS"],
    "SYMPTOMS":{
      "DISORDER":"GENERALIZED ANXIETY DISORDER",
      "SWEATING" :"yes"
    }
  },
  "15":{
    "TREATMENT": ["BENZODIAZEPINES", "PAXIL", "EYE MOVEMENT DESENSITIZATION AND
REPROCESSING"],
    "SYMPTOMS":{
      "DISORDER":"PANIC DISORDER WITH AGORAPHOBIA",
      "POUNDING HEART RATE" :"yes"
    }
  },
  "16":{
    "TREATMENT": ["PROZAC", "ZOLOFT", "PSYCHOEDUCATION"],
    "SYMPTOMS":{
      "DISORDER":"BODY DISMORPHIC DISORDER",
      "GENERAL DISCONTENT" :"yes"
    }
  },
  "17":{
    "TREATMENT": ["HYPNOTHERAPY", "ADJUNCTIVE THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"DISSOCIATIVE IDENTITY DISORDER",
      "DEREALIZATION" :"yes"
    }
  },
  "18":{
    "TREATMENT": ["RISPERIDONE", "QUETIAPINE", "FAMILY THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"PSYCHOSIS",
      "NONSENSE WORD REPETITION" :"yes"
    }
  },
  "19":{
    "TREATMENT": ["ANTIPSYCHOTIC", "PYSCHOTHERAPY", "SOCIAL RHYTHM THERAPY"],
    "SYMPTOMS":{
      "DISORDER":"SCHIZOAFFECTIVE DISORDER",
      "MANIA" :"yes"
    }
  }
}
```

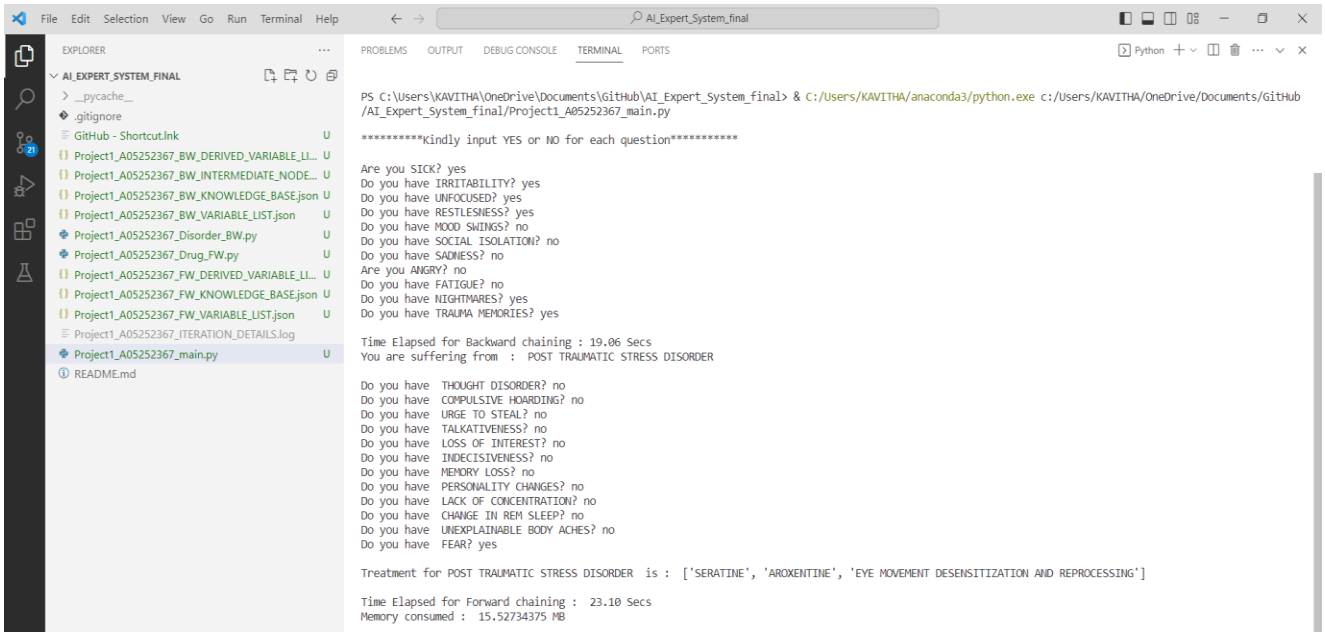
8. PROGRAM EXECUTION RESULTS

Output generated after running Backward chaining algorithm is -



Screen 1. Backward chaining output

Output generated after running forward chaining algorithm is –



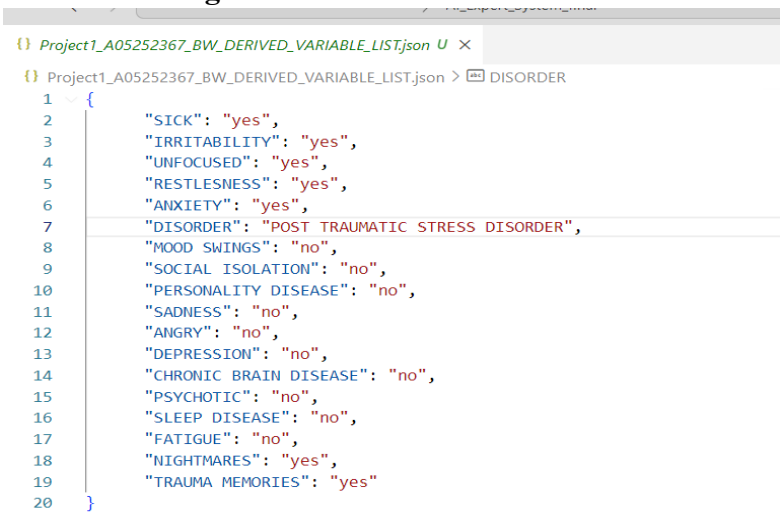
Screen 2. Forward chaining output

Along with this output we have interface log files -

Screen 3. Interface Log files

Derived variable list is stored as json

1. **Backward chaining derived variable list**



Screen 4. Backward derived variable list

2. **Forward chaining derived variable list**

Screen 5. Forward derived variable list

9. ANALYSIS OF THE PROGRAM

The main significance of this project is to develop an intelligent expert system which diagnoses the mental disorder by asking the user several questions and based on the diagnosed disease, the treatment is determined. Implementation of this project is done in python programming. The main functionality for building the expert system is to develop a knowledge base or the rules based on the facts. Once this is done an inference engine is developed which can process these rules. Through the user interface the user provides responses to the questions asked by the inference engine. Based on user response, conclusion is determined.

Below changes have improved the program execution time:

1. Knowledge base and variable list are stored as Json files for better readability and these are key, value pairs. By using Json, we can access by using keys directly and it reduces the time for accessing rules and variables rather than using lists which consumes a lot of time for fetching the data.

2. search_con function of both backward and forward chaining, we must decide on which rule or clause number to return. If a rule or clause number is already executed then for the next call it should return a next occurrence. In order to handle this, we have just included visited_rules or visited_clause variables which will maintain executed rule or clause numbers. So, this will ease the process of finding the rule number or clause number.

3. In Backward chaining algorithm, initially developed code is used to ask the user multiple questions for every rule even though intermediate node is evaluated as “no” the process continue to ask questions for the rules that include intermediate node as “yes” but it is clearly says that rule will not be satisfied and asking user input for remaining variables is not necessary. In order to overcome asking multiple questions to the user, we have included one condition whenever we encounter the intermediate node which is evaluated as “no” then update_VL(clause_number) function will just return nothing and skips asking the questions. This avoids asking unnecessary questions to users.

4. Similarly, if an intermediate node is already evaluated as “yes” or “no” then it should not call the process function again for the next time.

5. Furthermore, “ANXIETY” is a common symptom for all the disorders that we have considered for this project. If the user is not having “ANXIETY” then he will not be having any disorder. In order to handle this in the process function of backward chaining we have added that if ANXIETY is evaluated as “no” then the loop will break and none is returned to main.

10. ANALYSIS OF THE RESULTS

The results of the program are achieved as follows-

The program detects the type of mental disorder the user is having and this is based on the response given by the user. This diagnosed disorder will be used to suggest treatment for the user. The following modifications have affected program results.

For example1: Consider diagnosis of Kleptomania disorder.

- If a user is suffering from kleptomania, then he will be answering yes for the symptoms- Irritability, unfocused, restlessness, sadness, anger, urge to steal items, pleasure after stealing items, guilt. So, for this disorder we have 3 intermediate nodes that will be processed: anxiety, personality disease, depression.
- First the user will be asked whether he is sick or not.
- If this response is yes then he will be asked for further symptoms (irritability, unfocused, restlessness) as the user is having these symptoms the rule will be satisfied and control reaches the anxiety intermediate node. As the anxiety intermediate node is evaluated as yes then next questions will be on mood swings, social isolation. As users not suffering from these symptoms and the personality disease intermediate node will be evaluated as no.

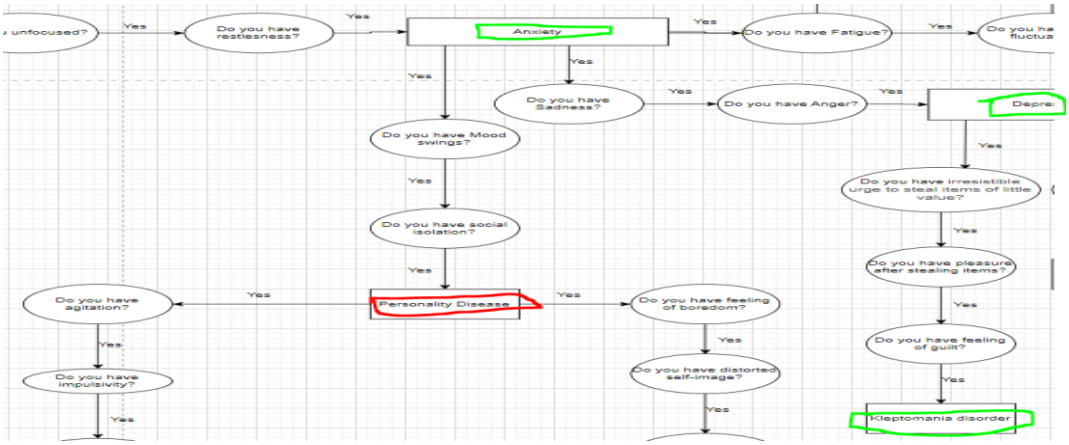


Fig. 3 Kleptomania disorder

- Due to modification the user will not be asked questions that involve the rule which include personality disease intermediate node which is shown in above decision tree. This improves the expert system by not asking unnecessary questions (questions like agitation and boredom).
- Now, the user will be asked questions on anger, sadness, urge to steal items, pleasure after stealing items, guilt. For all these questions user response is yes and kleptomania disorder is returned to the main function.

Example 2:

Consider below rules

1. IF SICK=NO
   THEN DISORDER=NO
2. IF SICK=YES AND IRRITABILITY=YES AND UNFOCUSED=YES AND RESTLESSNESS=YES
   THEN ANXIETY=YES
3. IF ANXIETY=YES AND MOODSWING=YES AND SOCIAL_ISOLATION=YES
   THEN PERSONALITY_DISEASE=YES
4. IF PERSONALITY_DISEASE=YES AND AGITATION=YES AND IMPULSIVITY=YES AND COMPULSIVE_BEHAVIOUR=YES AND

HYPERVIGILANCE=YES AND RITUALISTIC_BEHAVIOUR =YES AND REPETITIVE_BEHAVIOUR=YES
THEN DISORDER=OCD

- Firstly, the process function of the backward chaining algorithm called search_con("DISORDER") will return a rule number which is 1.
- Suppose if rule 1 is not satisfied, then for next time search_con("DISORDER") should return rule number 4 instead of 1 as rule 1 is already validated.
- So, this is implemented in the program by introducing visited_rule data structure which stores the rules that are validated.
- Similarly, in the forward chaining algorithm visited_clause data structure is maintained which stores the clause that are already visited.

11. CONCLUSION

Finally, the development of an intelligent computer expert system for diagnosing and proposing treatment for mental diseases is a huge step forward in the field of mental healthcare. We set out with this initiative to address the critical need for accurate and rapid diagnosis and treatment recommendations for people dealing with a variety of mental health issues.

The experiment met its aims, which included the diagnosis of 19 different mental diseases. This accomplishment was made possible by intensive study, knowledge gathering, and the deployment of powerful AI-driven inference methods, including Backward Chaining and Forward Chaining.

This project's expert system not only correctly identified mental diseases based on patient symptoms, but it also offered evidence-based treatment alternatives customized to each diagnosis. This comprehensive strategy has the potential to transform the way mental healthcare is delivered, ensuring that people receive prompt and appropriate care.

In addition, the project emphasizes the significance of association between technology and healthcare. We have taken a huge step in improving patient outcomes and reducing the burden on mental health practitioners by leveraging the capabilities of artificial intelligence and data-driven decision-making.

This expert system's influence extends beyond the scope of this project. Its use in real-world clinical settings has the potential to speed the diagnostic process, eliminate errors, and ultimately improve the quality of care delivered to those dealing with mental health issues. It provides mental health clinic professionals with a helpful tool that supplements their experience, allowing them to make more informed patient care decisions.

As with any pioneering technology, there is always room for improvement and extension. Continuous knowledge updates, refining the system's user interface, and expanding the variety of mental diseases addressed may be part of future effort. Nonetheless, the project's foundation is an important step toward a more accessible, efficient, and effective mental healthcare system.

To summarize, the intelligent computer expert system created in this project has the potential to make a significant difference in the lives of people dealing with mental health concerns. We

have set the road for a brighter and more positive future for those in need by integrating cutting-edge technology with a commitment to improving mental healthcare.

12. TEAM MEMBER CONTRIBUTIONS

12.1 Kavitha Lodagala

- Learned how to use visual studio IDE.
- Took initiative in improving the code and this proactive approach has benefited the project.
- Played a crucial role in solving problems or overcoming the obstacles within the project.
- Learned how to use Github (pulling and pushing the developed code).
- Tested the code for all the rules. In addition to these tested negative scenarios.
- Completing the tasks ahead of scheduled time and solving challenging problems.
- shared the knowledge and supported team members.
- Took additional responsibility for extra work in order to meet the project deadline.
- Finding the relevant data for the project
- Worked on both backward chaining and forward chaining algorithm
- Constructed Decision tree and formed rules
- I have collaborated with the team and did the initial analysis.

12.2 Roopika Ganesh

- Actively contributed to the team's success through effective collaboration.
- Took additional responsibility for extra work in order to meet the project deadline.
- Played a significant role by setting meetings, assigning tasks to the team members.
- Effectively communicated with the team members and conveyed information clearly.
- Completing the tasks ahead of scheduled time.
- Shared the knowledge and supported team members.
- Finding the relevant data for the project
- Created Json files
- Worked on both backward chaining and forward chaining algorithm
- Constructed Decision tree and formed rules
- Tested the code for all the rules. In addition to these tested negative scenarios.

12.3 Sakshi Tripathi

- Completing the tasks ahead of scheduled time.
- Actively contributed to the team's success through effective collaboration.
- Shared the knowledge and supported team members.
- Finding the relevant data for the project
- Worked on both backward chaining and forward chaining algorithm
- Constructed Decision tree and formed rules
- I have collaborated with the team and did the initial analysis.
- Tested the code for all the rules. In addition to these tested negative scenarios.
- Actively contributed to the team's success through effective collaboration.

13 REFRENCES

[1] https://www.geeksforgeeks.org/read-json-file-using-python/

[2] https://www.w3schools.com/js/js_json_intro.asp

[3] https://pythonspeed.com/articles/measuring-memory-python/

[4] https://www.who.int/news-room/fact-sheets/detail/mental-disorders

[5] https://www.mayoclinic.org/diseases-conditions/mental-illness/symptoms-causes/syc-20374968

[6] https://psychcentral.com/conditions/conditions-index

[7] https://www.geeksforgeeks.org/time-perf_counter-function-in-python/