

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Knowledge Based System | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Basant Medhat Makram |  | | | |
|  | | | | —**Career Path Recommendation System**—Supervised by: Eng/ Ola Mahmoud |  | | | |
|  | | |  | | |  | | |
|  | | |  | | |  | | |
|  | | |  | | |  | | |

**1st: Introduction:**

Choosing a future career path is one of the most critical decisions students make during their school years. With numerous options and varying personal interests, skills, and academic strengths, many students struggle to identify the field that best aligns with their potential. To assist with this challenge, I developed an **Expert System for Student Career Path Recommendation** using artificial intelligence and decision-making logic.

Our system is designed to simulate a human career advisor by asking a series of targeted questions about the student's personality, interests, academic performance, and goals. Based on their responses, it uses **production rules, a knowledge-based system**, and a **decision tree logic** to recommend a suitable career field such as Engineering, Medicine, Accounting, or Field Research.

The goal is to provide school students with insightful, data-driven suggestions to support their career exploration at an early stage.

By integrating a simple and interactive GUI within Google Colab, we ensure that students can easily engage with the system and receive instant feedback. This project demonstrates how expert systems can be used effectively in the educational field to make complex decisions easier and more personalized.

**2nd: Analysis of the code of the project and it’s structure:**

**2.1**: **Knowledge Representation**:

* defined a StudentFacts class to store user responses as facts.
* Eight levels of questions were created to gather information.
  + Personality
  + Subject preference
  + GPA
  + Interest (people vs tech)
  + Skill
  + Motivation
  + Goal
  + Preferred Work Environment

**2.2: Rule Definition**:

* Using the @Rule decorator from the Experta library.
* The conditions consider attributes such as personality type, GPA, favorite subject, work environment preference, skills, and motivation.

**2.3: User Interface**:

* Implemented using Google Colab widgets (ipywidgets) for interactive question-and-answer flow.

**2.4: Inference Engine**:

* The system utilizes forward chaining to evaluate user-provided facts and match them against the defined rules.
* Once a rule fires, a personalized recommendation is shown, and the user can restart if they have no recommendations.

**2.5: code**:

! pip install experta  
!pip install frozendict==2.3.8

import ipywidgets as widgets

from IPython.display import display, clear\_output

from experta import Fact, Rule, KnowledgeEngine

user\_answers = {}

class StudentFacts(Fact): pass  
class CareerRecommender(KnowledgeEngine):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.recommendation = None

    @Rule(StudentFacts(personality='logical'),

          StudentFacts(subject='math'),

          StudentFacts(gpa='yes'),

          StudentFacts(interest='tech'),

          StudentFacts(skill='problem-solving'),

          StudentFacts(motivation='inventing'),

          StudentFacts(goal='stable'))

    def recommend\_engineering(self):

        self.recommendation = "Engineering"

    @Rule(StudentFacts(personality='emotional'),

          StudentFacts(subject='biology'),

          StudentFacts(gpa='yes'),

          StudentFacts(interest='people'),

          StudentFacts(skill='communication'),

          StudentFacts(motivation='helping'),

          StudentFacts(goal='help others'))

    def recommend\_medical(self):

        self.recommendation = "Medical"

    @Rule(StudentFacts(subject='accounting'),

          StudentFacts(skill='math'),

          StudentFacts(goal='own business'))

    def recommend\_accounting(self):

        self.recommendation = "Business"

        @Rule(StudentFacts(motivation='exploring'),

          StudentFacts(subject='history'),

          StudentFacts(goal='research'),

          StudentFacts(environment='outdoor'))

    def recommend\_explorer(self):

        self.recommendation = "Explorer (Field Research or Nature Work)"

questions = [

    {"key": "personality", "text": "What's your personality type?", "options": ['logical', 'emotional', 'social', 'practical']},

    {"key": "subject", "text": "What's your favorite subject?", "options": ['math', 'biology', 'accounting', 'history']},

    {"key": "gpa", "text": "Is your GPA high (above 3.5)?", "options": ['yes', 'no', 'maybe']},

    {"key": "interest", "text": "Do you prefer working with people or technology?", "options": ['people', 'tech', 'both']},

    {"key": "skill", "text": "What’s your strength?", "options": ['problem-solving', 'creativity', 'communication', 'math']},

    {"key": "motivation", "text": "What motivates you more?", "options": ['helping', 'inventing', 'exploring']},

    {"key": "goal", "text": "What’s your long-term goal?", "options": ['stable', 'help others', 'research', 'own business']},

    {"key": "environment", "text": "Where do you prefer to work?", "options": ['indoor', 'outdoor', 'lab']},

]

current\_q = 0

def ask\_question(index=0):

    global current\_q

    current\_q = index

    clear\_output()

    if index >= len(questions):

        run\_engine()

        return

    q = questions[index]

    key = q["key"]

    question = q["text"]

    options = q["options"]

    header = widgets.HTML(f"<h4>Question {index+1} of {len(questions)}</h4>")

    label = widgets.Label(question)

    dropdown = widgets.RadioButtons(options=options)

    next\_btn = widgets.Button(description="Next", button\_style='primary')

    def on\_click(b):

        user\_answers[key] = dropdown.value

        ask\_question(index + 1)

    next\_btn.on\_click(on\_click)

    display(header, label, dropdown, next\_btn)

def run\_engine():

    clear\_output()

    engine = CareerRecommender()

    engine.reset()

    engine.declare(StudentFacts(\*\*user\_answers))

    engine.run()

    rec = engine.recommendation or "No clear recommendation found. Try different answers."

    display(widgets.HTML(f"<h2>Your Career Recommendation:</h2>"))

    display(widgets.HTML(f"<h3 style='color:blue'>{rec}</h3>"))

    restart\_btn = widgets.Button(description="Restart", button\_style='info')

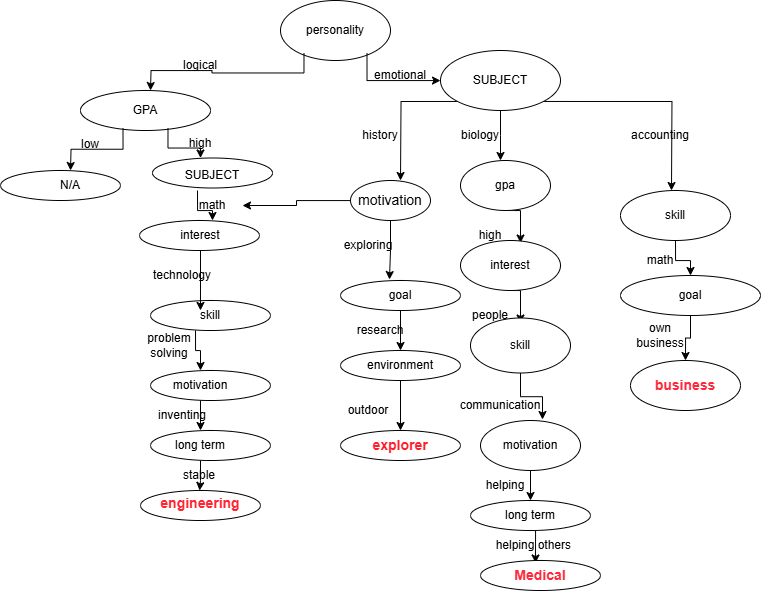
    restart\_btn.on\_click(lambda b: ask\_question(0))

    display(restart\_btn)

ask\_question(0)

**3rd: drawings and how it works:**

**3.1 Decision tree:**

****

**3.2 Production Rules:**

**1)** If Engineering Then

Personality is **logical**

And Favorite subject is **math**

And GPA is **high**

And Interest is in **technology**

And Skill is **problem-solving**

And Motivation is **inventing**

And Long-term goal is **stable**

**2) If** Medical Then

Personality is **emotional**

And Favorite subject is **biology**

And GPA is **high**

And Interest is in **people**

And Skill is **communication**

And Motivation is **helping**

And Long-term goal is **help others**

**3)** If Explorer/Researcher Then

subject is History

And Motivation is exploring

And goal is research

And environment is outdoor

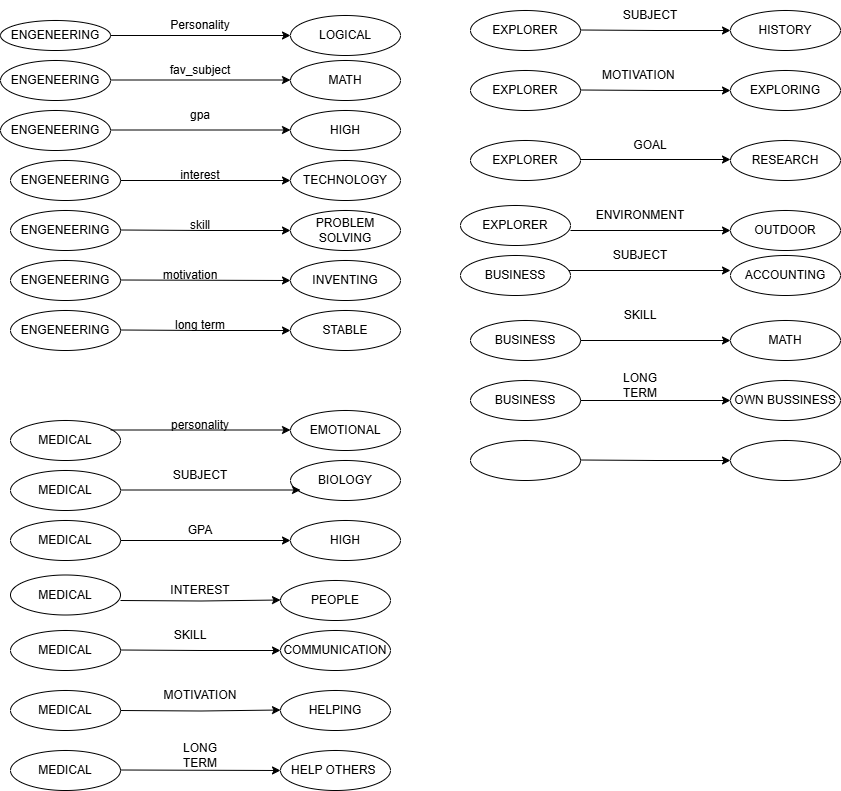
**4)** If Business Then

Favorite subject is **Accounting**

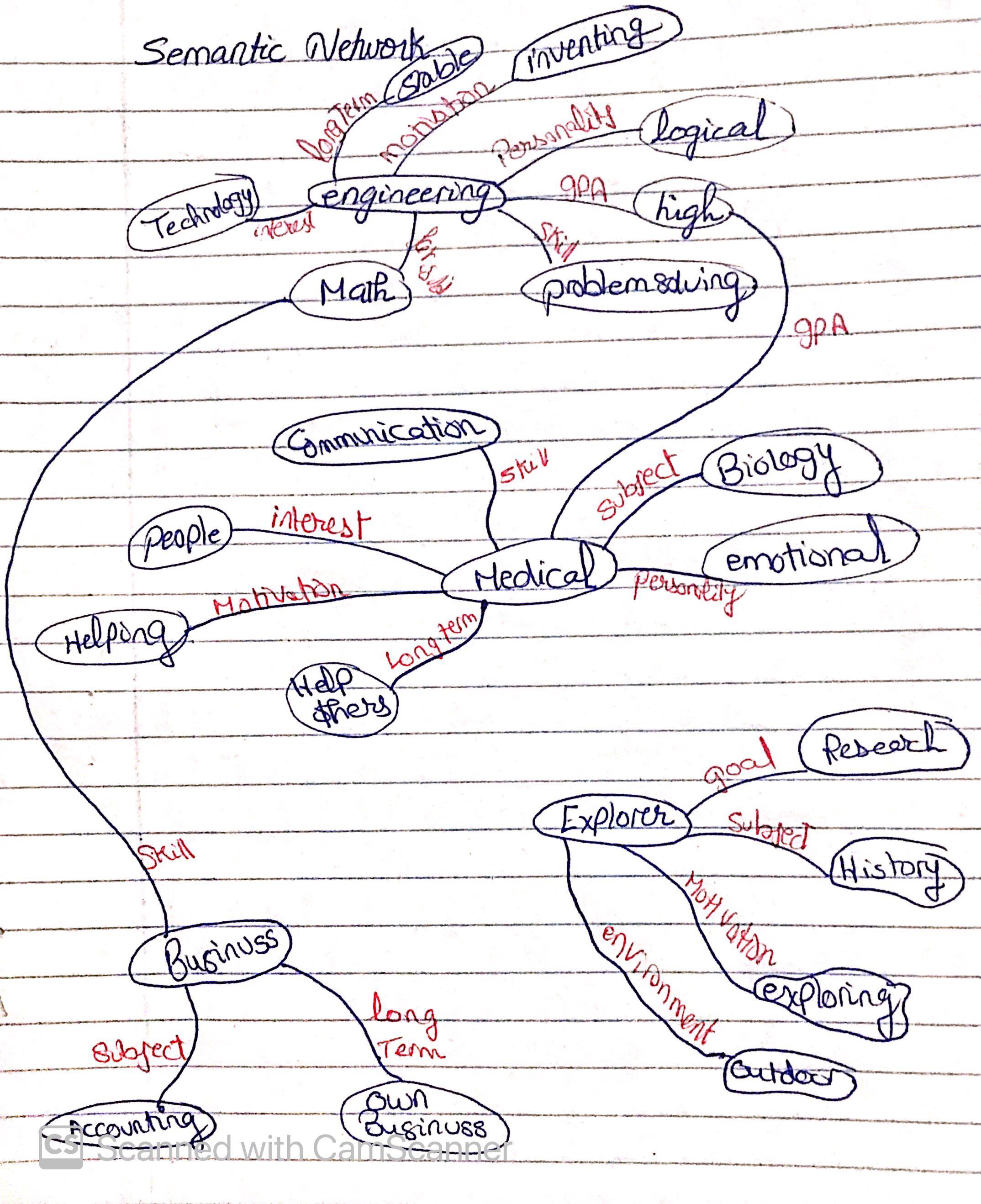
And Skill is **Math**

And Long-term goal is **own bussiness**

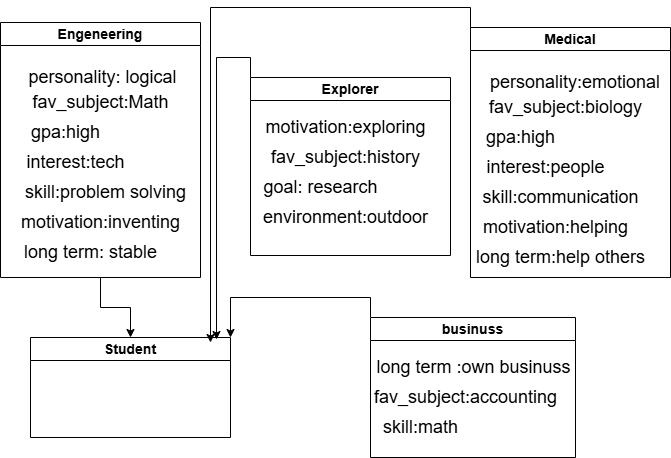
**3.3 OBJECT-ATTRIBUTE-VALUE (OAV):**

****

**3.4 Semantic Network:**

****

**3.5 Frames:**

****

**3.6 How the system works:**

* Student responses are stored as instances of the StudentFacts class.
* Rules are defined using the @Rule decorator from the Experta library.
* Each rule checks for specific combinations of responses and outputs a career suggestion.
* The system presents questions across **8** logical levels.
* Questions are implemented with ipywidgets for an interactive experience in a Jupyter/Colab notebook.
* The Experta engine uses forward chaining to match student facts with rule conditions.
* When a matching rule is found, the corresponding recommendation is stored and presented.
* The recommendation is displayed in a visually clear format.
* Students can restart the process to explore different paths.

**4th: Conclusion:**

This project successfully demonstrates the design and implementation of a knowledge-based expert system that simulates human decision-making to assist students in selecting a suitable career path. By incorporating eight structured levels of questions, the system collects detailed personal, academic, and behavioral data. Using the Experta library in Python, we translated this data into actionable recommendations through clearly defined rules and logical inference.

The interactive interface developed in Google Colab ensures an engaging and user-friendly experience. The use of forward chaining in the inference engine enables real-time evaluation. The system not only meets the functional requirements but is also modular and easily extendable, allowing for future enhancements with more careers, conditions, or even weighted decision-making.

In summary, this project provides a practical and scalable approach to career guidance for students, offering personalized support that can be refined and deployed in educational contexts.