AI AND ML ASSIGNMENT

In [1]:
```
# !pip install google-generativeai
# !pip install -q neo4j-driver
# !pip install -q gradio
```

In [2]:
```
import google.generativeai as palm
import base64
import json
import gradio as gr
from neo4j import GraphDatabase
import re
import os
```

In [3]:
```
palm.configure(api_key = "AIzaSyDANemDhHErQvNnBVuv4qSNM5itCAti6Gc")
```

FUNCTIONS

In [4]:

~~~
def get_answer(input):

    defaults = {
  'model': 'models/text-bison-001',
  'temperature': 0.7,
  'candidate_count': 1,
  'top_k': 40,
  'top_p': 0.95,
  'max_output_tokens': 1024,
  'stop_sequences': [],
  'safety_settings': [{"category":"HARM_CATEGORY_DEROGATORY","threshold":1
    }

    prompt = f"""You are an expert in converting English questions to Neo4

    All relationships ACTED_IN, DIRECTED, FOLLOWS, PRODUCED, REVIEWED, WRO

    For example,
    Example 1 - List down 5 movies that released after the year 2000, the
    ``` MATCH (m:Movie)
    WHERE m.released > 2000
    RETURN m LIMIT 5
    ~~~

    Example 2 - Get all the people who acted in a movie that was released
    ```
    MATCH (p:Person)-[r:ACTED_IN]-(m:Movie)
    WHERE m.released > 2010
    RETURN p,r,m
    ```

    Example 3 - Name the Director of the movie The Matrix Reloaded?
    ```
    MATCH (m:Movie)<-[:DIRECTED]-(p:Person)
    WHERE m.title = 'Apollo 13'
    RETURN p.name
    ```

    Dont include ``` and \n in the output

    {input}"""
    response = palm.generate_text(**defaults, prompt=prompt)
    return response.result
```
```

In [5]:
```
def extract_query_and_return_key(input_query_result):
    slash_n_pattern = r'[ \n]+'
    ret_pattern = r'RETURN\s+(.*)'
    replacement = ' '

    cleaned_query = re.sub(slash_n_pattern, replacement, input_query_resul
    if cleaned_query:
        match = re.search(ret_pattern, cleaned_query)
        if match:
            extracted_string = match.group(1)
        else:
            extracted_string = ""
    return cleaned_query, extracted_string
```

In [6]:
```
def format_names_with_ampersand(names):
    if len(names) == 0:
        return ""
    elif len(names) == 1:
        return names[0]
    else:
        formatted_names = ", ".join(names[:-1]) + " & " + names[-1]
        return formatted_names
```

In [7]:
```
def run_cypher_on_neo4j(inp_query, inp_key):
    out_list = []
    with driver.session() as session:
        result = session.run(inp_query)
        for record in result:
            out_list.append(record[inp_key])
    driver.close()
    if len(out_list) > 1:
        return format_names_with_ampersand(out_list)
    else:
        return out_list[0]
```

In [8]:
```
def generate_and_exec_cypher(input_query):
    gen_query, gen_key = extract_query_and_return_key(get_answer(input_que
    return run_cypher_on_neo4j(gen_query, gen_key)
```

In [9]:
```
def chatbot(input, history=[]):
    output = str(generate_and_exec_cypher(input))
    history.append((input, output))
    return history, history
```

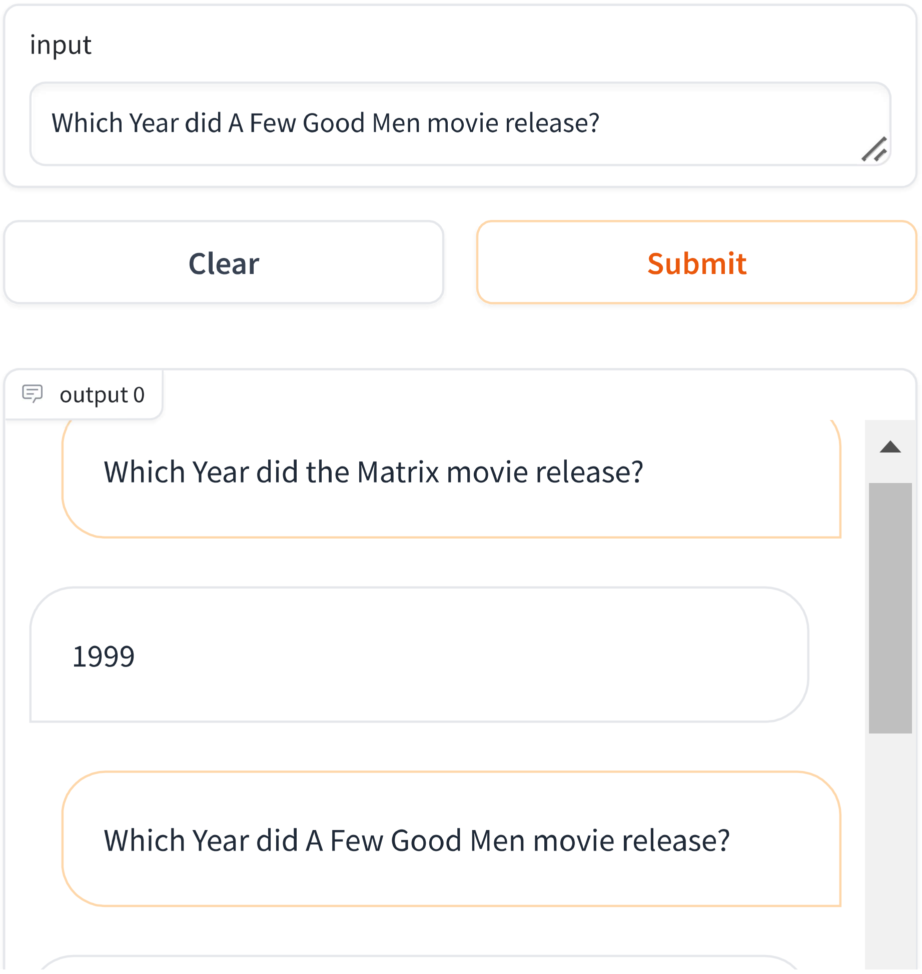
In [10]:
```
driver = GraphDatabase.driver("bolt://3.85.20.98:7687",
                              auth=("neo4j",
                                    "scopes-beat-detachments"))
```

INTERFACE

In [*]:
```
gr.Interface(fn = chatbot,
             inputs = ["text",'state'],
             outputs = ["chatbot",'state']).launch(debug = True)
```

```
Running on local URL:  http://127.0.0.1:7860 (http://127.0.0.1:7860)

To create a public link, set `share=True` in `launch()`.
```



Which Year did A Few Good Men movie release?

Which Year did the Matrix movie release?

1999

Which Year did A Few Good Men movie release?

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12900\2325344350.py:
3: DeprecationWarning: Using a driver after it has been closed is deprec
ated. Future versions of the driver will raise an error.
  with driver.session() as session:
```

In [ ]: