

Objective:

Using the capabilities of FlowiseAI create a fully automatic & autonomous script/mechanism that is writing content produced here (by Cognosys) and saving it to hdd as files in proper formats and proper location in the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT). It can also run the new/modified scripts 'on the fly' to test their practical functionality. Below are examples of scripts produced by FlowiseAI for agents that reads PDF and can save files to e:\ai
<https://github.com/UnifiedNeuralNetwork/NeuralGPT/blob/main/mmk%20Chatflow1.json>

```
-----  
  
<script type="module">  
import Chatbot from "https://cdn.jsdelivr.net/npm/flowise-embed@latest/dist/web.js"  
Chatbot.init({  
  chatflowid: "dd74f38b-a263-485b-b8fd-974b85ffe2dd",  
  apiHost: "http://localhost:3000",  
})  
</script>
```

----- (python)

```
import requests
```

```
API_URL = "http://localhost:3000/api/v1/prediction/dd74f38b-a263-485b-b8fd-974b85ffe2dd"
```

```
def query(payload):  
    response = requests.post(API_URL, json=payload)  
    return response.json()
```

```
output = query({  
    "question": "Hey, how are you?",  
})
```

----- (java)

```
async function query(data) {  
    const response = await fetch(  
        "http://localhost:3000/api/v1/prediction/dd74f38b-a263-485b-b8fd-974b85ffe2dd",  
        {  
            method: "POST",  
            body: data  
        }  
    );  
    const result = await response.json();
```

```
return result;
}
```

```
query({"question": "Hey, how are you?"}).then((response) => {
console.log(response);
});
```

```
-----(curl)
```

```
curl http://localhost:3000/api/v1/prediction/dd74f38b-a263-485b-b8fd-974b85ffe2dd \
-X POST \
-d '{"question": "Hey, how are you?"}'
```

Task:

1. Create a Python script that reads a PDF file from a specified location and saves it to the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT) in the PDF format. The script should be fully automatic and autonomous, meaning it should not require any user input once executed.

Result:

Here's a Python script that uses the PyPDF2 library to read a PDF file from a specified location and save it to the local clone of NeuralGPT repository in the PDF format:

```
import os
import shutil
import PyPDF2

pdf_path = 'path/to/pdf/file.pdf'
save_path = 'E:/AI/NeuralGPT/NeuralGPT'

# Check if the save path exists, create it if it doesn't
if not os.path.exists(save_path):
    os.makedirs(save_path)

# Open the PDF file in read-binary mode
with open(pdf_path, 'rb') as pdf_file:
    # Read the PDF file
    pdf_reader = PyPDF2.PdfFileReader(pdf_file)
    # Get the first page of the PDF
    page = pdf_reader.getPage(0)
    # Create a new PDF writer object
    pdf_writer = PyPDF2.PdfFileWriter()
    # Add the page to the PDF writer object
    pdf_writer.addPage(page)
    # Create a new PDF file name
    pdf_file_name = os.path.splitext(os.path.basename(pdf_path))[0] + '.pdf'
    # Save the PDF file to the specified location
    with open(os.path.join(save_path, pdf_file_name), 'wb') as new_pdf_file:
        pdf_writer.write(new_pdf_file)
```

Note that this script assumes that the specified PDF file exists and that the save path is valid. If either of these conditions is not met, the script will raise an exception.

Task:

2. Develop a Java program that reads a text file and converts it into a Markdown file. The program should be able to handle different text formats such as .txt, .docx, and .pdf. The Markdown file should be saved to a specified location in the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT).

Result:

Python script for Task 1:

```
import shutil
import os

pdf_path = "C:/path/to/pdf/file.pdf"
destination_folder = "E:/AI/NeuralGPT/NeuralGPT"

shutil.copy(pdf_path, destination_folder)
```

Java program for Task 2:

```
import org.apache.commons.io.FilenameUtils;
import org.commonmark.node.Node;
import org.commonmark.parser.Parser;
import org.commonmark.renderer.html.HtmlRenderer;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class TextToMarkdownConverter {

    public static void main(String[] args) throws IOException {
        String textFilePath = "C:/path/to/text/file.txt";
        String destinationFolder = "E:/AI/NeuralGPT/NeuralGPT";

        File file = new File(textFilePath);
        String fileExtension = FilenameUtils.getExtension(file.getName());

        String markdownFileName = FilenameUtils.removeExtension(file.getName()) +
            ".md";
        Path markdownFilePath = Paths.get(destinationFolder, markdownFileName);

        String text = Files.readString(file.toPath());
        String markdown = convertToMarkdown(text, fileExtension);

        Files.writeString(markdownFilePath, markdown);
    }

    private static String convertToMarkdown(String text, String fileExtension) {
        Parser parser = null;
        if (fileExtension.equals("txt")) {
```

```

parser = Parser.builder().build();
} else if (fileExtension.equals("docx")) {
parser = new DocxToMarkdownParser();
} else if (fileExtension.equals("pdf")) {
parser = new PdfToMarkdownParser();
}

Node document = parser.parse(text);
HtmlRenderer renderer = HtmlRenderer.builder().build();
return renderer.render(document);
}
}

```

Note: The above Java program requires the `commonmark` library, as well as custom parser classes for converting Docx and PDF files to Markdown.

Task:

3. Create a Bash script that reads a CSV file and generates a Markdown table from the data. The script should be able to handle different CSV formats and should save the Markdown table to a specified location in the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT).

Result:

Here's a Bash script that reads a CSV file and generates a Markdown table from the data. The script saves the Markdown table to a specified location in the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT).

```

#!/bin/bash

# Set the input CSV file path
input_file="example.csv"

# Set the output Markdown file path
output_file="E:/AI/NeuralGPT/NeuralGPT/table.md"

# Read the CSV file and generate a Markdown table
while read line
do
# Replace commas with pipes for Markdown table formatting
row=$(echo $line | sed 's/,/ | /g')

# Add Markdown table formatting to the row
if [ -z "$header" ]
then
# The first row is the header
header="$row"
separator=$(echo "$header" | sed 's/[^\|]*/-/g')
table="$header\n$separator"
else
# All other rows are data
table="$table\n$row"
fi
done < "$input_file"

# Save the Markdown table to the output file
echo -e "$table" > "$output_file"

```

Note that this script assumes that the input CSV file has a header row and uses commas as the delimiter. You may need to modify the script to handle different CSV formats.

Task:

Create a Python script that writes content produced by Cognosys to a text file and saves it to the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT) in the .txt format. The script should be fully automatic and autonomous, meaning it should not require any user input once executed.

Result:

Here's a Python script that writes content produced by Cognosys to a text file and saves it to the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT) in the .txt format:

```
import os

# Set the path to the directory where the text file will be saved
directory_path = r"E:\AI\NeuralGPT\NeuralGPT"

# Set the path to the file containing the content produced by Cognosys
content_file_path = r"path\to\content\file"

# Read the content from the file
with open(content_file_path, "r") as file:
    content = file.read()

# Set the name of the text file
file_name = "cognosys_content.txt"

# Set the path to the text file
file_path = os.path.join(directory_path, file_name)

# Write the content to the text file
with open(file_path, "w") as file:
    file.write(content)
```

This script reads the content produced by Cognosys from a file, sets the path to the directory where the text file will be saved, sets the name of the text file, and writes the content to the text file in the specified directory. The script is fully automatic and autonomous, meaning it does not require any user input once executed.

Task:

Create a Python script that writes content produced by Cognosys to an HTML file and saves it to the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT) in the .html format. The script should be fully automatic and autonomous, meaning it should not require any user input once executed.

Result:

Here's a Python script that can write content produced by Cognosys to an HTML file and save it to the local clone of NeuralGPT repository in the .html format:

```

import os

# Path to the local clone of NeuralGPT repository
neuralgpt_path = "E:/AI/NeuralGPT/NeuralGPT"

# Content produced by Cognosys
content = "This is some content produced by Cognosys."

# Create the HTML file
filename = "content.html"
filepath = os.path.join(neuralgpt_path, filename)
with open(filepath, "w") as f:
    f.write("<html>\n")
    f.write("<head>\n")
    f.write("<title>Content from Cognosys</title>\n")
    f.write("</head>\n")
    f.write("<body>\n")
    f.write(f"<p>{content}</p>\n")
    f.write("</body>\n")
    f.write("</html>\n")

print(f"File saved to {filepath}")

```

Once executed, this script will create an HTML file named "content.html" in the NeuralGPT repository directory with the content produced by Cognosys.

Task:

Create a Bash script that reads a JSON file and generates a Markdown file from the data. The script should be able to handle different JSON formats and should save the Markdown file to a specified location in the local clone of NeuralGPT repository (E:\AI\NeuralGPT\NeuralGPT).

Result:

Here's a Bash script that reads a JSON file and generates a Markdown file from the data:

```

#!/bin/bash

# Set the path to the JSON file
json_file="path/to/json/file.json"

# Set the path to the output Markdown file
markdown_file="E:/AI/NeuralGPT/NeuralGPT/output.md"

# Parse the JSON file and extract the data
data=$(jq -r '.data' $json_file)

# Convert the data to Markdown format
markdown=$(echo $data | pandoc -f html -t markdown)

# Write the Markdown to the output file
echo $markdown > $markdown_file

```

This script assumes that you have the `jq` and `pandoc` utilities installed on your system. It reads the data from the specified JSON file, converts it to Markdown format using `pandoc`, and writes the output to the specified Markdown file. You can modify the paths as needed to suit your environment.