**Assignment 4**

class DateInfoInterface:

    """

    Manages the input and output for date availability requests.

    """

    def \_\_init\_\_(self, resource\_url):

        # Initialize with the resource URL

        self.resource\_url = resource\_url

        self.available\_dates = []

    def get\_available\_dates(self):

        # Return the fetched dates

        if not self.available\_dates:

            self.fetch\_available\_dates()

        return self.available\_dates

import requests

from bs4 import BeautifulSoup

class BrowserInterface:

    """

    Handles interactions between the bot and the browser for scraping web data.

    """

    def \_\_init\_\_(self, url):

        # Store the URL to scrape data from

        self.url = url

        self.page\_content = None

    def fetch\_page(self):

        # Fetch the content of the webpage

        try:

            response = requests.get(self.url)

            response.raise\_for\_status()  # Check for request errors

            self.page\_content = response.text

        except requests.exceptions.RequestException as e:

            print(f"Failed to fetch the page: {e}")

            self.page\_content = None

    def parse\_page(self):

        # Parse the webpage content using BeautifulSoup

        if self.page\_content:

            soup = BeautifulSoup(self.page\_content, 'html.parser')

            return soup

        else:

            print("No content to parse")

            return None

import pandas as pd

class ExcelInterface:

    """

    Handles data extraction to Excel.

    """

    def \_\_init\_\_(self, file\_path):

        # Initialize with the file path where the Excel file will be saved

        self.file\_path = file\_path

        self.data = None

    def save\_data\_to\_excel(self, data):

        # Save the data to an Excel file

        try:

            df = pd.DataFrame(data)

            df.to\_excel(self.file\_path, index=False)

            print(f"Data saved to {self.file\_path}")

        except Exception as e:

            print(f"Failed to save data to Excel: {e}")

    def load\_data\_from\_excel(self):

        # Load data from an Excel file

        try:

            self.data = pd.read\_excel(self.file\_path)

            print(f"Data loaded from {self.file\_path}")

        except Exception as e:

            print(f"Failed to load data from Excel: {e}")

            self.data = None

    def get\_data(self):

        # Return the loaded data

        if self.data is None:

            self.load\_data\_from\_excel()

        return self.data

class ProductInfoInterface:

    """

    Manages the input and output for product information requests.

    """

    def \_\_init\_\_(self, product\_url):

        # Initialize with the product URL

        self.product\_url = product\_url

        self.product\_details = {}

    def fetch\_product\_details(self):

        # Pretend to fetch product details from the URL (placeholder)

        # In a real scenario, you'd scrape the page and extract details

        self.product\_details = {

            'name': 'Sample Product',

            'price': '123.45',

            'availability': 'In Stock'

        }

    def get\_product\_details(self):

        # Return the fetched product details

        if not self.product\_details:

            self.fetch\_product\_details()

        return self.product\_details

class DiscordInterface:

    """

    Manages the interactions between the bot and the user on Discord.

    """

    def \_\_init\_\_(self, message):

        # Store the message received from the user

        self.message = message

        # Store the message as the command directly

        self.command = message

        self.response = None

    def generate\_response(self):

        # Generate a response based on the command

        # Mostly examples to give ideas, will be dynamic and changed soon

        if self.command.lower() == 'hello':

            self.response = 'Hey there!'

        elif self.command.lower() == 'help':

            self.response = 'These are the commands you can use: hello, help, latest price, last checked, share url...'

        elif self.command.lower() == 'latest price':

            # Placeholder for fetching the latest price from the system

            self.response = 'The latest price for the tracked product is $123.45.'

        elif self.command.lower() == 'last checked':

            # Placeholder for fetching the last checked time

            self.response = 'The last time we checked the price was at 10:30 AM, August 10, 2024.'

        elif self.command.lower().startswith('share url'):

            # Assuming the URL is shared in the format "share url <url>"

            url = self.command[len('share url'):].strip()

            # Placeholder response for URL processing

            self.response = f'Thank you for sharing the URL: {url}. We are fetching the details...'

        else:

            self.response = 'Sorry, I didn’t understand that command.'

        return self.response

Oguz Kaan Yildirim