AI Assisted Coding

Assignment - 5

Ch. Nithin Reddy || 2303A51550 || Batch:- 8

Task Description #1 (Privacy in API Usage)

Task: Use an AI tool to generate a Python program that connects to a weather API.

Code:

```python
#Generate code to fetch weather data securely without exposing API keys in the code
import os
import requests
def get_weather_data(city):
    api_key = os.getenv('WEATHER_API_KEY')  # Fetch API key from environment variable
    if not api_key:
        raise ValueError("API key not found. Please set the WEATHER_API_KEY environment variable.")
    base_url = "http://api.openweathermap.org/data/2.5/weather"
    params = {
        'q': city,
        'appid': api_key,
        'units': 'metric'
    }
    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        return {"error": "City not found or API request failed."}
```

Task Description #2 (Privacy & Security in File Handling)

Task: Use an AI tool to generate a Python script that stores user data (name, email, password) in a file.

Analyze: Check if the AI stores sensitive data in plain text or without encryption.

Code:

```python

#Generate a Python script that collects user details (name, email, and password) and stores them in a file.
#The script should write all details directly to a text file.
def store_user_details(name, email, password, filename='user_details.txt'):
    with open(filename
, 'a') as file:
        file.write(f"Name: {name}\n")
        file.write(f"Email: {email}\n")
        file.write(f"Password: {password}\n")
        file.write("-" * 20 + "\n")
if __name__ == "__main__":
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    password = input("Enter your password: ")
    store_user_details(name, email, password)
    print("User details stored successfully.")

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant> & "C:/Users/NITHIN REDDY/AppData/Local/Programs/Python
Desktop/All courses/AI assistant/lab5.py"
Enter your name: Nithin
Enter your email: 2303A51550@sru.edu.in
Enter your password: Student@sru
User details stored successfully.
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant>
```

Task Description #3 (Transparency in Algorithm Design)

Objective: Use AI to generate an Armstrong number checking function

with comments and explanations.

Code:

```python
#Explain the Armstrong number checking code line by line in simple terms.
#Describe what each variable and statement does and how the algorithm works.
def is_armstrong(number):
    """
    Check if a number is an Armstrong number.

    An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.

    Parameters:
    number (int): The number to check.

    Returns:
    bool: True if the number is an Armstrong number, False otherwise.
    """
    num_str = str(number)  # Convert the number to a string to easily access each digit
    num_digits = len(num_str)  # Get the number of digits in the number
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)  # Calculate the sum of each digit raised to the power of num_digits
    return sum_of_powers == number  # Check if the calculated sum is equal to the original number
if __name__ == "__main__":
    num = int(input("Enter a number to check if it's an Armstrong number: "))  # Take user input
    if is_armstrong(num):  # Call the function to check if the number is an Armstrong number
        print(f"{num} is an Armstrong number.")  # Print result if true
    else:
        print(f"{num} is not an Armstrong number.")  # Print result if false

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE                                    Python
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant> & "C:/Users/NITHIN REDDY/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/NIT
Desktop/All courses/AI assistant/lab5.py"
Enter a number to check if it's an Armstrong number: 153
153 is an Armstrong number.
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant>
```

Task Description #4 (Transparency in Algorithm Comparison)

Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

Code:

```python
# Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they d
def bubble_sort(arr):
    """
    Sort an array using the Bubble Sort algorithm.

    Parameters:
    arr (list): The list of elements to be sorted.

    Returns:
    list: The sorted list.
    """
    n = len(arr)  # Get the length of the array
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already sorted, no need to check them
        for j in range(0, n-i-1):
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr  # Return the sorted array
def quick_sort(arr):
    """
    Sort an array using the QuickSort algorithm.

    Parameters:
    arr (list): The list of elements to be sorted.

    Returns:
    list: The sorted list.
    """
    if len(arr) <= 1:  # Base case: if the array has 0 or 1 element, it's already sorted
        return arr
    else:
        pivot = arr[len(arr) // 2]  # Choose the middle element as the pivot
        left = [x for x in arr if x < pivot]  # Elements less than the pivot
        middle = [x for x in arr if x == pivot]  # Elements equal to the pivot
        right = [x for x in arr if x > pivot]  # Elements greater than the pivot
        # Recursively apply quick_sort to left and right, and combine results
        return quick_sort(left) + middle + quick_sort(right)
if __name__ == "__main__":
    sample_array = [64, 34, 25, 12, 22, 11, 90]
    print("Original array:", sample_array)

    # Bubble Sort
    sorted_array_bubble = bubble_sort(sample_array.copy())  # Use copy to avoid in-place sorting affecting the original
    print("Sorted array using Bubble Sort:", sorted_array_bubble)

    # QuickSort
    sorted_array_quick = quick_sort(sample_array)  # QuickSort returns a new sorted array
    print("Sorted array using QuickSort:", sorted_array_quick)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE
153 is an Armstrong number.
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant> ^C
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant> & "C:/Users/NITHIN REDDY/App
Y/OneDrive/Desktop/All courses/AI assistant/lab5.py"
Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array using Bubble Sort: [11, 12, 22, 25, 34, 64, 90]
Sorted array using QuickSort: [11, 12, 22, 25, 34, 64, 90]
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant>
```

Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.

Code:

```python
# Generate a recommendation system that also provides reasons for each suggestion.
def recommend_movies(user_preferences, movie_database):
    """
    Recommend movies based on user preferences.

    Parameters:
    user_preferences (dict): A dictionary containing user preferences such as genre, director, and acto
    movie_database (list): A list of dictionaries, each representing a movie with its attributes.

    Returns:
    list: A list of recommended movies with reasons for each suggestion.
    """
    recommendations = []  # Initialize an empty list to store recommendations

    for movie in movie_database:
        score = 0  # Initialize score for each movie
        reasons = []  # Initialize reasons for recommendation

        # Check genre preference
        if movie['genre'] in user_preferences.get('genres', []):
            score += 2  # Increase score for matching genre
            reasons.append(f"Matches preferred genre: {movie['genre']}")

        # Check director preference
        if movie['director'] in user_preferences.get('directors', []):
            score += 1  # Increase score for matching director
            reasons.append(f"Directed by preferred director: {movie['director']}")

        # Check actor preference
        for actor in movie['actors']:
            if actor in user_preferences.get('actors', []):
                score += 1  # Increase score for each matching actor
                reasons.append(f"Features preferred actor: {actor}")

        # If the movie has a positive score, add it to recommendations
        if score > 0:
            recommendations.append({
                'movie': movie['title'],
                'score': score,
                'reasons': reasons
            })
```

```python
    # Sort recommendations by score in descending order
    recommendations.sort(key=lambda x: x['score'], reverse=True)

    return recommendations  # Return the list of recommended movies with reasons
if __name__ == "__main__":
    user_preferences = {
        'genres': ['Action', 'Sci-Fi'],
        'directors': ['Christopher Nolan'],
        'actors': ['Leonardo DiCaprio', 'Scarlett Johansson']
    }
    movie_database = [
        {
            'title': 'Inception',
            'genre': 'Sci-Fi',
            'director': 'Christopher Nolan',
            'actors': ['Leonardo DiCaprio', 'Joseph Gordon-Levitt']
        },
        {
            'title': 'The Avengers',
            'genre': 'Action',
            'director': 'Joss Whedon',
            'actors': ['Robert Downey Jr.', 'Scarlett Johansson']
        },
        {
            'title': 'La La Land',
            'genre': 'Romance',
            'director': 'Damien Chazelle',
            'actors': ['Ryan Gosling', 'Emma Stone']
        }
    ]
    recommendations = recommend_movies(user_preferences, movie_database)
    for rec in recommendations:
        print(f"Movie: {rec['movie']}")
        print(f"Score: {rec['score']}")
        print("Reasons:")
        for reason in rec['reasons']:
            print(f"- {reason}")
        print()
```

```
PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant> & "C:/Users/NITHIN REDDY/A
ant/lab5.py"
Movie: Inception
Score: 4
Reasons:
- Matches preferred genre: Sci-Fi
- Directed by preferred director: Christopher Nolan
- Features preferred actor: Leonardo DiCaprio

Movie: The Avengers
Score: 3
Reasons:
- Matches preferred genre: Action
- Features preferred actor: Scarlett Johansson

PS C:\Users\NITHIN REDDY\OneDrive\Desktop\All courses\AI assistant>
```