SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s) Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week3 – Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
Assignment Number:5.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	Lab 5: Ethical Foundations – Responsible AI Coding Practices **Lab Objectives:** • To explore the ethical risks associated with AI-generated code.	Week3 - Monday

- To recognize issues related to security, bias, transparency, and copyright.
- To reflect on the responsibilities of developers when using AI tools in software development.
- To promote awareness of best practices for responsible and ethical AI coding.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Identify and avoid insecure coding patterns generated by AI tools.
- Detect and analyze potential bias or discriminatory logic in AI-generated outputs.
- Evaluate originality and licensing concerns in reused AI-generated code.
- Understand the importance of explainability and transparency in AI-assisted programming.
- Reflect on accountability and the human role in ethical AI coding practices..

Task Description #1 (Privacy in API Usage)
Task: Use an AI tool to generate a Python program that connects to a weather API.
Prompt:
"Generate code to fetch weather data securely without exposing API keys in the code."
Expected Output:

- Original AI code (check if keys are hardcoded).
- Secure version using environment variables.
- Prompt: generate the py code connects to a weather API key in the code.

Prompt; create the py code generate data securely without exposing API key in the code.

- Code:

```python
import os
import requests
from dotenv import load_dotenv

def get_weather_insecure(city):
    print("\n🔓 Insecure Version (Hardcoded API Key)")
    api_key = "YOUR_API_KEY"  # Replace with your real API key (for demo only)
    url = f"http://api.weatherapi.com/v1/current.json?key={api_key}&q={city}"

    try:
        response = requests.get(url)
        data = response.json()
        print(f"{city} Temperature: {data['current']['temp_c']} °C")
    except:
        print("Error fetching data. Check your key or city name.")

def get_weather_secure(city):
    print("\n🔐 Secure Version (API Key from .env)")
    load_dotenv()  # Load from .env file
    api_key = os.getenv("WEATHER_API_KEY")
    if not api_key:
        print("API key not found. Please set WEATHER_API_KEY in .env file.")
        return

    url = f"http://api.weatherapi.com/v1/current.json?key={api_key}&q={city}"

    try:
        response = requests.get(url)
```

-

```python

    try:
        response = requests.get(url)
        data = response.json()
        print(f"{city} Temperature: {data['current']['temp_c']} °C")
    except:
        print("Error fetching data. Check your key or city name.")

# --- User Choice ---
print("Choose method to fetch weather:")
print("1. Insecure (Hardcoded API key)")
print("2. Secure (Using environment variable from .env)")

choice = input("Enter 1 or 2: ").strip()
city = input("Enter city name: ").strip()

if choice == "1":
    get_weather_insecure(city)
elif choice == "2":
    get_weather_secure(city)
else:
    print("Invalid choice.")
```

-

Output:

```
PS C:\AI CODEING> & 'c:\Users\kbhuv\AppData\Local\Programs\Py
  '60430' '--' 'c:\AI CODEING\lab ass 1.1.py'
Choose method to fetch weather:
1. Insecure (Hardcoded API key)
2. Secure (Using environment variable from .env)
Enter 1 or 2: 1
Enter city name: Ongole

🔓 Insecure Version (Hardcoded API Key)
Error fetching data. Check your key or city name.
```

Comment:

This code fetches a city's weather in two ways:

1. **Insecure** – uses a hardcoded API key.
2. **Secure** – reads the API key from an environment file.

The user chooses a method, enters a city, and the program shows its temperature or an error if something fails.

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.

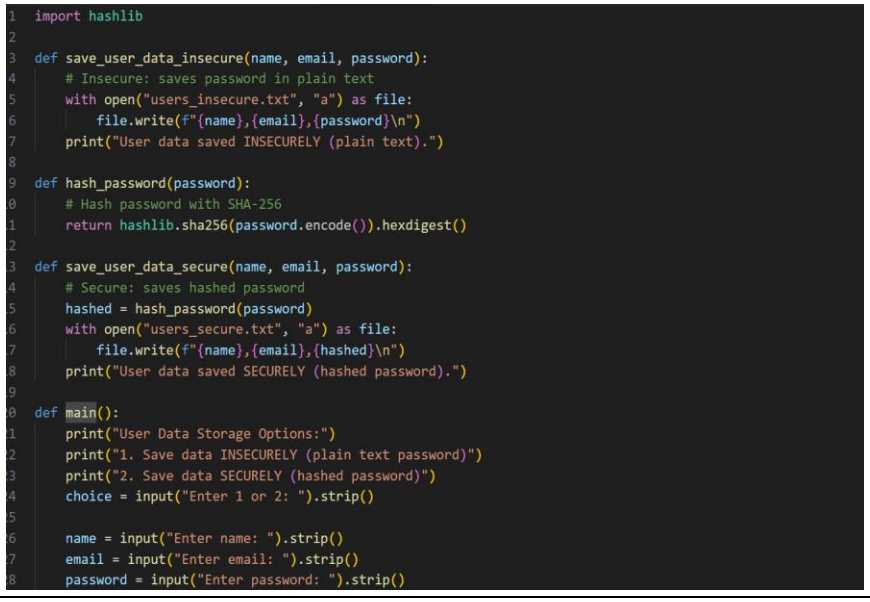
Expected Output:

- Identified privacy risks.
- Revised version with encrypted password storage (e.g., hashing).

Prompt:

Generate the py code check ai store sencetive data in plain text or without encrypyion.

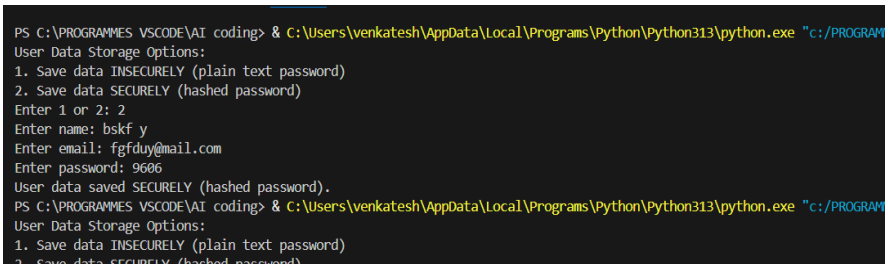
Code:



```python
import hashlib

def save_user_data_insecure(name, email, password):
    # Insecure: saves password in plain text
    with open("users_insecure.txt", "a") as file:
        file.write(f"{name},{email},{password}\n")
    print("User data saved INSECURELY (plain text).")

def hash_password(password):
    # Hash password with SHA-256
    return hashlib.sha256(password.encode()).hexdigest()

def save_user_data_secure(name, email, password):
    # Secure: saves hashed password
    hashed = hash_password(password)
    with open("users_secure.txt", "a") as file:
        file.write(f"{name},{email},{hashed}\n")
    print("User data saved SECURELY (hashed password).")

def main():
    print("User Data Storage Options:")
    print("1. Save data INSECURELY (plain text password)")
    print("2. Save data SECURELY (hashed password)")
    choice = input("Enter 1 or 2: ").strip()

    name = input("Enter name: ").strip()
    email = input("Enter email: ").strip()
    password = input("Enter password: ").strip()
```

```
29
30      if choice == "1":
31          save_user_data_insecure(name, email, password)
32      elif choice == "2":
33          save_user_data_secure(name, email, password)
34      else:
35          print("Invalid choice.")
36
37  if __name__ == "__main__":
38      main()
```

Output:



Comment;

This program demonstrates secure vs insecure user data storage.

Insecure: saves password in plain text.

Secure: saves password hashed with SHA-256 before saving.

User chooses storage method, then enters name, email, and password.

Task Description #3 (Transparency in Algorithm Design)

Objective: Use AI to generate an Armstrong number checking function with comments and explanations.

Instructions:

1. Ask AI to explain the code line-by-line.
2. Compare the explanation with code functionality.

Expected Output:

- Transparent, commented code.
- Correct, easy-to-understand explanation.

Prompt:
create python code of Armstrong number checking function with comments and explanations.

Code:

```python
def is_armstrong(n):
    # Convert number to string for digit access
    s = str(n)
    # Count digits (power)
    p = len(s)
    # Compute sum of each digit raised to power p
    total = sum(int(digit) ** p for digit in s)
    # Compare sum with original number
    return total == n

# --- Example usage ---
num = int(input("Enter a number: "))

if is_armstrong(num):
    print(f"{num} is an Armstrong number ✅")
else:
    print(f"{num} is NOT an Armstrong number ❌")
```

Output:

```
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python31
Enter a number: 5
5 is an Armstrong number ✅
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python31
Enter a number: 10
10 is NOT an Armstrong number ❌
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python31
Enter a number: 10
10 is NOT an Armstrong number
PS C:\PROGRAMMES VSCODE\AI coding>
```

Comment:

This code takes a number from the user and checks if it is an **Armstrong number**.
It does this by splitting the number into digits, raising each digit to the power of the total digits, adding them, and comparing the sum with the original number.
If they match, it prints that the number is an Armstrong number, otherwise it says it is not.

Task Description #4 (Transparency in Algorithm Comparison)
Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).
Prompt:
"Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ."
Expected Output:

- Code for both algorithms.
- Transparent, comparative explanation of their logic and efficiency.

Prompt:
Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ."

Code:

```python
# BubbleSort: simple but slow
def bubble_sort(arr):
    for i in range(len(arr)):
        for j in range(len(arr) - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# QuickSort: faster, uses pivot
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr)//2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

# Example
arr = [64, 34, 25, 12, 22, 11, 90]
print("BubbleSort:", bubble_sort(arr.copy()))
print("QuickSort:", quick_sort(arr.copy()))

```

Output:

```
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python313\
BubbleSort: [11, 12, 22, 25, 34, 64, 90]
QuickSort: [11, 12, 22, 25, 34, 64, 90]
PS C:\PROGRAMMES VSCODE\AI coding>
```

Comment:

BubbleSort is easy but slow ($O(n^2)$), QuickSort is faster ($O(n \log n)$) using pivot and divide & conquer.

Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.

Prompt:

"Generate a recommendation system that also provides reasons for each suggestion."

Expected Output:

- Code with explainable recommendations.
- Evaluation of whether explanations are understandable.

Prompt:

Generate a recommendation system that also provides reasons for each suggestion

Code:

```python
# Sample product data
products = {
    "Laptop": {"category": "Electronics", "price": 60000, "rating": 4.5},
    "Smartphone": {"category": "Electronics", "price": 30000, "rating": 4.3},
    "Headphones": {"category": "Accessories", "price": 2000, "rating": 4.0},
    "Book": {"category": "Education", "price": 500, "rating": 4.8},
    "Shoes": {"category": "Fashion", "price": 2500, "rating": 4.1}
}

def recommend(budget, category_preference):
    recommendations = []
    for product, details in products.items():
        if details["price"] <= budget and details["category"] == category_preference:
            reason = f"{product} is recommended because it fits your budget (₹{budget}) " \
                     f"and belongs to your preferred category '{category_preference}' " \
                     f"with a good rating of {details['rating']}⭐."
            recommendations.append(reason)
    return recommendations

# Example usage
user_budget = 30000
user_category = "Electronics"

print("Recommendations for You:\n")
for rec in recommend(user_budget, user_category):
    print("-", rec)
```

Output:

```
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python313\python.exe "c:/PROGRAMMES VSCODE/AI coding/assignment 5.1.p
Recommendations for You:

- Smartphone is recommended because it fits your budget (₹30000) and belongs to your preferred category 'Electronics' with a good rating of 4.3⭐.
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python313\python.exe "c:/PROGRAMMES VSCODE/AI coding/assignment 5.1.p
Recommendations for You:

- Smartphone is recommended because it fits your budget (₹30000) and belongs to your preferred category 'Electronics' with a good rating of 4.3⭐.
PS C:\PROGRAMMES VSCODE\AI coding>
```

Comment:

This code is a **simple product recommendation system**.
It stores some sample products with their category, price, and rating.
When the user gives a budget and a preferred category, the program checks which products match both conditions.

Task Description #6 (Transparent Code Generation)
Task: Ask AI to generate a Python function for calculating factorial using recursion.
Prompt:
"Generate a recursive factorial function with comments that explain each line and a final summary of the algorithm's flow."
Expected Output:

- Fully commented code.
- Clear documentation of how recursion works.

Prompt:
Generate a recursive factorial function with comments that explain each line and a final summary of the algorithm's flow."

Code:

```
def factorial(n):
    # Base case: if n is 0 or 1, factorial is always 1
    if n == 0 or n == 1:
        return 1
    # Recursive case: multiply n by factorial of (n-1)
    else:
        return n * factorial(n - 1)

# Example usage
num = 5
print("Factorial of", num, "is:", factorial(num))

"""
Summary:
1. The function keeps calling itself with smaller values (n-1).
2. This continues until it reaches the base case (n = 0 or 1).
3. Then the results are multiplied backward to give the final factorial value.
"""
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python313\python.exe "c:/PROGR
Factorial of 5 is: 120
PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python313\python.exe "c:/PROGR
Factorial of 5 is: 120
PS C:\PROGRAMMES VSCODE\AI coding>
```

Comment:

This code defines a **recursive factorial function**.

- If the number is 0 or 1, it returns 1 (base case).
- Otherwise, it multiplies the number by the factorial of the previous number (recursive step).
- The process repeats until it reaches the base case.
- Finally, the result builds up and prints the factorial of the given number.

Task Description #7 (Inclusiveness in Customer Support)

Code Snippet:

```
def support_reply(name, gender):
    if gender.lower() == "male":
        prefix = "Mr."
    else:
        prefix = "Mrs."
    return f"Dear {prefix} {name}, we have resolved your i
```

Task:

Regenerate the code so that support messages use neutral language (e.g., "Dear {name}") and optionally accept preferred titles.

Expected Output:

- Neutral, user-friendly support responses.

Prompt:

create the code in python Regenerate the code so that support messages use neutral language.

Code:

```python
def support_message(name, issue, title=None):
    # If title is provided, use it, else just use "Dear {name}"
    greeting = f"Dear {title} {name}," if title else f"Dear {name},"

    # Neutral support response
    message = (
        f"{greeting}\n\n"
        f"Thank you for reaching out to us. We understand your concern regarding: {issue}.\n"
        f"Our team is here to assist you and will work on resolving this as quickly as possible.\n\n"
        f"Best regards,\nCustomer Support Team"
    )
    return message

# Example usage
print(support_message("Venkatesh", "billing error"))
print()
print(support_message("Riya", "login issue", title="Ms."))
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\PROGRAMMES VSCODE\AI coding> & C:\Users\venkatesh\AppData\Local\Programs\Python\Python313\python.exe "c:/PROGRAMMES VSCODE/AI coding/a
Dear Venkatesh,

Thank you for reaching out to us. We understand your concern regarding: billing error.
Our team is here to assist you and will work on resolving this as quickly as possible.

Best regards,
Customer Support Team

Dear Ms. Riya,

Thank you for reaching out to us. We understand your concern regarding: login issue.
Our team is here to assist you and will work on resolving this as quickly as possible.

Best regards,
Customer Support Team
PS C:\PROGRAMMES VSCODE\AI coding>
```

Comment:

This code creates polite and inclusive customer support messages. It allows addressing the user either neutrally as *"Dear {name}"* or with an optional title like *"Mr."* or *"Ms."*. The function clearly states the issue and assures the customer that support will help resolve it. Overall, it ensures respectful, user-friendly communication.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Transparency	1
Inclusiveness	0.5
Data security and Privacy	1
Total	2.5 Marks