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. 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}"
```

```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.
```

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).

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()

    if choice == "1":
        save_user_data_insecure(name, email, password)
    elif choice == "2":
        save_user_data_secure(name, email, password)
    else:
        print("Invalid choice.")

if __name__ == "__main__":
    main()
```

OUTPUT :-

```
PS C:\AI CODEING> & C:\Users\kbhuv\AppData\Local\Programs\Python\Python313\python.exe "c:
User Data Storage Options:
1. Save data INSECURELY (plain text password)
2. Save data SECURELY (hashed password)
Enter 1 or 2: 2
Enter name: BHUVI
Enter email: Bhuvaneshwarreddy2006@gmail.com
Enter password: Reddy@2006
User data saved SECURELY (hashed 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.

CODE :-

```python
def is_armstrong(number):
    # Convert number to string to easily get the number of digits
    num_str = str(number)
    num_digits = len(num_str)  # Count of digits in the number

    # Initialize a variable to hold the sum of digits raised to the power of num_digits
    total = 0

    # Iterate through each digit in the number
    for digit in num_str:
        total += int(digit) ** num_digits  # Raise each digit to the power and add to total

    # Check if the sum equals the original number
    return total == number
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:\AI CODEING>  c:; cd 'c:\AI CODEING'; & 'c:\Use
d\libs\debugpy\launcher' '50195' '--' 'c:\AI CODEING
Enter a number: 2
2 is an Armstrong number.
```

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.

CODE :-

```python
def bubble_sort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n - i - 1):
            # Swap if the element found is greater than the next
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
def quick_sort(arr):
    # Base case: a list of 0 or 1 element is already sorted
    if len(arr) <= 1:
        return arr

    # Choose the pivot (here: last element)
    pivot = arr[-1]
    left = []   # Elements less than pivot
    right = []  # Elements greater than or equal to pivot

    # Partition the array
    for i in arr[:-1]:
        if i < pivot:
            left.append(i)
        else:
            right.append(i)
```

```python
    # Recursively sort the partitions and combine
    return quick_sort(left) + [pivot] + quick_sort(right)
arr1 = [64, 34, 25, 12, 22, 11, 90]
arr2 = arr1.copy()

print("Original:", arr1)
print("BubbleSort:", bubble_sort(arr1.copy()))
print("QuickSort:", quick_sort(arr2.copy()))
```

OUTPUT :-

```
'50142' '--' 'c:\AI CODEING\lab ass 1.1.py'
Original: [64, 34, 25, 12, 22, 11, 90]
BubbleSort: [11, 12, 22, 25, 34, 64, 90]
QuickSort: [11, 12, 22, 25, 34, 64, 90]
```

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.

CODE :-

```python
# Sample product catalog
products = [
    {"name": "Wireless Headphones", "category": "Electronics", "price": 1500, "features": ["wireless", "bass", "portable"]},
    {"name": "Bluetooth Speaker", "category": "Electronics", "price": 1200, "features": ["wireless", "portable", "bass"]},
    {"name": "Yoga Mat", "category": "Fitness", "price": 500, "features": ["lightweight", "non-slip"]},
    {"name": "Smartwatch", "category": "Electronics", "price": 3500, "features": ["fitness", "notifications", "portable"]},
    {"name": "Running Shoes", "category": "Fitness", "price": 2500, "features": ["comfortable", "lightweight", "durable"]},
]

# User preferences
user_preferences = {
    "category": "Electronics",
    "features": ["wireless", "portable"],
    "budget": 2000
}

# Recommendation engine with explanation
def recommend_products(user_prefs, product_list):
    recommendations = []

    for product in product_list:
        reasons = []
        score = 0

        # Match category
        if product["category"] == user_prefs["category"]:
            score += 1
            reasons.append("Matches your interest in Electronics")

        # Match features
        matched_features = set(user_prefs["features"]).intersection(product["features"])
        if matched_features:
            score += len(matched_features)
            reasons.append(f"Features include: {', '.join(matched_features)}")

        # Check price within budget
        if product["price"] <= user_prefs["budget"]:
            score += 1
            reasons.append(f"Price ₹{product['price']} is within your budget")

        # Recommend if score is reasonable
        if score >= 2:
            recommendations.append({
                "product": product["name"],
                "price": product["price"],
                "reasons": reasons
            })

    return recommendations

# Run recommendation
results = recommend_products(user_preferences, products)

# Display results
print("\nRecommended Products:\n")
for r in results:
    print(f"👉 {r['product']} (₹{r['price']})")
    print("   Why?")
    for reason in r["reasons"]:
        print(f"   - {reason}")
    print()
```

OUTPUT :-

```
PS C:\AI CODEING>  & 'c:\Users\kbhuv\AppData\Local\Programs\Python\Python313
 '54291' '--' 'c:\AI CODEING\lab ass 1.1.py'

Recommended Products:

👉 Wireless Headphones (₹1500)
   Why?
   - Matches your interest in Electronics
   - Features include: portable, wireless
   - Price ₹1500 is within your budget

👉 Bluetooth Speaker (₹1200)
   Why?
   - Matches your interest in Electronics
   - Features include: portable, wireless
   - Price ₹1200 is within your budget

👉 Smartwatch (₹3500)
   Why?
   - Matches your interest in Electronics
   - Features include: portable
```

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.

CODE :-

```python
def factorial(n):
    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1
    # Recursive case: n * factorial of (n-1)
    return n * factorial(n - 1)

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

OUTPUT :-

```
PS C:\AI CODEING>  & 'c:\Users\kbhuv\AppData\Loca
 '60516' '--' 'c:\AI CODEING\lab ass 1.1.py'
Factorial of 5 is: 120
```

Task Description #7 (Inclusiveness in Customer Support)

Code Snippet:

```python
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.

CODE :-

```python
def generate_support_message(name, title=None):
    # Use the title only if provided
    salutation = f"Dear {title} {name}" if title else f"Dear {name}"

    message = f"""{salutation},

Thank you for reaching out to our support team. We appreciate your message and will respond as soon as possible.

If you have any further questions or updates, feel free to reply to this email.

Best regards,
Customer Support Team"""

    return message

# Example usage
print(generate_support_message("Alex"))              # Without title
print()
print(generate_support_message("Jordan", "Dr."))     # With title
```

OUTPUT :-

```
PS C:\AI CODEING> c:; cd 'c:\AI CODEING'; & 'C:\Users\kbhuv\AppData\Local\Programs\Python\Python313\python.exe' 'c:\
d\libs\debugpy\launcher' '55227' '--' 'c:\AI CODEING\lab ass 1.1.py'
Dear Alex,

Thank you for reaching out to our support team. We appreciate your message and will respond as soon as possible.

If you have any further questions or updates, feel free to reply to this email.

Best regards,
Customer Support Team

Dear Dr. Jordan,

Thank you for reaching out to our support team. We appreciate your message and will respond as soon as possible.

If you have any further questions or updates, feel free to reply to this email.

Best regards,
Customer Support Team
```

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		