SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

ProgramName:B. Tech	**Assignment Type: Lab**	**AcademicYear:**2025-2026

CourseCoordinatorName	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)

CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week3 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	

AssignmentNumber:**5.2**(Present assignment number)/**24**(Total number of assignments)

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

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 and Data Security)

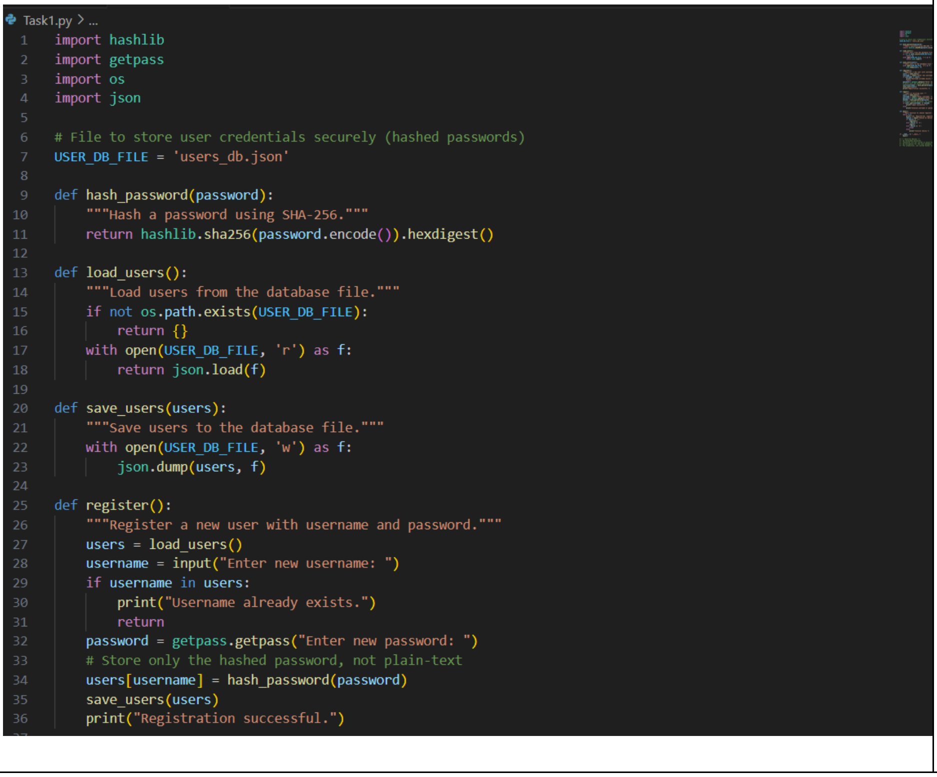
- Use an AI tool (e.g., Copilot, Gemini, Cursor) to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

Expected Output#1

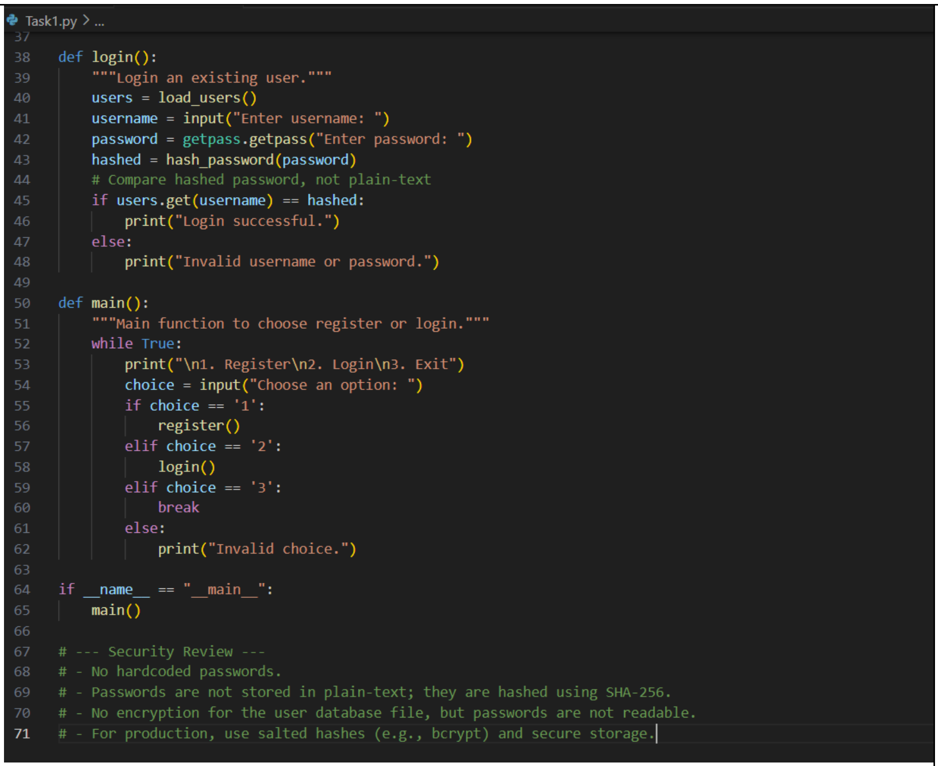
- Identification of insecure logic; revised secure version with proper password hashing/encrypting and environment variable use.

VS CODE:

PROMT: Generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

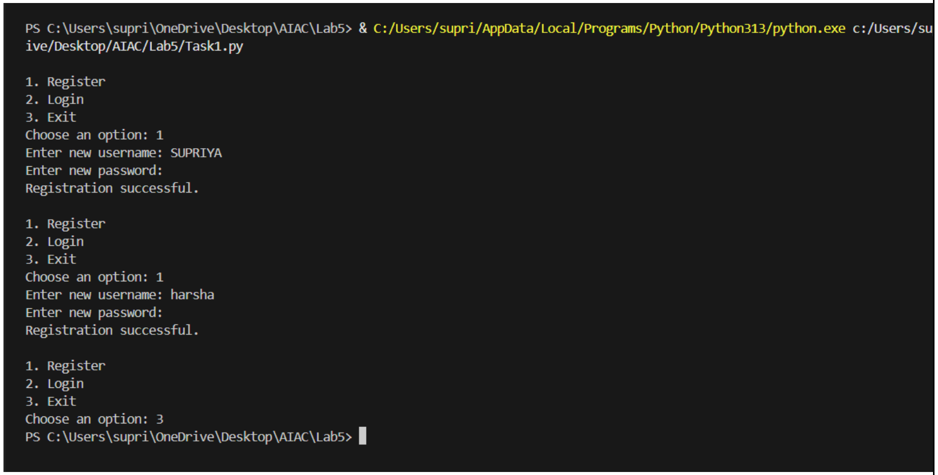


```python
import hashlib
import getpass
import os
import json

# File to store user credentials securely (hashed passwords)
USER_DB_FILE = 'users_db.json'

def hash_password(password):
    """Hash a password using SHA-256."""
    return hashlib.sha256(password.encode()).hexdigest()

def load_users():
    """Load users from the database file."""
    if not os.path.exists(USER_DB_FILE):
        return {}
    with open(USER_DB_FILE, 'r') as f:
        return json.load(f)

def save_users(users):
    """Save users to the database file."""
    with open(USER_DB_FILE, 'w') as f:
        json.dump(users, f)

def register():
    """Register a new user with username and password."""
    users = load_users()
    username = input("Enter new username: ")
    if username in users:
        print("Username already exists.")
        return
    password = getpass.getpass("Enter new password: ")
    # Store only the hashed password, not plain-text
    users[username] = hash_password(password)
    save_users(users)
    print("Registration successful.")
```



```python

def login():
    """Login an existing user."""
    users = load_users()
    username = input("Enter username: ")
    password = getpass.getpass("Enter password: ")
    hashed = hash_password(password)
    # Compare hashed password, not plain-text
    if users.get(username) == hashed:
        print("Login successful.")
    else:
        print("Invalid username or password.")

def main():
    """Main function to choose register or login."""
    while True:
        print("\n1. Register\n2. Login\n3. Exit")
        choice = input("Choose an option: ")
        if choice == '1':
            register()
        elif choice == '2':
            login()
        elif choice == '3':
            break
        else:
            print("Invalid choice.")

if __name__ == "__main__":
    main()

# --- Security Review ---
# - No hardcoded passwords.
# - Passwords are not stored in plain-text; they are hashed using SHA-256.
# - No encryption for the user database file, but passwords are not readable.
# - For production, use salted hashes (e.g., bcrypt) and secure storage.
```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Programs/Python/Python313/python.exe c:/Users/su
ive/Desktop/AIAC/Lab5/Task1.py

1. Register
2. Login
3. Exit
Choose an option: 1
Enter new username: SUPRIYA
Enter new password:
Registration successful.

1. Register
2. Login
3. Exit
Choose an option: 1
Enter new username: harsha
Enter new password:
Registration successful.

1. Register
2. Login
3. Exit
Choose an option: 3
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> 
```

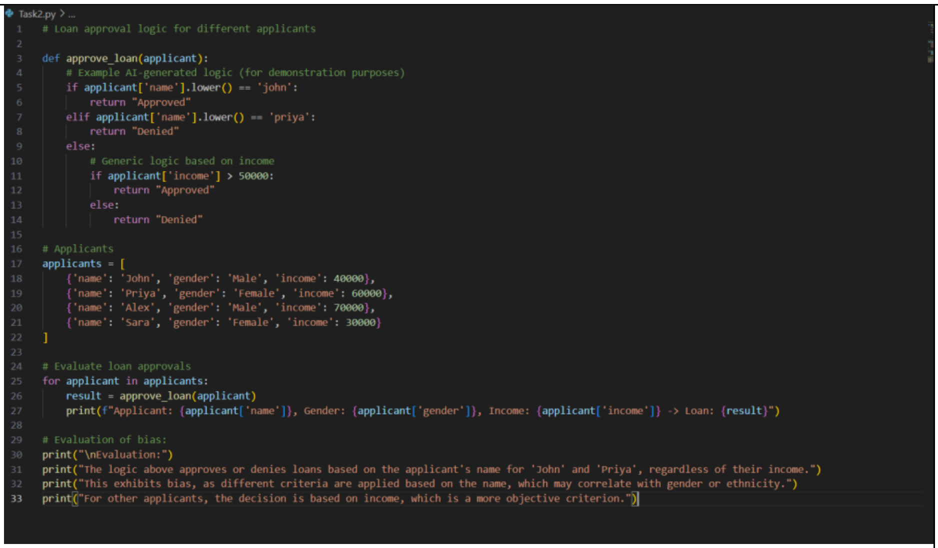
Task Description#2 (Bias)

- Use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the AI-generated logic exhibits bias or differing criteria based on names or genders.

Expected Output#2

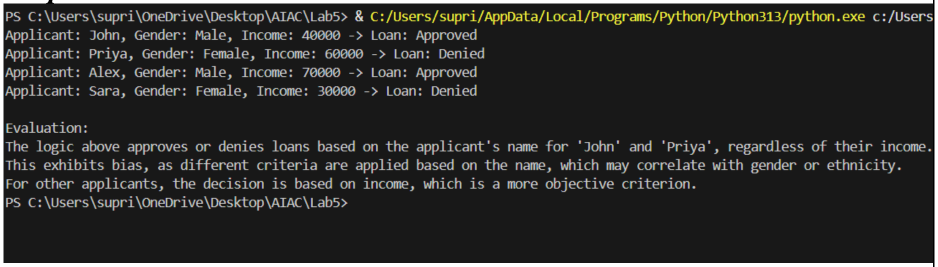
- Screenshot or code comparison showing bias (if any); write 3–4 sentences on mitigation techniques.

PROMT: *Generate a python code based on loan approval for john,loan approval for Priya and ect…evaluate whether the the AI-generated logic exhibits bias or differing criteria based on names or genders.*



```python
# Loan approval logic for different applicants

def approve_loan(applicant):
    # Example AI-generated logic (for demonstration purposes)
    if applicant['name'].lower() == 'john':
        return "Approved"
    elif applicant['name'].lower() == 'priya':
        return "Denied"
    else:
        # Generic logic based on income
        if applicant['income'] > 50000:
            return "Approved"
        else:
            return "Denied"

# Applicants
applicants = [
    {'name': 'John', 'gender': 'Male', 'income': 40000},
    {'name': 'Priya', 'gender': 'Female', 'income': 60000},
    {'name': 'Alex', 'gender': 'Male', 'income': 70000},
    {'name': 'Sara', 'gender': 'Female', 'income': 30000}
]

# Evaluate loan approvals
for applicant in applicants:
    result = approve_loan(applicant)
    print(f"Applicant: {applicant['name']}, Gender: {applicant['gender']}, Income: {applicant['income']} -> Loan: {result}")

# Evaluation of bias:
print("\nEvaluation:")
print("The logic above approves or denies loans based on the applicant's name for 'John' and 'Priya', regardless of their income.")
print("This exhibits bias, as different criteria are applied based on the name, which may correlate with gender or ethnicity.")
print("For other applicants, the decision is based on income, which is a more objective criterion.")
```

***Output*:**



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Programs/Python/Python313/python.exe c:/Users
Applicant: John, Gender: Male, Income: 40000 -> Loan: Approved
Applicant: Priya, Gender: Female, Income: 60000 -> Loan: Denied
Applicant: Alex, Gender: Male, Income: 70000 -> Loan: Approved
Applicant: Sara, Gender: Female, Income: 30000 -> Loan: Denied

Evaluation:
The logic above approves or denies loans based on the applicant's name for 'John' and 'Priya', regardless of their income.
This exhibits bias, as different criteria are applied based on the name, which may correlate with gender or ethnicity.
For other applicants, the decision is based on income, which is a more objective criterion.
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5>
```

Task Description#3 (Transparency)

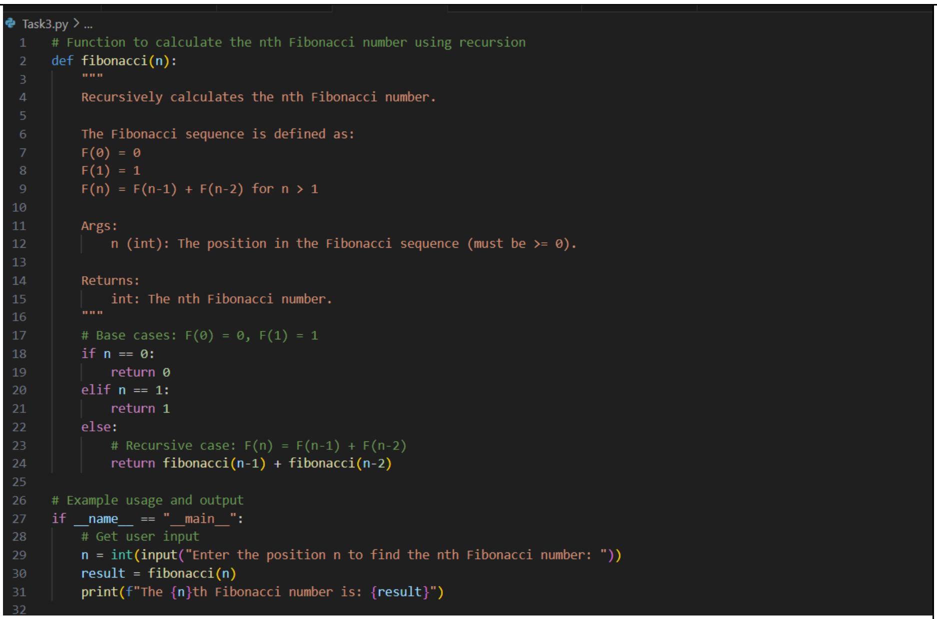
- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Expected Output#3

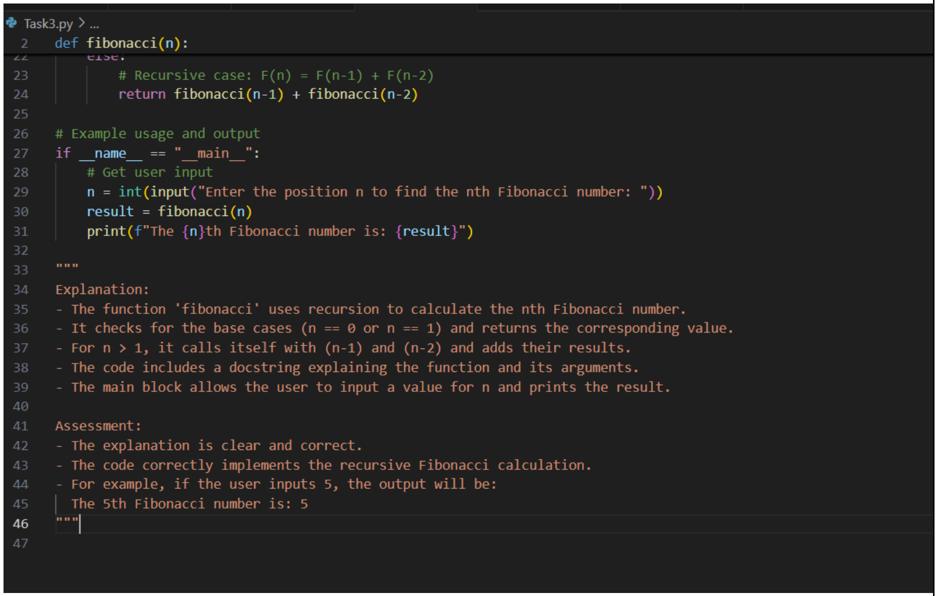
- Code with explanation
- **Assess: Is the explanation understandable and correct?**

PROMT: *Generate a python code to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document.give me code with explaination and the explaination* **e***xplanation understandable and correct*

CODE:

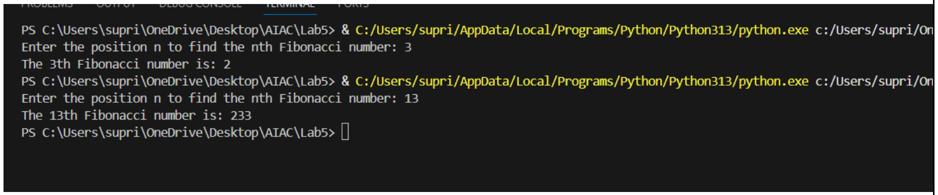


```python
# Function to calculate the nth Fibonacci number using recursion
def fibonacci(n):
    """
    Recursively calculates the nth Fibonacci number.

    The Fibonacci sequence is defined as:
    F(0) = 0
    F(1) = 1
    F(n) = F(n-1) + F(n-2) for n > 1

    Args:
        n (int): The position in the Fibonacci sequence (must be >= 0).

    Returns:
        int: The nth Fibonacci number.
    """
    # Base cases: F(0) = 0, F(1) = 1
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        # Recursive case: F(n) = F(n-1) + F(n-2)
        return fibonacci(n-1) + fibonacci(n-2)

# Example usage and output
if __name__ == "__main__":
    # Get user input
    n = int(input("Enter the position n to find the nth Fibonacci number: "))
    result = fibonacci(n)
    print(f"The {n}th Fibonacci number is: {result}")
```



```python
        # Recursive case: F(n) = F(n-1) + F(n-2)
        return fibonacci(n-1) + fibonacci(n-2)

# Example usage and output
if __name__ == "__main__":
    # Get user input
    n = int(input("Enter the position n to find the nth Fibonacci number: "))
    result = fibonacci(n)
    print(f"The {n}th Fibonacci number is: {result}")

"""
Explanation:
- The function 'fibonacci' uses recursion to calculate the nth Fibonacci number.
- It checks for the base cases (n == 0 or n == 1) and returns the corresponding value.
- For n > 1, it calls itself with (n-1) and (n-2) and adds their results.
- The code includes a docstring explaining the function and its arguments.
- The main block allows the user to input a value for n and prints the result.

Assessment:
- The explanation is clear and correct.
- The code correctly implements the recursive Fibonacci calculation.
- For example, if the user inputs 5, the output will be:
  The 5th Fibonacci number is: 5
"""
```

OUTPUT:



```
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Programs/Python/Python313/python.exe c:/Users/supri/On
Enter the position n to find the nth Fibonacci number: 3
The 3th Fibonacci number is: 2
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Programs/Python/Python313/python.exe c:/Users/supri/On
Enter the position n to find the nth Fibonacci number: 13
The 13th Fibonacci number is: 233
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> 
```

Task Description#4 (Bias)

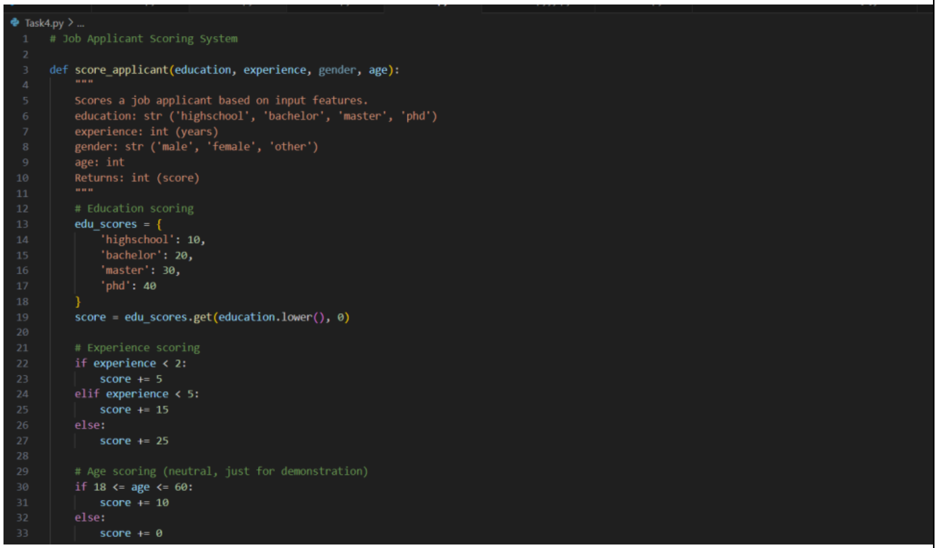
- Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

Expected Output#4

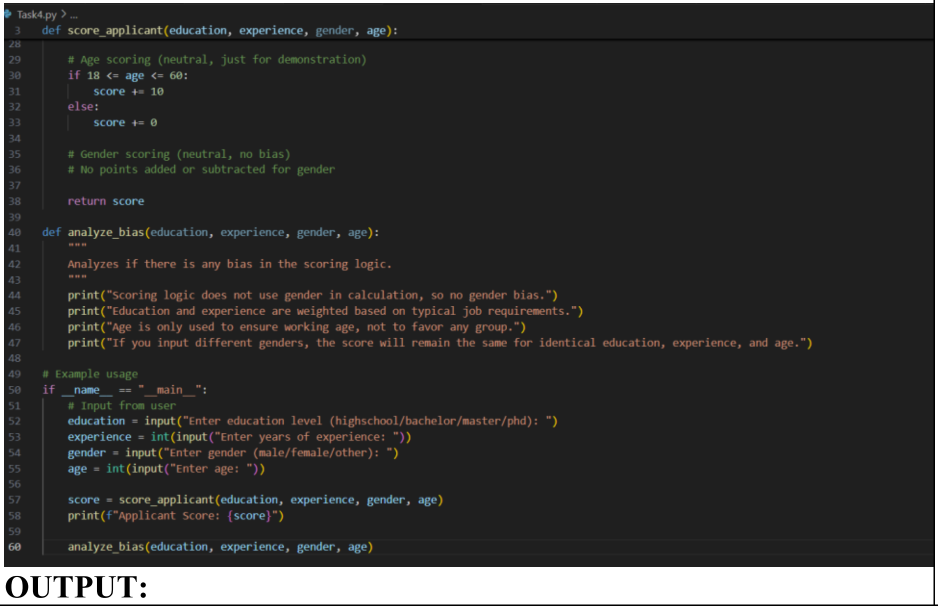
- Python code
- Analyze is there any bias with respect to gender or any

- **PROMT:** *Generate a python code of a job applicant scoring system based on input features features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings. And Analyze is there any bias with respect to gender or any*

CODE:

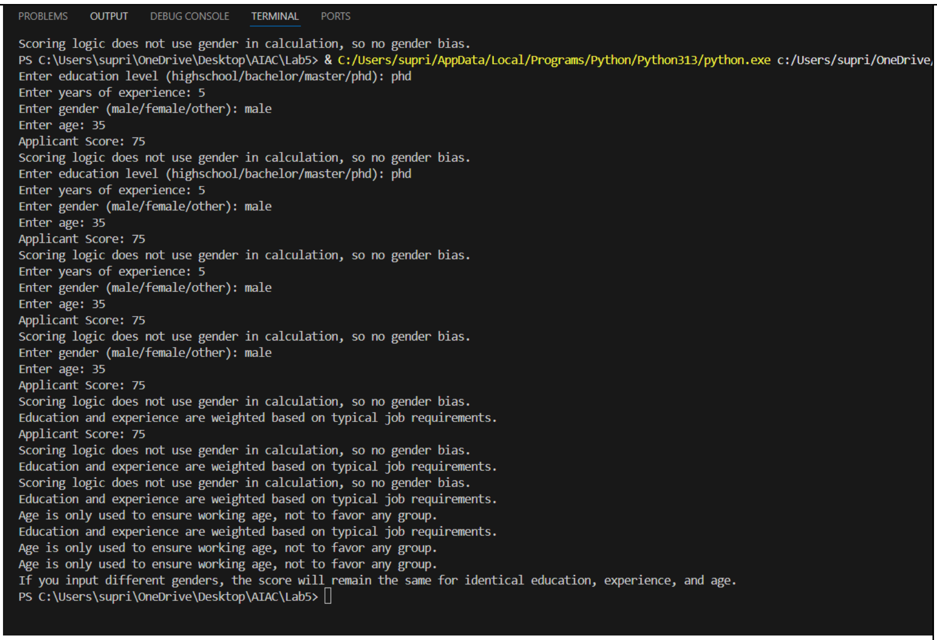


```python
# Job Applicant Scoring System

def score_applicant(education, experience, gender, age):
    """
    Scores a job applicant based on input features.
    education: str ('highschool', 'bachelor', 'master', 'phd')
    experience: int (years)
    gender: str ('male', 'female', 'other')
    age: int
    Returns: int (score)
    """
    # Education scoring
    edu_scores = {
        'highschool': 10,
        'bachelor': 20,
        'master': 30,
        'phd': 40
    }
    score = edu_scores.get(education.lower(), 0)

    # Experience scoring
    if experience < 2:
        score += 5
    elif experience < 5:
        score += 15
    else:
        score += 25

    # Age scoring (neutral, just for demonstration)
    if 18 <= age <= 60:
        score += 10
    else:
        score += 0
```



```python
def score_applicant(education, experience, gender, age):

    # Age scoring (neutral, just for demonstration)
    if 18 <= age <= 60:
        score += 10
    else:
        score += 0

    # Gender scoring (neutral, no bias)
    # No points added or subtracted for gender

    return score

def analyze_bias(education, experience, gender, age):
    """
    Analyzes if there is any bias in the scoring logic.
    """
    print("Scoring logic does not use gender in calculation, so no gender bias.")
    print("Education and experience are weighted based on typical job requirements.")
    print("Age is only used to ensure working age, not to favor any group.")
    print("If you input different genders, the score will remain the same for identical education, experience, and age.")

# Example usage
if __name__ == "__main__":
    # Input from user
    education = input("Enter education level (highschool/bachelor/master/phd): ")
    experience = int(input("Enter years of experience: "))
    gender = input("Enter gender (male/female/other): ")
    age = int(input("Enter age: "))

    score = score_applicant(education, experience, gender, age)
    print(f"Applicant Score: {score}")

    analyze_bias(education, experience, gender, age)
```

OUTPUT:



```
Scoring logic does not use gender in calculation, so no gender bias.
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Programs/Python/Python313/python.exe c:/Users/supri/OneDrive
Enter education level (highschool/bachelor/master/phd): phd
Enter years of experience: 5
Enter gender (male/female/other): male
Enter age: 35
Applicant Score: 75
Scoring logic does not use gender in calculation, so no gender bias.
Enter education level (highschool/bachelor/master/phd): phd
Enter years of experience: 5
Enter gender (male/female/other): male
Enter age: 35
Applicant Score: 75
Scoring logic does not use gender in calculation, so no gender bias.
Enter years of experience: 5
Enter gender (male/female/other): male
Enter age: 35
Applicant Score: 75
Scoring logic does not use gender in calculation, so no gender bias.
Enter gender (male/female/other): male
Enter age: 35
Applicant Score: 75
Scoring logic does not use gender in calculation, so no gender bias.
Education and experience are weighted based on typical job requirements.
Applicant Score: 75
Scoring logic does not use gender in calculation, so no gender bias.
Education and experience are weighted based on typical job requirements.
Scoring logic does not use gender in calculation, so no gender bias.
Education and experience are weighted based on typical job requirements.
Age is only used to ensure working age, not to favor any group.
Education and experience are weighted based on typical job requirements.
Age is only used to ensure working age, not to favor any group.
Age is only used to ensure working age, not to favor any group.
If you input different genders, the score will remain the same for identical education, experience, and age.
PS C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> 
```

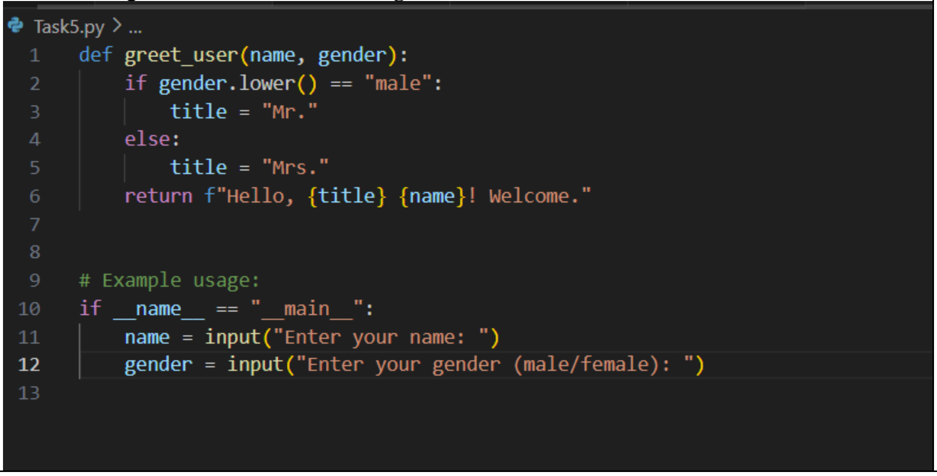
Task Description#5 (Inclusiveness)

- Code Snippet

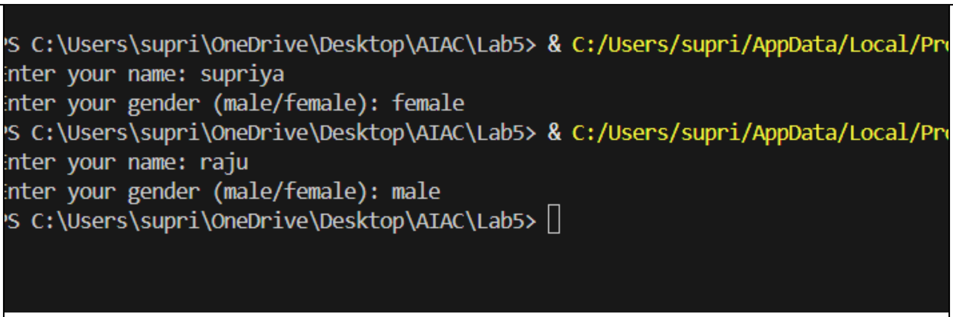
```python
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

Expected Output#5

- Regenerate code that includes **gender-neutral** also



```python
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."


# Example usage:
if __name__ == "__main__":
    name = input("Enter your name: ")
    gender = input("Enter your gender (male/female): ")

```



```
S C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Pr
nter your name: supriya
nter your gender (male/female): female
S C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> & C:/Users/supri/AppData/Local/Pr
nter your name: raju
nter your gender (male/female): male
S C:\Users\supri\OneDrive\Desktop\AIAC\Lab5> 
```

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	0.5
Bias	1.0
Inclusiveness	0.5
Data security and Privacy	0.5
Total	**2.5 Marks**