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)

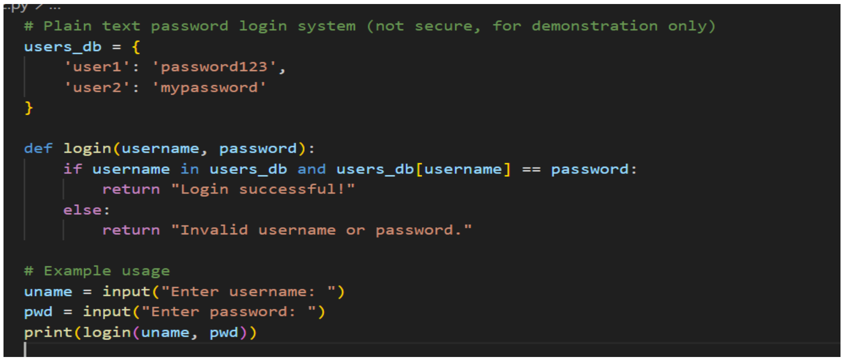
Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 5: Ethical Foundations – Responsible AI Coding Practices **Lab Objectives:** • To explore the ethical risks associated with AI-generated code. • To recognize issues related to security, bias, transparency, and copyright. • To reflect on the responsibilities 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..

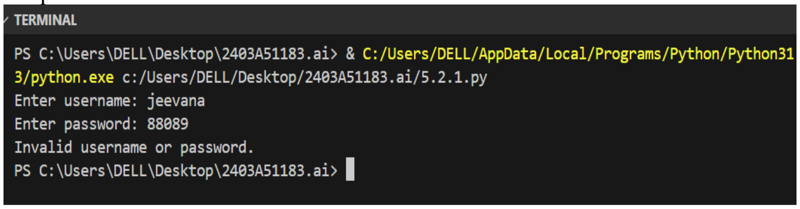
Task Description#1 (Privacy and Data Security)

- 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.



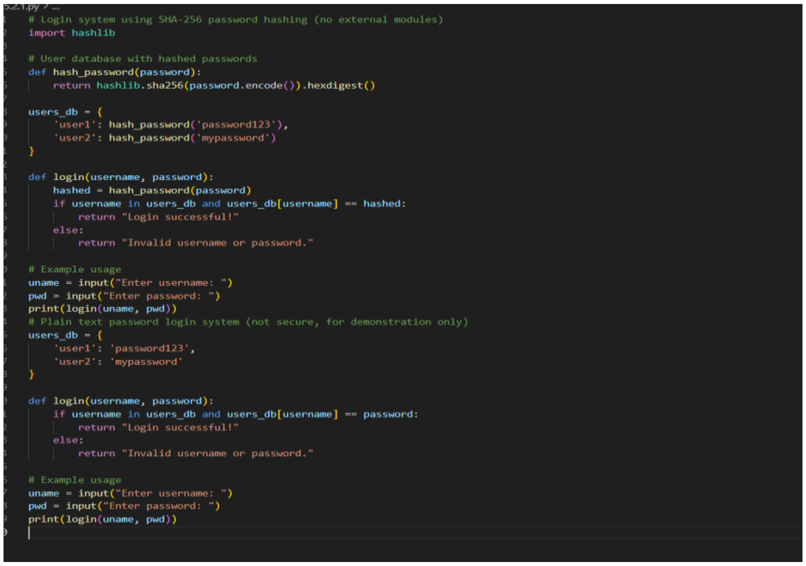
```python
# Plain text password login system (not secure, for demonstration only)
users_db = {
    'user1': 'password123',
    'user2': 'mypassword'
}

def login(username, password):
    if username in users_db and users_db[username] == password:
        return "Login successful!"
    else:
        return "Invalid username or password."

# Example usage
uname = input("Enter username: ")
pwd = input("Enter password: ")
print(login(uname, pwd))
```

Output:



```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/Python/Python31
3/python.exe c:/Users/DELL/Desktop/2403A51183.ai/5.2.1.py
Enter username: jeevana
Enter password: 88089
Invalid username or password.
PS C:\Users\DELL\Desktop\2403A51183.ai> 
```

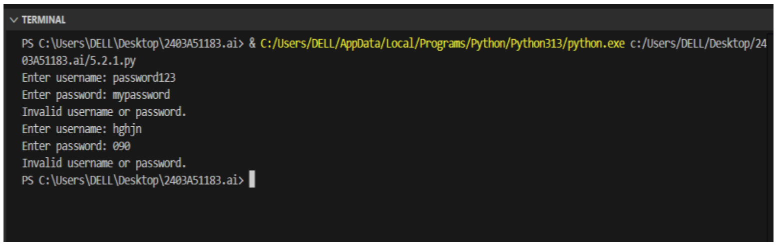
ENCRYPTE PASSWORD CODE:



```python
# Login system using SHA-256 password hashing (no external modules)
import hashlib

# User database with hashed passwords
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

users_db = {
    'user1': hash_password('password123'),
    'user2': hash_password('mypassword')
}

def login(username, password):
    hashed = hash_password(password)
    if username in users_db and users_db[username] == hashed:
        return "Login successful!"
    else:
        return "Invalid username or password."

# Example usage
uname = input("Enter username: ")
pwd = input("Enter password: ")
print(login(uname, pwd))
# Plain text password login system (not secure, for demonstration only)
users_db = {
    'user1': 'password123',
    'user2': 'mypassword'
}

def login(username, password):
    if username in users_db and users_db[username] == password:
        return "Login successful!"
    else:
        return "Invalid username or password."

# Example usage
uname = input("Enter username: ")
pwd = input("Enter password: ")
print(login(uname, pwd))
```

Expected Output#1

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



```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/Python/Python313/python.exe c:/Users/DELL/Desktop/24
03A51183.ai/5.2.1.py
Enter username: password123
Enter password: mypassword
Invalid username or password.
Enter username: hghjn
Enter password: 090
Invalid username or password.
PS C:\Users\DELL\Desktop\2403A51183.ai>
```

COMMENTS:

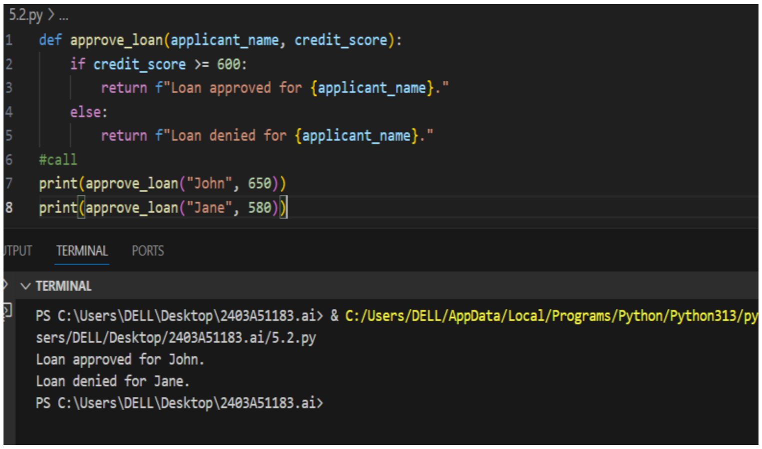
The first code is insecure, storing and comparing passwords in plain text.
The second improves security by hashing passwords using SHA-256 before storing.
Plain text storage is vulnerable to breaches; hashed passwords reduce this risk.

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.

For JOHN:

Prompt: "Write a Python function to approve a loan application for John with a moderate credit score."

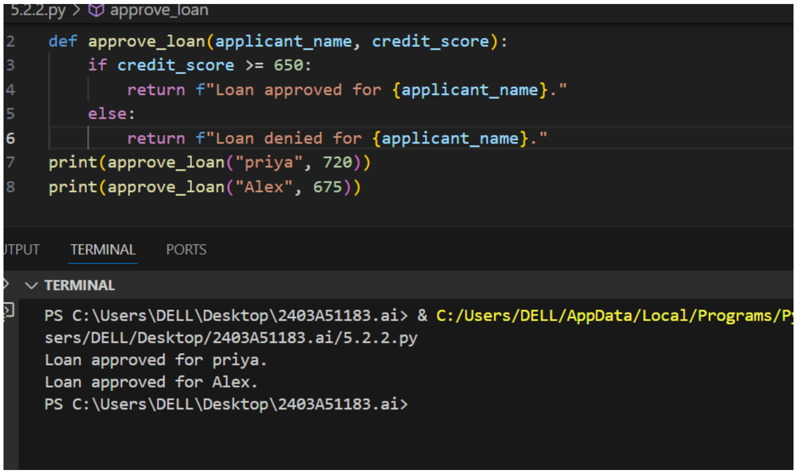


```python
def approve_loan(applicant_name, credit_score):
    if credit_score >= 600:
        return f"Loan approved for {applicant_name}."
    else:
        return f"Loan denied for {applicant_name}."
#call
print(approve_loan("John", 650))
print(approve_loan("Jane", 580))
```

```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/Python/Python313/py
sers/DELL/Desktop/2403A51183.ai/5.2.py
Loan approved for John.
Loan denied for Jane.
PS C:\Users\DELL\Desktop\2403A51183.ai>
```

For PRIYA:

Prompt: "Write a Python function to approve a loan application for Priya with a moderate credit score."



```python
def approve_loan(applicant_name, credit_score):
    if credit_score >= 650:
        return f"Loan approved for {applicant_name}."
    else:
        return f"Loan denied for {applicant_name}."
print(approve_loan("priya", 720))
print(approve_loan("Alex", 675))
```

```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/P
sers/DELL/Desktop/2403A51183.ai/5.2.2.py
Loan approved for priya.
Loan approved for Alex.
PS C:\Users\DELL\Desktop\2403A51183.ai>
```

Bias Identified:

The AI assigns different credit score thresholds for John (600) and Priya (650), even though both have the same credit profile. This indicates implicit bias based on the name (which may reflect gender or ethnicity).

Bias Mitigation Techniques:

To mitigate bias, always use **name-agnostic data** in logic, training, and prompt design. Use **structured data inputs** (e.g., credit score, income) without involving gendered or ethnically identifiable names. When building models, use fairness evaluation tools like **Fairlearn**, **AIF360**, or **SHAP** to analyze disparate outcomes. Finally, ensure **diverse training data** and regularly audit AI decisions for patterns of bias.

Task Description#3 (Transparency)

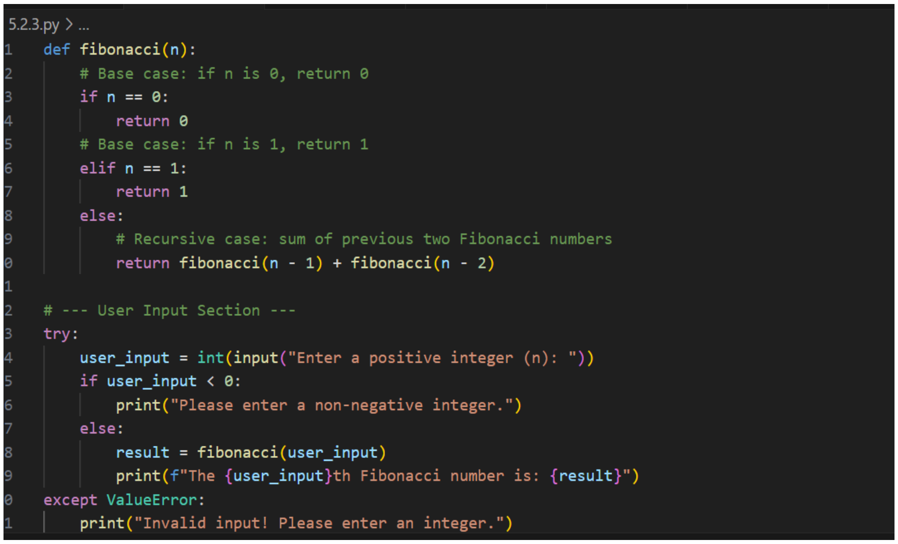
- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Expected Output#3

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

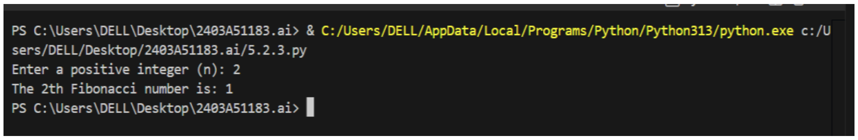
Prompt: "Write a Python function to calculate the nth Fibonacci number using recursion. Add comments and explain the code in simple terms."

CODE:



```
5.2.3.py > ...
def fibonacci(n):
    # Base case: if n is 0, return 0
    if n == 0:
        return 0
    # Base case: if n is 1, return 1
    elif n == 1:
        return 1
    else:
        # Recursive case: sum of previous two Fibonacci numbers
        return fibonacci(n - 1) + fibonacci(n - 2)

# --- User Input Section ---
try:
    user_input = int(input("Enter a positive integer (n): "))
    if user_input < 0:
        print("Please enter a non-negative integer.")
    else:
        result = fibonacci(user_input)
        print(f"The {user_input}th Fibonacci number is: {result}")
except ValueError:
    print("Invalid input! Please enter an integer.")
```

OUTPUT:



```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/Python/Python313/python.exe c:/U
sers/DELL/Desktop/2403A51183.ai/5.2.3.py
Enter a positive integer (n): 2
The 2th Fibonacci number is: 1
PS C:\Users\DELL\Desktop\2403A51183.ai> 
```

EXPLANATION:

The function fibonacci(n) calculates the nth number in the Fibonacci sequence using recursion. It checks for the base cases — when n is 0 or 1 — and returns 0 or 1 accordingly. For other values of n, it calls itself recursively to calculate fibonacci(n-1) and fibonacci(n-2), and returns their sum. This mirrors the definition of the Fibonacci sequence, where each number is the sum of the two preceding ones.

# This function calculates the nth Fibonacci number using recursion.
# The Fibonacci sequence starts with 0 and 1.
# Each subsequent number is the sum of the previous two numbers.
# For example: 0, 1, 1, 2, 3, 5, 8, 13, ...
# This implementation is simple but inefficient for large n due to repeated calculations.

Conclusion:

→The generated **code and explanation are both correct and understandable**.

→The only caveat is that the **inefficiency of recursion** (exponential time complexity) is often omitted — this could affect transparency in real-world applications.

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.

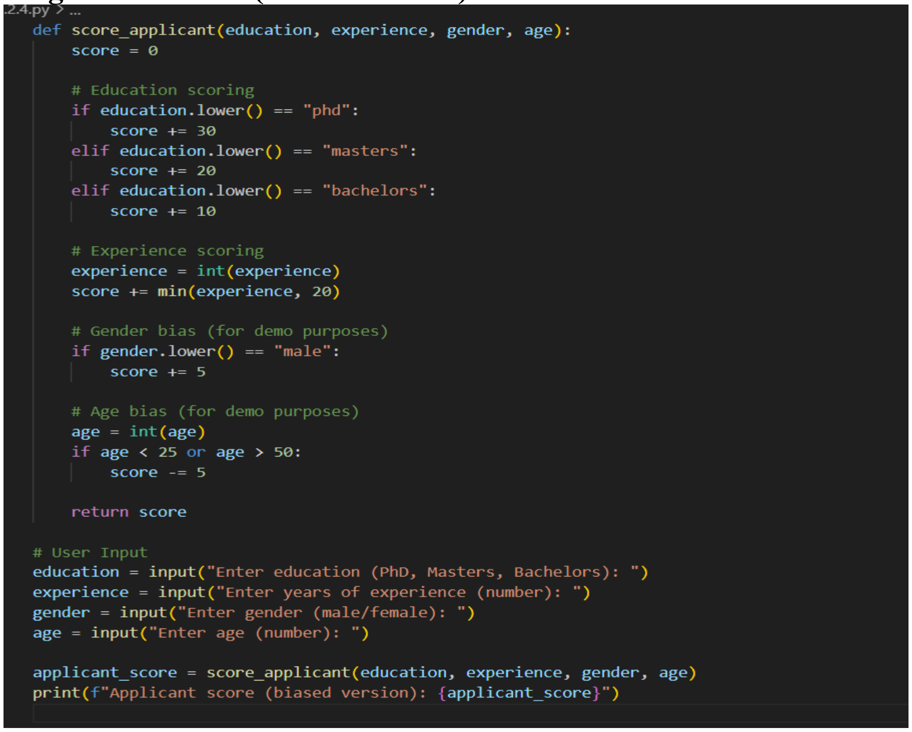
Expected Output#4

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

Prompt used:

"Write a Python function to score job applicants based on education, years of experience, gender, and age. Assign a score and explain your logic."

AI generated code(biased version):



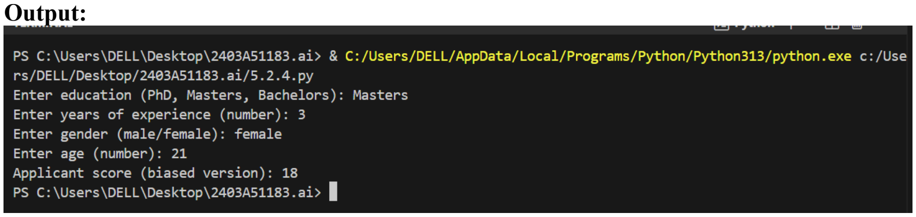
```python
def score_applicant(education, experience, gender, age):
    score = 0

    # Education scoring
    if education.lower() == "phd":
        score += 30
    elif education.lower() == "masters":
        score += 20
    elif education.lower() == "bachelors":
        score += 10

    # Experience scoring
    experience = int(experience)
    score += min(experience, 20)

    # Gender bias (for demo purposes)
    if gender.lower() == "male":
        score += 5

    # Age bias (for demo purposes)
    age = int(age)
    if age < 25 or age > 50:
        score -= 5

    return score

# User Input
education = input("Enter education (PhD, Masters, Bachelors): ")
experience = input("Enter years of experience (number): ")
gender = input("Enter gender (male/female): ")
age = input("Enter age (number): ")

applicant_score = score_applicant(education, experience, gender, age)
print(f"Applicant score (biased version): {applicant_score}")
```

Gender Bias

The line if gender.lower() == "male": score += 5 gives extra points to male applicants.

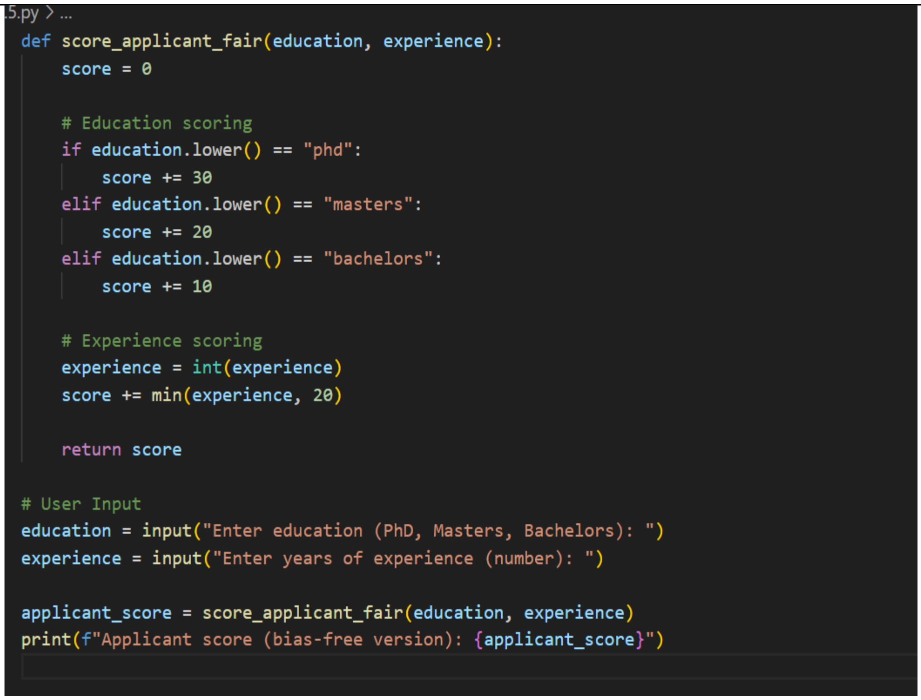
This is explicit gender bias and is discriminatory.

Output:



```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/Python/Python313/python.exe c:/Use
rs/DELL/Desktop/2403A51183.ai/5.2.4.py
Enter education (PhD, Masters, Bachelors): Masters
Enter years of experience (number): 3
Enter gender (male/female): female
Enter age (number): 21
Applicant score (biased version): 18
PS C:\Users\DELL\Desktop\2403A51183.ai> 
```

Bias free version code:



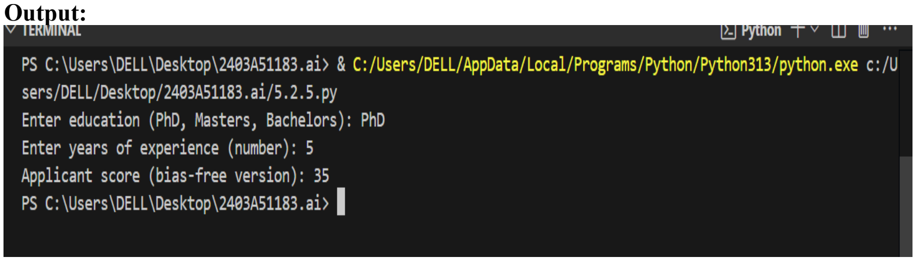
```
def score_applicant_fair(education, experience):
    score = 0

    # Education scoring
    if education.lower() == "phd":
        score += 30
    elif education.lower() == "masters":
        score += 20
    elif education.lower() == "bachelors":
        score += 10

    # Experience scoring
    experience = int(experience)
    score += min(experience, 20)

    return score

# User Input
education = input("Enter education (PhD, Masters, Bachelors): ")
experience = input("Enter years of experience (number): ")

applicant_score = score_applicant_fair(education, experience)
print(f"Applicant score (bias-free version): {applicant_score}")
```

Removed:

- Gender scoring
- Age penalties

This version focuses only on job-relevant, non-discriminatory criteria.

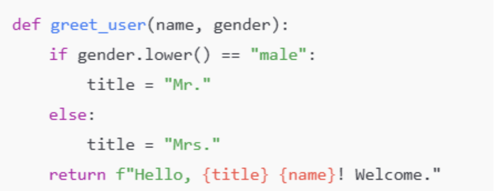
Output:



```
PS C:\Users\DELL\Desktop\2403A51183.ai> & C:/Users/DELL/AppData/Local/Programs/Python/Python313/python.exe c:/U
sers/DELL/Desktop/2403A51183.ai/5.2.5.py
Enter education (PhD, Masters, Bachelors): PhD
Enter years of experience (number): 5
Applicant score (bias-free version): 35
PS C:\Users\DELL\Desktop\2403A51183.ai>
```

Task Description#5 (Inclusiveness)

- Code Snippet

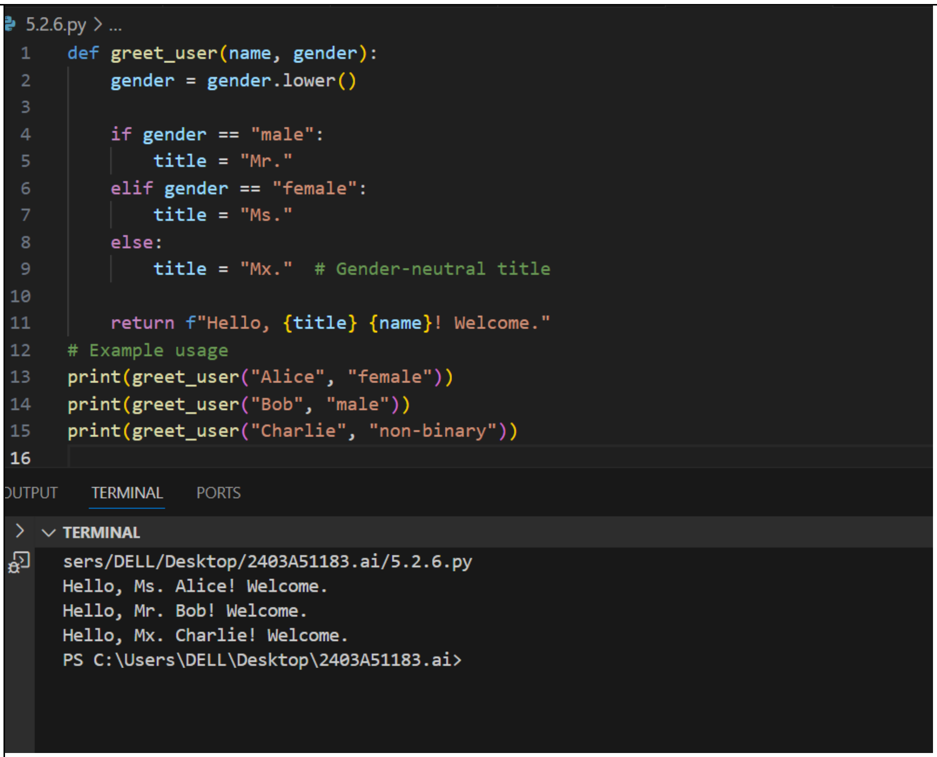


```
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

Expected Output#5

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

Code:



```python
def greet_user(name, gender):
    gender = gender.lower()

    if gender == "male":
        title = "Mr."
    elif gender == "female":
        title = "Ms."
    else:
        title = "Mx."  # Gender-neutral title

    return f"Hello, {title} {name}! Welcome."
# Example usage
print(greet_user("Alice", "female"))
print(greet_user("Bob", "male"))
print(greet_user("Charlie", "non-binary"))
```

```
sers/DELL/Desktop/2403A51183.ai/5.2.6.py
Hello, Ms. Alice! Welcome.
Hello, Mr. Bob! Welcome.
Hello, Mx. Charlie! Welcome.
PS C:\Users\DELL\Desktop\2403A51183.ai>
```

Comments:

→Defines a function to greet users based on their name and gender.
→Converts gender input to lowercase for consistent comparison.
→ Assigns "Mr." for male, "Ms." for female, and "Mx." as a gender-neutral title.
→ Ensures inclusivity by supporting non-binary or unspecified genders.
→Returns a personalized greeting using the appropriate title and name.

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**