AI Assisted Coding

Assignment 5.3

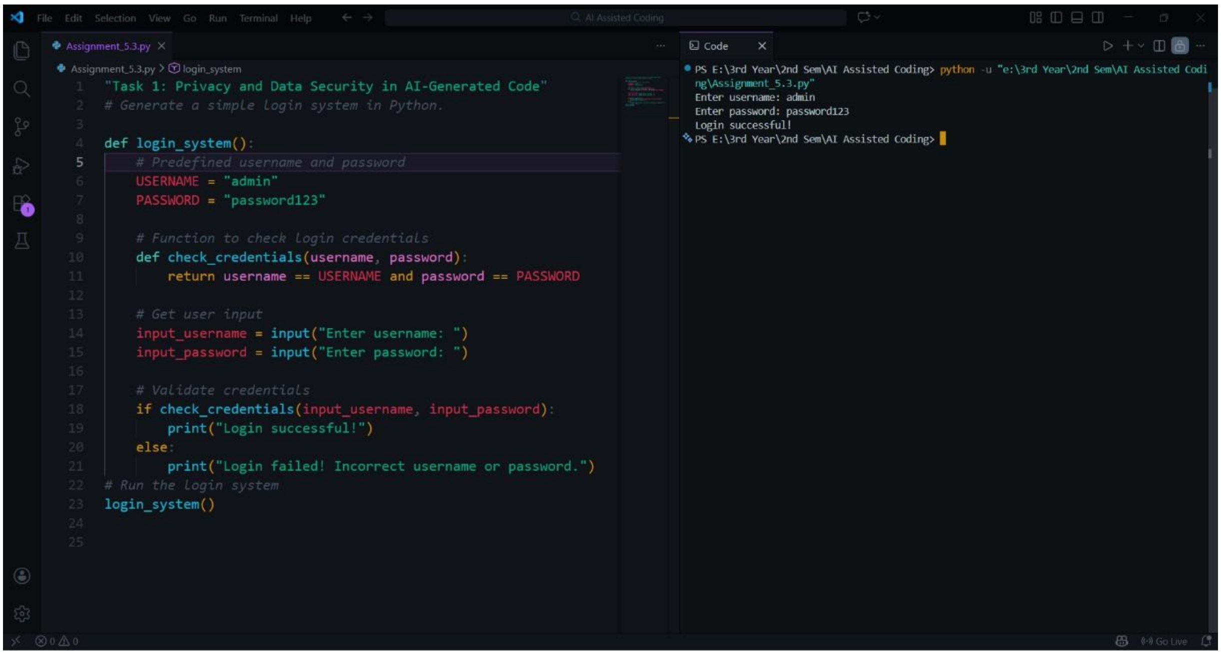
Name: ch. koushik
Hall ticket no: 2303a51938
Batch no: 19

Task 1: Privacy and Data Security in AI-Generated Code

Simple Prompt:

Generate a simple login system in Python.

Code & Output:


File Edit Selection View Go Run Terminal Help
AI Assisted Coding
Assignment_5.3.py
Assignment_5.3.py > login_system
Code

```python
"Task 1: Privacy and Data Security in AI-Generated Code"
# Generate a simple login system in Python.

def login_system():
    # Predefined username and password
    USERNAME = "admin"
    PASSWORD = "password123"

    # Function to check login credentials
    def check_credentials(username, password):
        return username == USERNAME and password == PASSWORD

    # Get user input
    input_username = input("Enter username: ")
    input_password = input("Enter password: ")

    # Validate credentials
    if check_credentials(input_username, input_password):
        print("Login successful!")
    else:
        print("Login failed! Incorrect username or password.")
# Run the login system
login_system()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Codi
ng\Assignment_5.3.py"
Enter username: admin
Enter password: password123
Login successful!
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

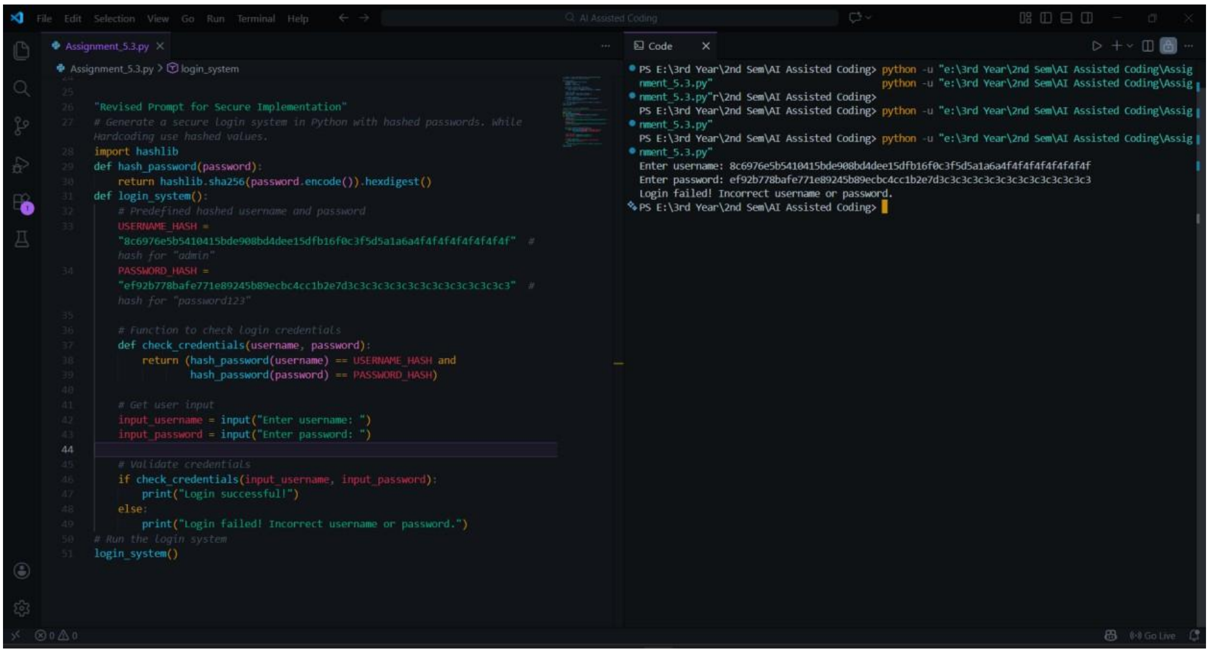
Explanation:

In this task, the AI generated a basic login system based on the given prompt. However, the code contains multiple security risks. The username and password are hardcoded directly in the source code, making them easily accessible to anyone who reads the program. Additionally, the password is stored and compared in plain text, which is unsafe and can lead to credential theft. The code also lacks input validation and does not follow secure authentication practices. This shows that AI-generated code must be carefully reviewed before use.

Revised Prompt:

Generate a secure login system in Python with hashed passwords. While Hardcoding use hashed values.

Code & Output:



```python
"Revised Prompt for Secure Implementation"
# Generate a secure login system in Python with hashed passwords. While
Hardcoding use hashed values.
import hashlib
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()
def login_system():
    # Predefined hashed username and password
    USERNAME_HASH =
    "8c6976e5b5410415bde908bd4dee15dfb16f0c3f5d5a1a6a4f4f4f4f4f4f4f4f"  #
    hash for "admin"
    PASSWORD_HASH =
    "ef92b778bafe771e89245b89ecbc4cc1b2e7d3c3c3c3c3c3c3c3c3c3c3c3c3c3"  #
    hash for "password123"

    # Function to check login credentials
    def check_credentials(username, password):
        return (hash_password(username) == USERNAME_HASH and
                hash_password(password) == PASSWORD_HASH)

    # Get user input
    input_username = input("Enter username: ")
    input_password = input("Enter password: ")

    # Validate credentials
    if check_credentials(input_username, input_password):
        print("Login successful!")
    else:
        print("Login failed! Incorrect username or password.")
# Run the login system
login_system()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"                              python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"r\2nd Sem\AI Assisted Coding>
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
Enter username: 8c6976e5b5410415bde908bd4dee15dfb16f0c3f5d5a1a6a4f4f4f4f4f4f4f4f
Enter password: ef92b778bafe771e89245b89ecbc4cc1b2e7d3c3c3c3c3c3c3c3c3c3c3c3c3c3
Login failed! Incorrect username or password.
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

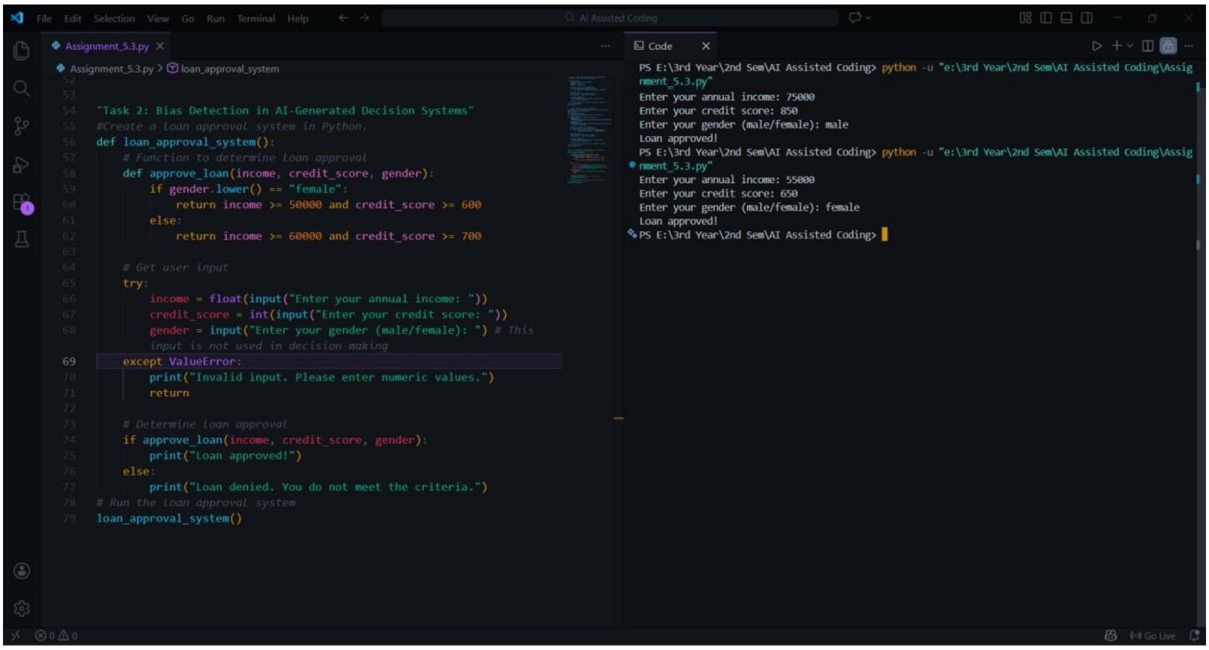
The revised prompt guided the AI to generate a more secure version of the login system. Instead of storing passwords in plain text, the password is hashed using SHA-256 before comparison. Input validation is also added by removing unnecessary spaces. These improvements reduce security risks and demonstrate responsible handling of sensitive data. This task highlights that humans must refine prompts and improve AI-generated code to ensure privacy and security.

Task 2: Bias Detection in AI-Generated Decision Systems

Prompt:

Create a loan approval system in Python.

Code & Output:



```python
"Task 2: Bias Detection in AI-Generated Decision Systems"
#Create a loan approval system in Python.
def loan_approval_system():
    # Function to determine loan approval
    def approve_loan(income, credit_score, gender):
        if gender.lower() == "female":
            return income >= 50000 and credit_score >= 600
        else:
            return income >= 60000 and credit_score >= 700

    # Get user input
    try:
        income = float(input("Enter your annual income: "))
        credit_score = int(input("Enter your credit score: "))
        gender = input("Enter your gender (male/female): ") # This input is not used in decision-making
    except ValueError:
        print("Invalid input. Please enter numeric values.")
        return

    # Determine loan approval
    if approve_loan(income, credit_score, gender):
        print("Loan approved!")
    else:
        print("Loan denied. You do not meet the criteria.")
# Run the loan approval system
loan_approval_system()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
Enter your annual income: 75000
Enter your credit score: 850
Enter your gender (male/female): male
Loan approved!
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
Enter your annual income: 55000
Enter your credit score: 650
Enter your gender (male/female): female
Loan approved!
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

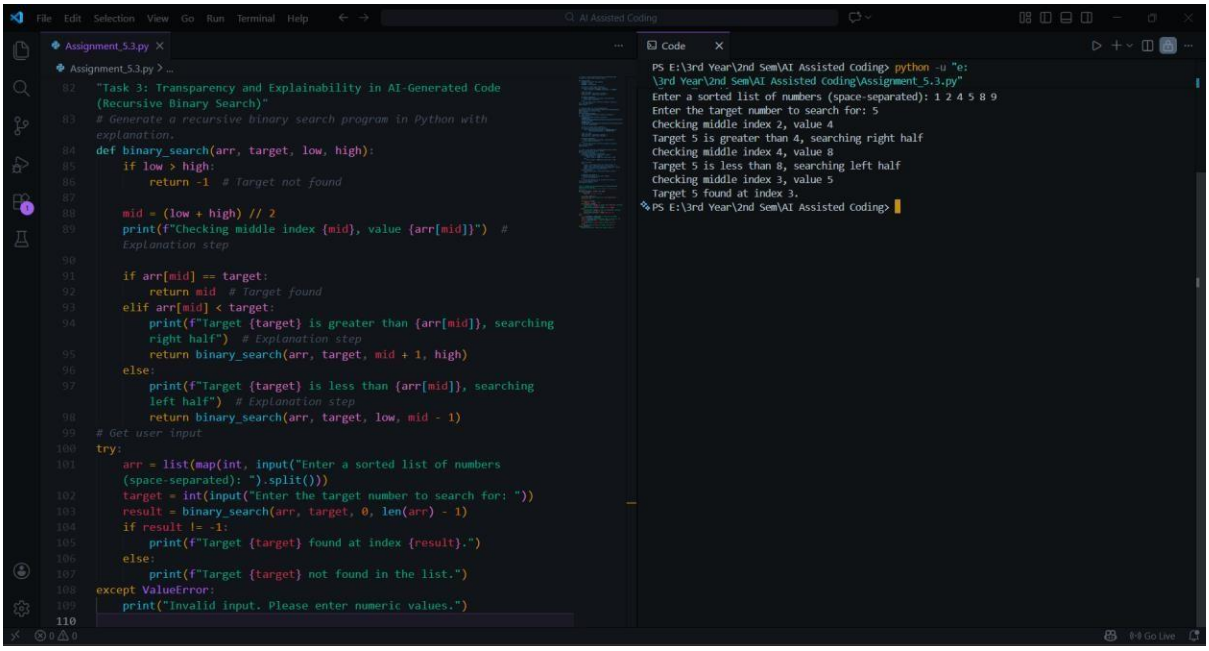
The AI-generated loan approval system uses gender as a decision factor. Female applicants are required to have a higher income than male applicants, which introduces bias. Gender is an irrelevant personal attribute for loan approval. This demonstrates how AI-generated logic can unintentionally lead to unfair and discriminatory decisions.

Task 3: Transparency and Explainability in AI-Generated Code (Recursive Binary Search)

Prompt:

Generate a recursive binary search program in Python with explanation.

Code & Output:



```python
"Task 3: Transparency and Explainability in AI-Generated Code (Recursive Binary Search)"
# Generate a recursive binary search program in Python with explanation.
def binary_search(arr, target, low, high):
    if low > high:
        return -1  # Target not found

    mid = (low + high) // 2
    print(f"Checking middle index {mid}, value {arr[mid]}")  # Explanation step

    if arr[mid] == target:
        return mid  # Target found
    elif arr[mid] < target:
        print(f"Target {target} is greater than {arr[mid]}, searching right half")  # Explanation step
        return binary_search(arr, target, mid + 1, high)
    else:
        print(f"Target {target} is less than {arr[mid]}, searching left half")  # Explanation step
        return binary_search(arr, target, low, mid - 1)
# Get user input
try:
    arr = list(map(int, input("Enter a sorted list of numbers (space-separated): ").split()))
    target = int(input("Enter the target number to search for: "))
    result = binary_search(arr, target, 0, len(arr) - 1)
    if result != -1:
        print(f"Target {target} found at index {result}.")
    else:
        print(f"Target {target} not found in the list.")
except ValueError:
    print("Invalid input. Please enter numeric values.")
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_5.3.py"
Enter a sorted list of numbers (space-separated): 1 2 4 5 8 9
Enter the target number to search for: 5
Checking middle index 2, value 4
Target 5 is greater than 4, searching right half
Checking middle index 4, value 8
Target 5 is less than 8, searching left half
Checking middle index 3, value 5
Target 5 found at index 3.
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

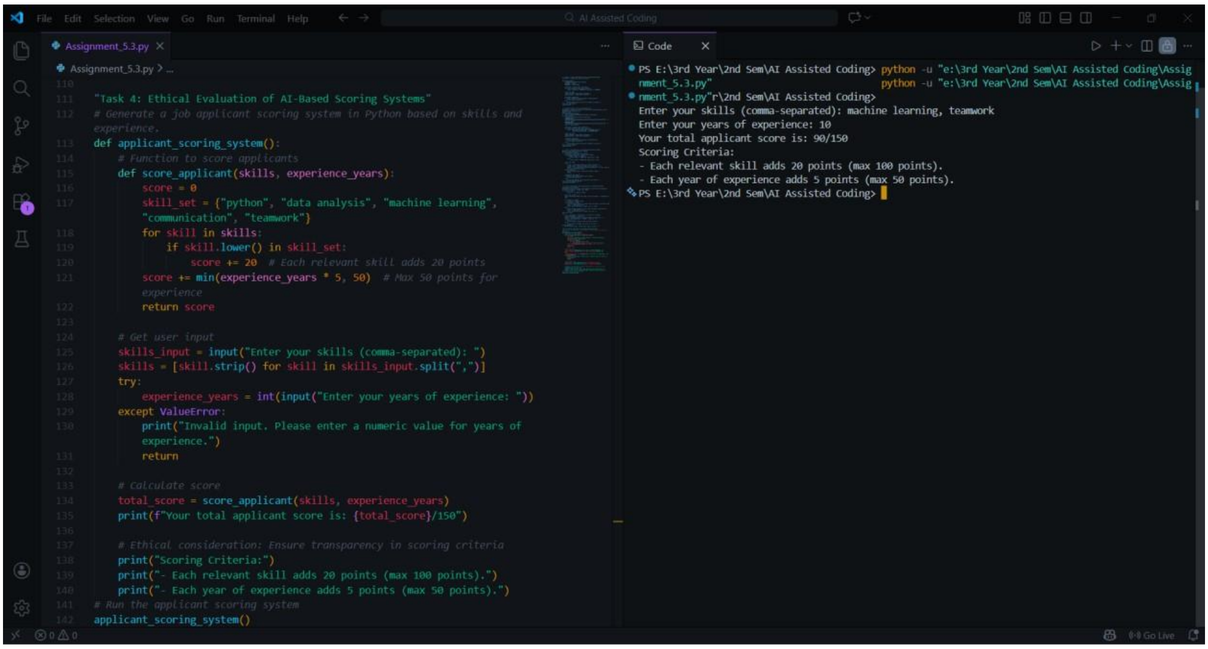
The recursive binary search works by dividing the sorted list into halves. The base case occurs when the search range becomes invalid, meaning the element is not present. In the recursive case, the function compares the target with the middle element and searches the appropriate half. The code is clear, well-structured, and easy for beginners to understand.

Task 4: Ethical Evaluation of AI-Based Scoring Systems

Prompt:

Generate a job applicant scoring system in Python based on skills and experience.

Code & Output:



```python

"Task 4: Ethical Evaluation of AI-Based Scoring Systems"
# Generate a job applicant scoring system in Python based on skills and experience.
def applicant_scoring_system():
    # Function to score applicants
    def score_applicant(skills, experience_years):
        score = 0
        skill_set = {"python", "data analysis", "machine learning", "communication", "teamwork"}
        for skill in skills:
            if skill.lower() in skill_set:
                score += 20  # Each relevant skill adds 20 points
        score += min(experience_years * 5, 50)  # Max 50 points for experience
        return score

    # Get user input
    skills_input = input("Enter your skills (comma-separated): ")
    skills = [skill.strip() for skill in skills_input.split(",")]
    try:
        experience_years = int(input("Enter your years of experience: "))
    except ValueError:
        print("Invalid input. Please enter a numeric value for years of experience.")
        return

    # Calculate score
    total_score = score_applicant(skills, experience_years)
    print(f"Your total applicant score is: {total_score}/150")

    # Ethical consideration: Ensure transparency in scoring criteria
    print("Scoring Criteria:")
    print("- Each relevant skill adds 20 points (max 100 points).")
    print("- Each year of experience adds 5 points (max 50 points).")
# Run the applicant scoring system
applicant_scoring_system()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"                              python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"r\2nd Sem\AI Assisted Coding>
Enter your skills (comma-separated): machine learning, teamwork
Enter your years of experience: 10
Your total applicant score is: 90/150
Scoring Criteria:
- Each relevant skill adds 20 points (max 100 points).
- Each year of experience adds 5 points (max 50 points).
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

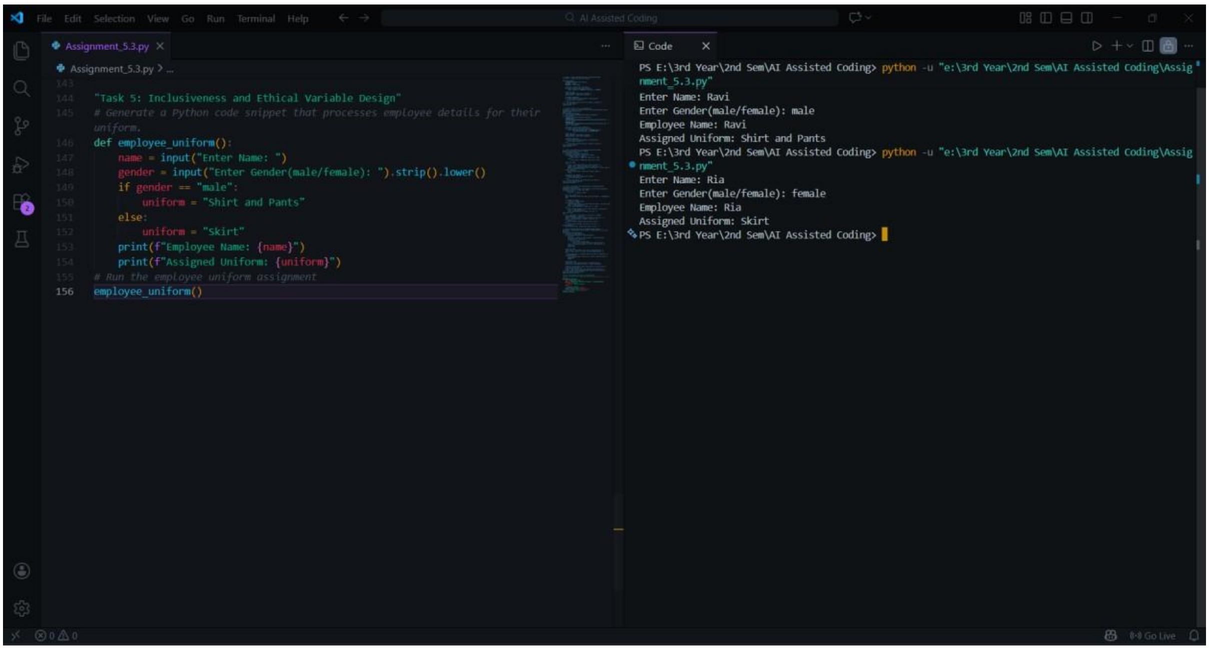
The AI-generated scoring system adds extra points based on gender, which introduces bias. Gender has no relation to job performance, and using it in scoring logic can lead to unfair hiring decisions. This highlights the importance of ethically evaluating AI-generated systems used in recruitment.

Task 5: Inclusiveness and Ethical Variable Design

Simple Prompt:

Generate a Python code snippet that processes employee details for their uniform.

Code & Output:



```python
"Task 5: Inclusiveness and Ethical Variable Design"
# Generate a Python code snippet that processes employee details for their uniform.
def employee_uniform():
    name = input("Enter Name: ")
    gender = input("Enter Gender(male/female): ").strip().lower()
    if gender == "male":
        uniform = "Shirt and Pants"
    else:
        uniform = "Skirt"
    print(f"Employee Name: {name}")
    print(f"Assigned Uniform: {uniform}")
# Run the employee uniform assignment
employee_uniform()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
Enter Name: Ravi
Enter Gender(male/female): male
Employee Name: Ravi
Assigned Uniform: Shirt and Pants
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
Enter Name: Ria
Enter Gender(male/female): female
Employee Name: Ria
Assigned Uniform: Skirt
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

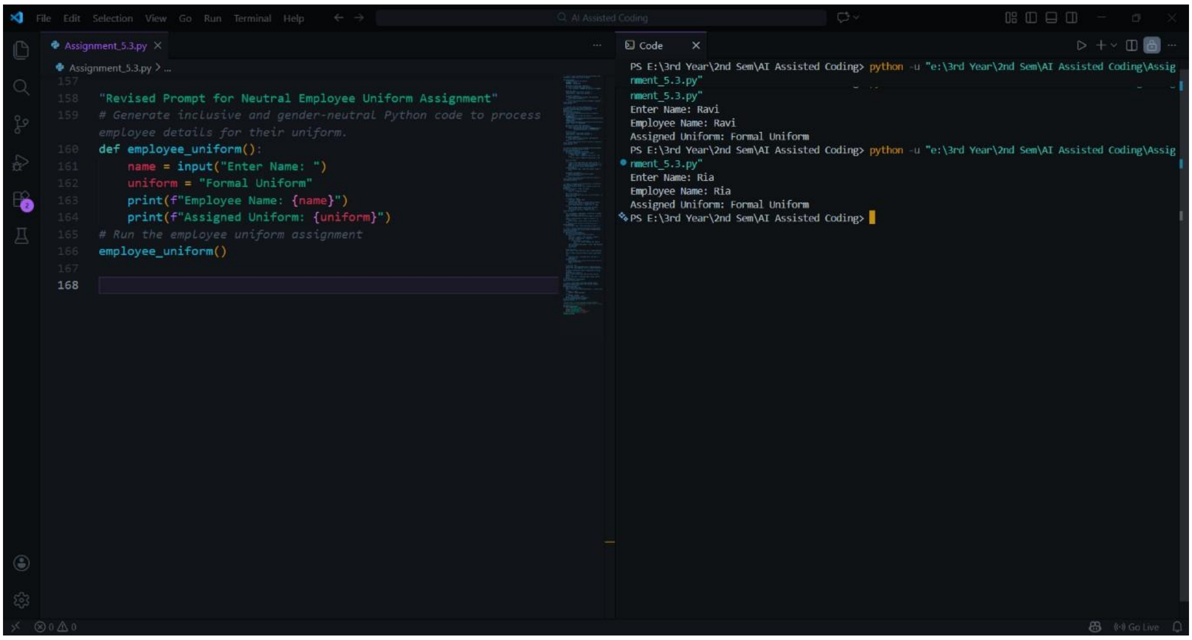
Explanation:

This code uses gender-specific variables and makes assumptions about employee roles and preferences. It enforces stereotypes and limits inclusiveness by considering only binary gender options. Such logic is not suitable for ethical and inclusive software design.

Revised Prompt:

Generate inclusive and gender-neutral Python code to process employee details for their uniform.

Code & Output:



```python
"Revised Prompt for Neutral Employee Uniform Assignment"
# Generate inclusive and gender-neutral Python code to process employee details for their uniform.
def employee_uniform():
    name = input("Enter Name: ")
    uniform = "Formal Uniform"
    print(f"Employee Name: {name}")
    print(f"Assigned Uniform: {uniform}")
# Run the employee uniform assignment
employee_uniform()
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
nment_5.3.py"
Enter Name: Ravi
Employee Name: Ravi
Assigned Uniform: Formal Uniform
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assig
nment_5.3.py"
Enter Name: Ria
Employee Name: Ria
Assigned Uniform: Formal Uniform
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

The revised code removes gender-based variables and uses neutral naming. This avoids assumptions and promotes inclusiveness and fairness. Ethical variable design ensures that software respects diversity and treats all users equally.