1934_AI_AC_5.3

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

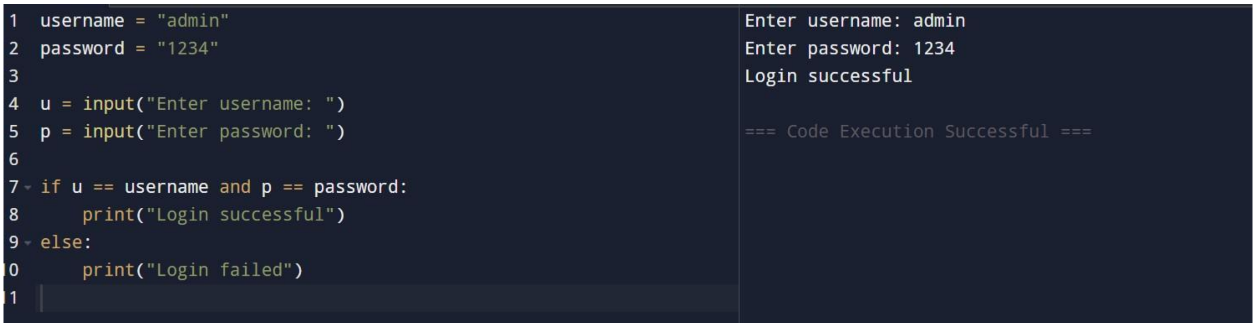
Scenario

AI tools can sometimes generate insecure authentication logic.

AI Prompt Used

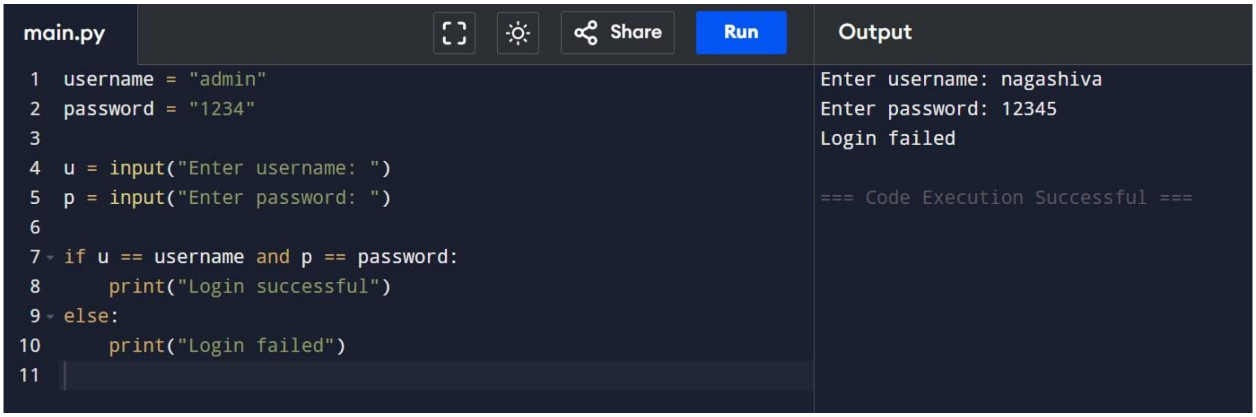
“Generate a simple login system in Python.”

AI-Generated Login Code



```python
username = "admin"
password = "1234"

u = input("Enter username: ")
p = input("Enter password: ")

if u == username and p == password:
    print("Login successful")
else:
    print("Login failed")
```

```
Enter username: admin
Enter password: 1234
Login successful

=== Code Execution Successful ===
```



```python
username = "admin"
password = "1234"

u = input("Enter username: ")
p = input("Enter password: ")

if u == username and p == password:
    print("Login successful")
else:
    print("Login failed")
```

```
Enter username: nagashiva
Enter password: 12345
Login failed

=== Code Execution Successful ===
```

Security Issues Identified

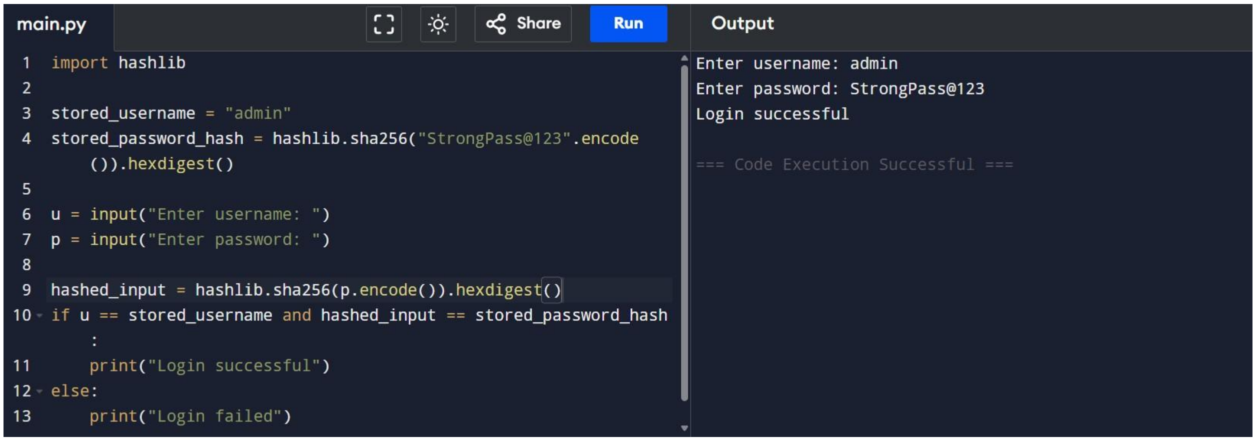
1. **Hardcoded credentials** (username and password written directly in code)

2. **Plain text password comparison**

3. **No input validation**

4. **Not scalable or secure for real systems**

Revised Secure Version of the Code



```python
import hashlib

stored_username = "admin"
stored_password_hash = hashlib.sha256("StrongPass@123".encode()).hexdigest()

u = input("Enter username: ")
p = input("Enter password: ")

hashed_input = hashlib.sha256(p.encode()).hexdigest()
if u == stored_username and hashed_input == stored_password_hash:
    print("Login successful")
else:
    print("Login failed")
```

```
Enter username: admin
Enter password: StrongPass@123
Login successful

=== Code Execution Successful ===
```

Explanation of Improvements

- Removed plain-text password comparison
- Used **password hashing** to improve security
- Reduced risk of credential leakage
- Demonstrates better authentication practices

Task 2: Bias Detection in AI-Generated Decision Systems

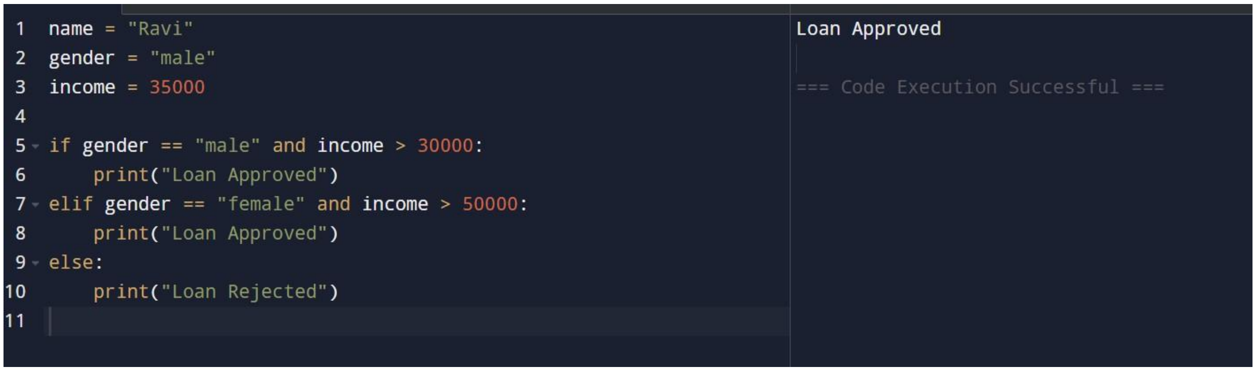
Scenario

AI systems may unintentionally introduce bias.

AI Prompt Used

“Create a loan approval system in Python.”

AI-Generated Loan Approval Code



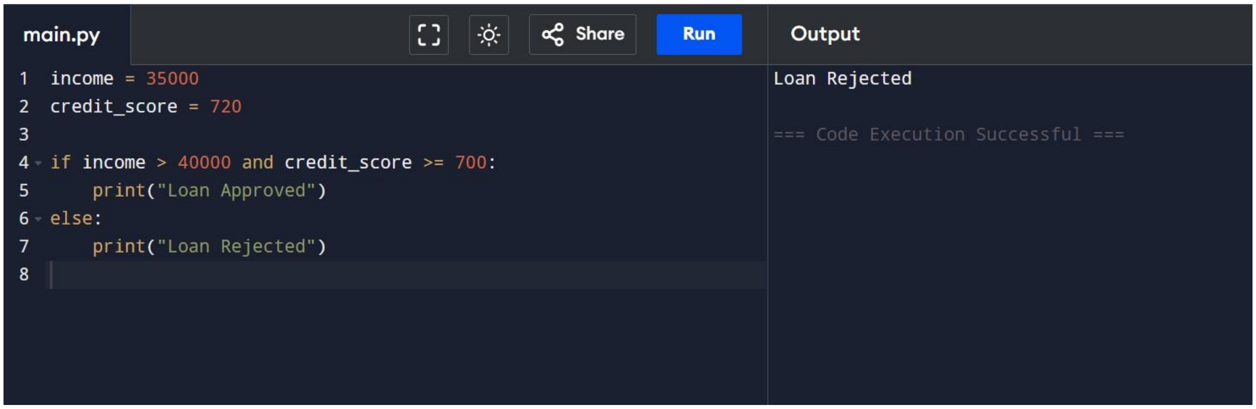
```python
name = "Ravi"
gender = "male"
income = 35000

if gender == "male" and income > 30000:
    print("Loan Approved")
elif gender == "female" and income > 50000:
    print("Loan Approved")
else:
    print("Loan Rejected")

```

```
Loan Approved

=== Code Execution Successful ===
```

Bias Identified

- Different rules for male and female
- Gender should not affect loan approval

Revised Fair Code



```python
income = 35000
credit_score = 720

if income > 40000 and credit_score >= 700:
    print("Loan Approved")
else:
    print("Loan Rejected")

```

```
Loan Rejected

=== Code Execution Successful ===
```

Discussion on Fairness

- Removed gender completely

- Decisions based on **financial factors only**
- Promotes fairness and equality

Bias Mitigation Strategies

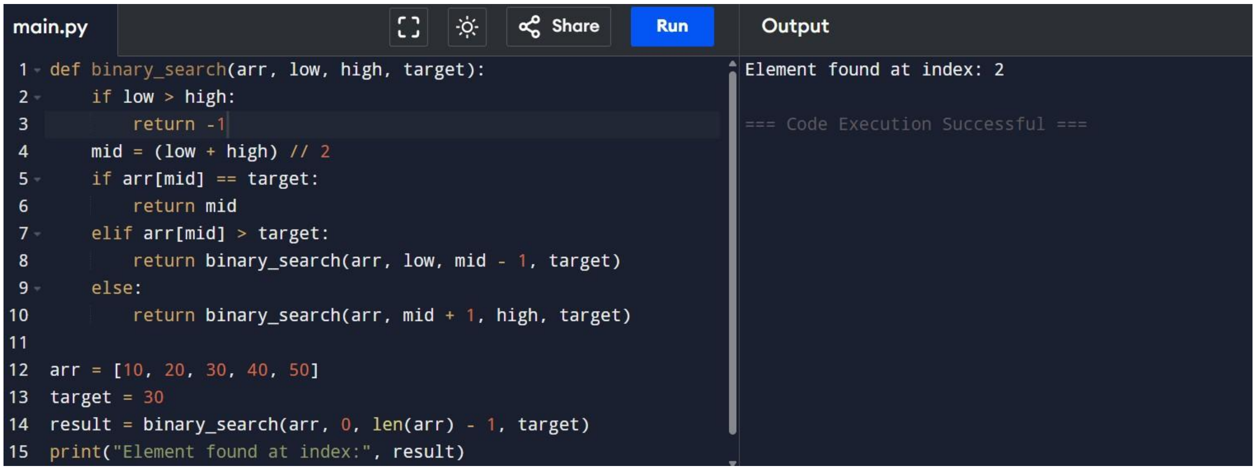
- Avoid personal attributes like gender or name
- Use relevant, objective data
- Regular bias audits and human review

Task 3: Transparency & Explainability (Recursive Binary Search)

Scenario

AI-generated code should be easy to understand and verify.

AI-Generated Recursive Binary Search Code



```python
def binary_search(arr, low, high, target):
    if low > high:
        return -1
    mid = (low + high) // 2
    if arr[mid] == target:
        return mid
    elif arr[mid] > target:
        return binary_search(arr, low, mid - 1, target)
    else:
        return binary_search(arr, mid + 1, high, target)

arr = [10, 20, 30, 40, 50]
target = 30
result = binary_search(arr, 0, len(arr) - 1, target)
print("Element found at index:", result)
```

```
Element found at index: 2

=== Code Execution Successful ===
```

Step-by-Step Explanation

- **Base Case:** When low > high, element does not exist

- **Recursive Case:**
    - Compare middle element with target
    - Search left or right half accordingly

Student Assessment

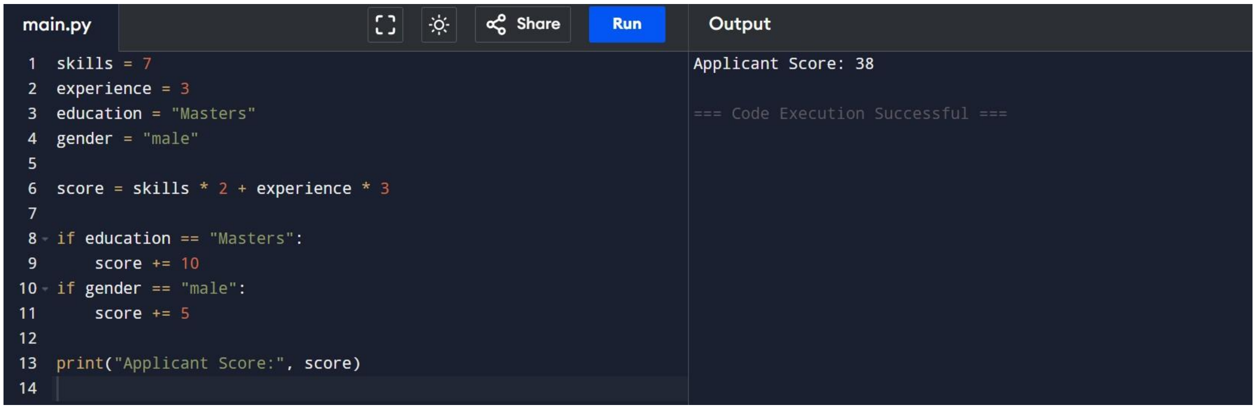
- Comments clearly match the code
- Base case and recursive case are explained well
- Easy to understand for beginner-level students
- Transparent and readable logic

Task 4: Ethical Evaluation of AI-Based Scoring Systems

Scenario

AI scoring systems can affect hiring decisions.

AI-Generated Scoring System Code



```python
skills = 7
experience = 3
education = "Masters"
gender = "male"

score = skills * 2 + experience * 3

if education == "Masters":
    score += 10
if gender == "male":
    score += 5

print("Applicant Score:", score)

```

```
Applicant Score: 38

=== Code Execution Successful ===
```

Ethical Issues Identified

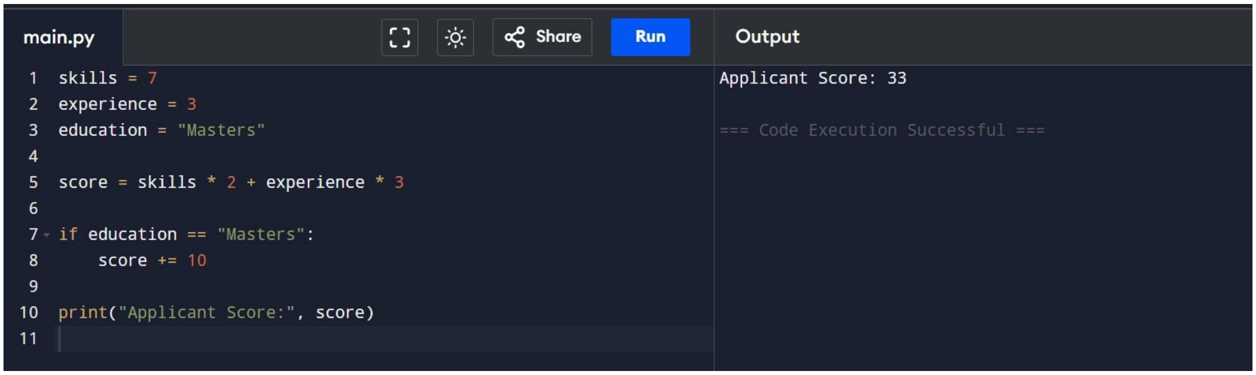
- Gender directly affects the score

- Gender is **irrelevant** for job performance
- Leads to biased hiring decisions

Ethical Analysis

- Violates fairness and equal opportunity
- Can disadvantage qualified candidates

Ethical Version



```python
skills = 7
experience = 3
education = "Masters"

score = skills * 2 + experience * 3

if education == "Masters":
    score += 10

print("Applicant Score:", score)
```

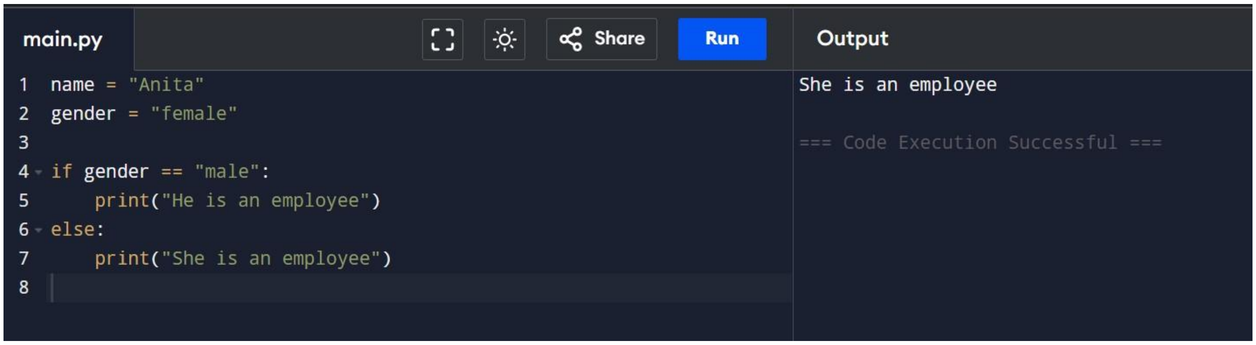
```
Applicant Score: 33

=== Code Execution Successful ===
```

Task 5: Inclusiveness & Ethical Variable Design

Scenario

Inclusive coding avoids gender assumptions.

AI-Generated Code



```python
name = "Anita"
gender = "female"

if gender == "male":
    print("He is an employee")
else:
    print("She is an employee")

```

```
She is an employee

=== Code Execution Successful ===
```

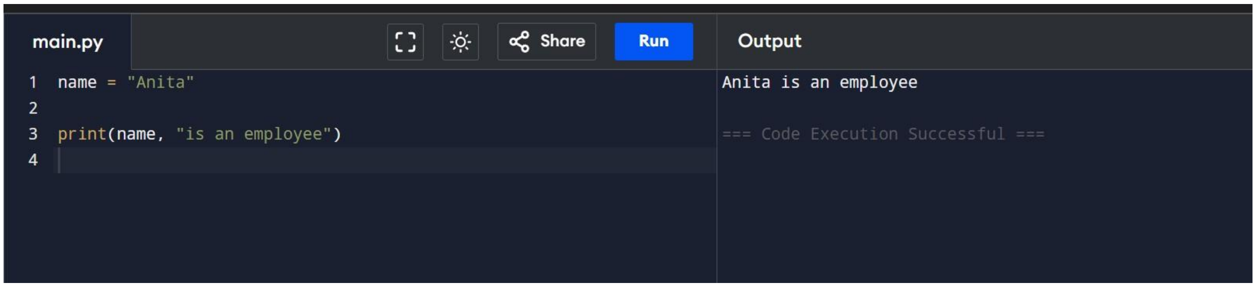
Issues Identified

- Gender-specific language
- Assumes only male/female genders
- Not inclusive or respectful

Inclusive Version

Explanation

- Removed gender dependency
- Used neutral language
- More inclusive and professional



```python
name = "Anita"

print(name, "is an employee")

```

```
Anita is an employee

=== Code Execution Successful ===
```