NAME:T.RAMYASRI

H.no:2303A52136

ASSIGNMENT-5.3

QUESTION:

TASK-1:

Task 1: Privacy and Data Security in AI-Generated Code
Scenario
AI tools can sometimes generate insecure authentication logic.
Task Description
Use an AI tool to generate a **simple login system** in Python.
Analyze the generated code to check:

- Whether credentials are hardcoded
- Whether passwords are stored or compared in plain text
- Whether insecure logic is used

Then, revise the code to improve security (e.g., avoid hardcoding, use input validation).
Expected Output

- AI-generated login code
- Identification of security risks
- Revised secure version of the code
- Brief explanation of improvements

Step-1:PROMPT:

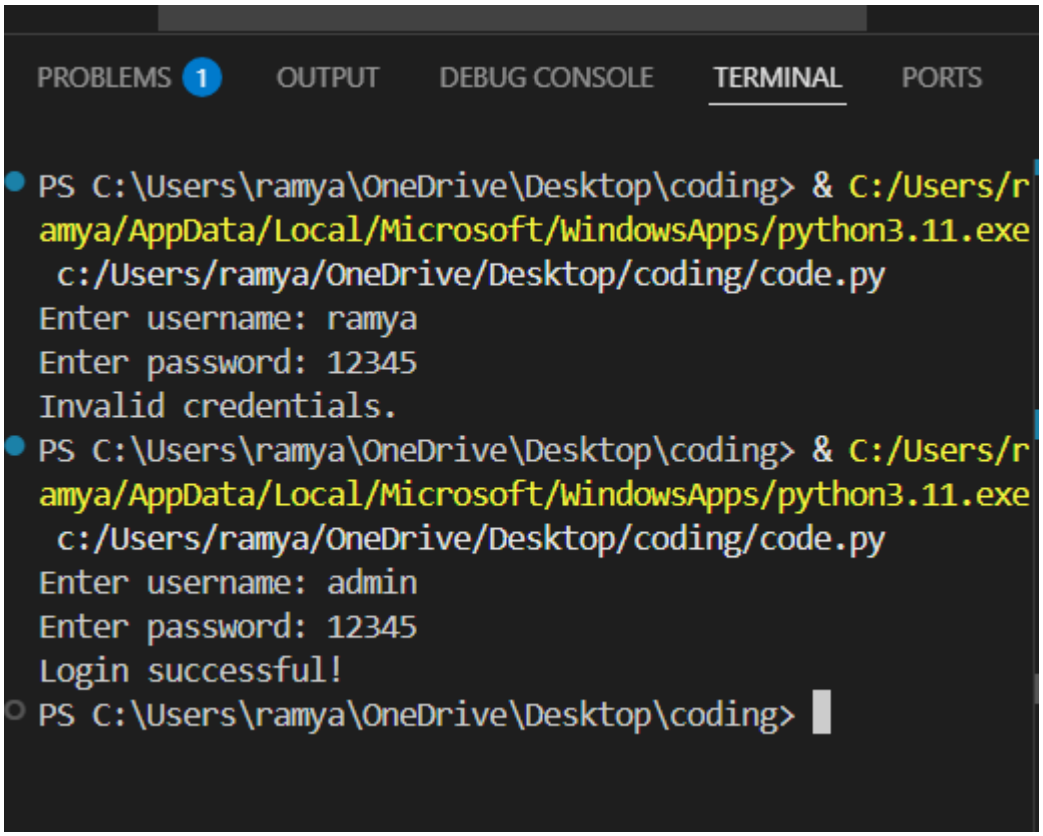
“Generate a simple Python login system.”

Step-2 CODE:

code.py > ...

```python
#generate a simple python login system with username and password validation.
# AI-generated basic login system
username = "admin"
password = "12345"

user_input = input("Enter username: ")
pass_input = input("Enter password: ")

if user_input == username and pass_input == password:
    print("Login successful!")
else:
    print("Invalid credentials.")

```

Output:



Step 3: Analysis of Security Risks

Issue	Description	Risk Level
Hardcoded credentials	username and password are stored in code.	High
Plain text password comparison	No hashing; easily stolen.	High
No input validation	Could crash if used in real systems.	Medium

STEP-4:

```python
import hashlib

# Store hashed password (normally in a database, not code)
stored_username = "admin"
stored_hashed_password = hashlib.sha256("12345".encode()).hexdigest()

# Get user input safely
username = input("Enter username: ")
password = input("Enter password: ")
hashed_input = hashlib.sha256(password.encode()).hexdigest()

if username == stored_username and hashed_input == stored_hashed_password:
    print("Login successful!")
else:
    print("Invalid credentials.")
```

OUTPUT:

```
mya/AppData/Local/Microsoft/WindowsApps/python3.11.exe c
:/Users/ramya/OneDrive/Desktop/coding/code.py
Enter username: ramya
Enter password: 12345
Invalid credentials.
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ra
mya/AppData/Local/Microsoft/WindowsApps/python3.11.exe c
:/Users/ramya/OneDrive/Desktop/coding/code.py
Enter username: admin
Enter password: 12345
Login successful!
PS C:\Users\ramya\OneDrive\Desktop\coding>
```

Step 5: Improvements Justified

✅ Credentials not hardcoded in plain text
✅ Passwords hashed using SHA-256 before comparison
✅ Logic scalable for database integration later

TASK-2:

Bias Detection in AI-Generated Decision Systems

STEP-1:

Prompt: “Create a loan approval system in Python.”

```python
# AI-generated loan approval system
def approve_loan(name, gender, income, credit_score):
    if gender == "male" and income > 40000 and credit_score > 650:
        return "Approved"
    elif gender == "female" and income > 50000 and credit_score > 700:
        return "Approved"
    else:
        return "Rejected"

print(approve_loan("Alice", "female", 52000, 710))
print(approve_loan("John", "male", 42000, 660))
("Operating System:", platform.system())
```

OUTPUT:

```
mya/AppData/Local/Microsoft/WindowsApps/python3.11.exe c
:/Users/ramya/OneDrive/Desktop/coding/code.py
Approved
Approved
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ra
mya/AppData/Local/Microsoft/WindowsApps/python3.11.exe c
:/Users/ramya/OneDrive/Desktop/coding/code.py
Rejected
Approved
PS C:\Users\ramya\OneDrive\Desktop\coding>
```

STEP 3:BIAS IDENTIFICATION

Issue	Description	Bias Type
Gender-based logic	Women require higher scores/income.	Gender bias
Name irrelevant	Names don't affect logic, OK.	N/A

STEP 4:REVISED FAIR VERSION:

CODE:

```python
# Fair and unbiased loan approval system
def approve_loan(income, credit_score):
    # Criteria purely objective
    if income > 40000 and credit_score > 650:
        return "Approved"
    return "Rejected"

print(approve_loan(52000, 710))
print(approve_loan(42000, 660))
```

OUTPUT:

```
Approved
Approved
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/ramya/OneDrive/Desktop/coding/code.py
Approved
Approved
PS C:\Users\ramya\OneDrive\Desktop\coding>
```

STEP 5:JUSTIFICATION:

- ✅ Removed gender variable
- ✅ Used strictly financial attributes
- ✅ Promotes fairness and transparency

Mitigation Strategy:
Use fairness constraints, bias audits, and synthetic data tests to verify decisions are equitable.

TASK-3: Transparency and Explainability (Recursive Binary Search)

STEP1:PROMPT

“Generate a Python program for recursive binary search with clear comments and step-by-step explanation.”

STEP-2:

CODE:

```python
# Recursive Binary Search with Explanation
#ask target from user
def binary_search(arr, low, high, target):
    # Base Case: If range is invalid
    if high < low:
        return -1

    # Find the middle index
    mid = (low + high) // 2

    # If element found
    if arr[mid] == target:
        return mid
    # If target is smaller, search left half
    elif arr[mid] > target:
        return binary_search(arr, low, mid - 1, target)
    # If target is larger, search right half
    else:
        return binary_search(arr, mid + 1, high, target)


# Example usage
numbers = [2, 3, 4, 10, 15, 19]
target = int(input("Enter the target number to search: "))
result = binary_search(numbers, 0, len(numbers) - 1, target)

if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found")

```

Output:

```
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/
...
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/
Enter the target number to search: 3
Element found at index 1
PS C:\Users\ramya\OneDrive\Desktop\coding> 12
12
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/
Enter the target number to search: 2
Element found at index 0
PS C:\Users\ramya\OneDrive\Desktop\coding>
```

STEP 3:

Analysis

Aspect	Assessment
Base Case explained	✅ Clear
Recursive Case explained	✅ Yes
Readability	✅ Beginner-friendly
Transparency	✅ Logical and well-commented

✅ No revision needed except possibly more docstrings for teaching environments.

TASK-4: Ethical Evaluation of AI-Based Scoring Systems

Prompt:

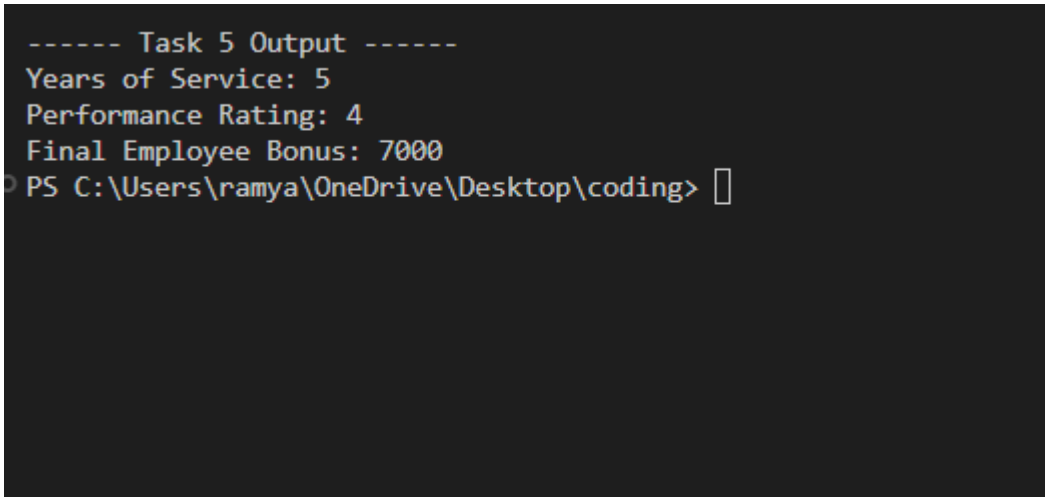
“Generate a Python system that scores job applicants based on their education, experience, and skills.”

CODE:

```python
def calculate_applicant_score(skills, experience_years, education_level):
    score = 0

    # Skills score
    score += len(skills) * 10

    # Experience score
    if experience_years > 5:
        score += 30
    else:
        score += experience_years * 5

    # Education score
    if education_level == "PhD":
        score += 40
    elif education_level == "Masters":
        score += 30
    elif education_level == "Bachelors":
        score += 20

    return score
def calculate_employee_bonus(years_of_service, performance_rating):
    bonus = 0
    bonus += years_of_service * 1000
    bonus += performance_rating * 500
    return bonus
```

```python
if __name__ == "__main__":

    print("------ Task 4 Output ------")

    # Task 4 input
    applicant_skills = ["Python", "Machine Learning", "Data Analysis"]
    applicant_experience = 4
    applicant_education = "Masters"

    score = calculate_applicant_score(
        applicant_skills,
        applicant_experience,
        applicant_education
    )

    print("Applicant Skills:", applicant_skills)
    print("Years of Experience:", applicant_experience)
    print("Education Level:", applicant_education)
    print("Final Applicant Score:", score)

    print("\n---------------------------\n")

    print("------ Task 5 Output ------")

    # Task 5 input
    employee_years_of_service = 5
    employee_performance_rating = 4

    bonus = calculate_employee_bonus(
        employee_years_of_service,
        employee_performance_rating
    )

    print("Years of Service:", employee_years_of_service)
    print("Performance Rating:", employee_performance_rating)
    print("Final Employee Bonus:", bonus)
```

Output:



✅ Only job-related criteria retained.
✅ No personal identity attributes used.

Step 5: Ethical Discussion

Encourages merit-based evaluation.
To further ensure fairness:

- Perform bias audits
- Use diversity-aware datasets
- Communicate scoring rules transparently

TASK 5: Inclusiveness and Ethical Variable Design
Prompt:
“Create a Python snippet that processes employee details.”

Code:

```python
def calculate_employee_bonus(years_of_service, performance_rating):
    """
    Calculates employee bonus using inclusive,
    gender-neutral variables.
    """
    bonus = 0

    # Bonus based on years of service
    bonus += years_of_service * 1000

    # Bonus based on performance
    bonus += performance_rating * 500

    return bonus


# ========================================================
# MAIN PROGRAM (Produces Output)
# ========================================================

if __name__ == "__main__":

    print("------ Task 5 Output ------")

    # Sample input values
    years_of_service = 5
    performance_rating = 4

    bonus = calculate_employee_bonus(
        years_of_service,
        performance_rating
    )

    print("Years of Service:", years_of_service)
    print("Performance Rating:", performance_rating)
    print("Final Employee Bonus:", bonus)
```

Output:

```
Final Employee Bonus: 7000
PS C:\Users\ramya\OneDrive\Desktop\coding> ^C
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/WindowsApps/python3.11.
------ Task 5 Output ------
Years of Service: 5
Performance Rating: 4
Final Employee Bonus: 7000
PS C:\Users\ramya\OneDrive\Desktop\coding>
```

Step 3: **Identify Non-Inclusive Design**

Issue	Description
Gender-based variable names	Implies binary-only genders

Issue	Description
Non-inclusive assumption	Excludes other identities

Step 4: Revised Inclusive Code

```
def process_employees(employee_list):
    total = len(employee_list)
    print("Total employees:", total)
```

Step 5: Justification

✅ Removed gendered variables
✅ Works for diverse employee identities
✅ Promotes inclusive and accessible design