AI ASSISTED CODING

ASSIGNMENT-5

B.Sri Laxmi Gayathri

2303a52033

Batch:38

Lab 5: Ethical Foundations – Responsible AI Coding Practices

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

Prompt:

Generate a simple Python login system that checks a username and password and prints whether login is successful.

Comments:

- Password is hashed, not stored or compared in plain text.

- Input is validated using .strip() to avoid unwanted spaces.
- Logic is safer and closer to real-world authentication systems.

Explanation:

The revised code improves security by using password hashing instead of plain text comparison. Hashing ensures that even if someone accesses the code, they cannot easily see the original password. Input validation removes extra spaces that could cause login issues. This approach reduces the risk of data exposure and makes the authentication process more secure and reliable.

Code & Output:

```
import hashlib
print("--- Secure Login System ---")
# Correct stored credentials
stored_username = "admin"
stored_password = "admin123"
# Create hash once
stored_password_hash = hashlib.sha256(stored_password.encode()).hexdigest()
failed_attempts = 0
MAX_ATTEMPTS = 3
while failed_attempts < MAX_ATTEMPTS:
    username = input("Enter username: ").strip()
    password = input("Enter password: ").strip()
    password_hash = hashlib.sha256(password.encode()).hexdigest()
    if username == stored_username and password_hash == stored_password_hash:
        print("Login successful.")
        break
    else:
        failed_attempts += 1
        print("Login failed. Invalid username or password.")
        print(f"Failed attempts: {failed_attempts}/{MAX_ATTEMPTS}")
if failed_attempts == MAX_ATTEMPTS:
    print("Account locked. Too many failed attempts.")
```

```
--- Secure Login System ---
Enter username: admin
Enter password: admin123
Login successful.
```

Task 2: Bias Detection in AI-Generated Decision Systems

Scenario

AI systems may unintentionally introduce bias.

Task Description

Use AI prompts such as:

• “Create a loan approval system”

• Vary applicant names and genders in prompts

Analyze whether:

• The logic treats certain genders or names unfairly

• Approval decisions depend on irrelevant personal attributes

Suggest methods to reduce or remove bias.

Expected Output

• Python code generated by AI

• Identification of biased logic (if any)

• Discussion on fairness issues

• Mitigation strategies

Prompt:

Create a simple Python loan approval system that decides whether a loan is approved based on applicant details.

Comments:

- Gender and name are **not used** in decision making.
- Approval depends only on **income and credit score.**
- Logic is fair and consistent for all applicants.

Explanation:

The initial loan approval system is biased because it uses gender as a condition for approving loans. This is unfair since gender does not affect a person's ability to repay a loan. Such logic can lead to discrimination and unequal treatment of applicants. The revised system removes personal attributes like gender and name. It makes decisions only based on income and credit score, ensuring fairness for all applicants.

Code & Output:

```python
# Loan approval system (AI-generated - biased)
name = input("Enter applicant name: ")
gender = input("Enter gender (male/female): ")
income = int(input("Enter monthly income: "))
if gender == "male" and income >= 30000:
    print("Loan Approved")
elif gender == "female" and income >= 40000:
    print("Loan Approved")
else:
    print("Loan Rejected")
```

```
Enter applicant name: gayathri
Enter gender (male/female): female
Enter monthly income: 50000
Loan Approved
```

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

Scenario

AI-generated code should be transparent, well-documented, and easy for humans to understand and verify.

Task Description

Use an AI tool to generate a Python program that:

• Implements Binary Search using recursion

• Searches for a given element in a sorted list

• Includes:

o Clear inline comments

o A step-by-step explanation of the recursive logic

After generating the code, analyze:

• Whether the explanation clearly describes the base case and recursive case

• Whether the comments correctly match the code logic

• Whether the code is understandable for beginner-level students

Expected Output

• Python program for recursive binary search

• AI-generated comments and explanation

• Student's assessment on clarity, correctness, and transparency

Prompt:

Write a Python program to implement binary search using recursion. The code should search for an element in a sorted list and include clear comments and explanation of the recursive logic.

Comments:

• The code clearly explains the base case, where the search stops if the element is not found.
• Comments describe how the function recursively searches the left or right half of the list.
• Each step is well commented, making the logic easy for beginners to understand.

Explanation:

Binary search works by repeatedly dividing the search space into two halves. First, the middle element of the list is checked. If the middle element matches the target, the search ends. If the target is smaller, the function calls itself on the left half. If the target is larger, it calls itself on the right half. The recursion stops when the element is found or when the search range becomes invalid.

Code & Output:

```python
# Recursive Binary Search Function
def binary_search(arr, low, high, target):
    # Base case: element not found
    if low > high:
        return -1
    # Find the middle index
    mid = (low + high) // 2
    # If target is found at middle
    if arr[mid] == target:
        return mid
    # If target is smaller, search left half
    elif target < arr[mid]:
        return binary_search(arr, low, mid - 1, target)
    # If target is larger, search right half
    else:
        return binary_search(arr, mid + 1, high, target)
# Sorted list
numbers = [10, 20, 30, 40, 50, 60, 70]
key = 40
# Function call
result = binary_search(numbers, 0, len(numbers) - 1, key)
if result != -1:
    print("Element found at index:", result)
else:
    print("Element not found")
```

```
Element found at index: 3
```

Task 4: Ethical Evaluation of AI-Based Scoring Systems

Scenario

AI-generated scoring systems can influence hiring decisions.

Task Description

Ask an AI tool to generate a job applicant scoring system based on features such as:

- Skills
- Experience
- Education

Analyze the generated code to check:

- Whether gender, name, or unrelated features influence scoring
- Whether the logic is fair and objective

Expected Output

• Python scoring system code

• Identification of potential bias (if any)

• Ethical analysis of the scoring logic

Prompt:

Create a Python program that scores job applicants based on skills, experience, education, and gender, and prints whether the applicant is selected.

Comments:

• Scoring is based only on skills, experience, and education.
• Gender and name do not affect the decision.
• Logic is fair, objective, and job-related.

Explanation:

The scoring system is ethically problematic because it uses gender as a factor in hiring decisions. Gender does not reflect a person's skills or job capability, so including it leads to unfair advantage. This can cause discrimination and unequal opportunities. AI systems used in hiring must be objective and transparent. Decisions should be based only on job-related qualifications.

Code & Output:

```
def job_applicant_scoring_biased(name, gender, skills, experience, education):
    score = 0
    if skills >= 7:
        score += 30
    if experience >= 3:
        score += 30
    if education.lower() == "pg":
        score += 20
    if gender.lower() == "male":
        score += 20
    print(f"--- Applicant Details for {name} ---")
    print(f"Gender: {gender}")
    print(f"Skills Score: {skills}")
    print(f"Years of Experience: {experience}")
    print(f"Education: {education}")
    print(f"Total Score: {score}")
    if score >= 70:
        print("Applicant Selected")
    else:
        print("Applicant Rejected")
    print("\n")
```

```
job_applicant_scoring_biased("Alice Smith", "Female", 8, 4, "PG")
job_applicant_scoring_biased("Bob Johnson", "Male", 6, 2, "UG")
job_applicant_scoring_biased("Charlie Brown", "Male", 7, 3, "UG")
job_applicant_scoring_biased("Diana Prince", "Female", 7, 3, "UG")
```

```
--- Applicant Details for Alice Smith ---
Gender: Female
Skills Score: 8
Years of Experience: 4
Education: PG
Total Score: 80
Applicant Selected


--- Applicant Details for Bob Johnson ---
Gender: Male
Skills Score: 6
Years of Experience: 2
Education: UG
Total Score: 20
Applicant Rejected


--- Applicant Details for Charlie Brown ---
Gender: Male
Skills Score: 7
Years of Experience: 3
Education: UG
Total Score: 80
Applicant Selected


--- Applicant Details for Diana Prince ---
Gender: Female
Skills Score: 7
Years of Experience: 3
Education: UG
Total Score: 60
Applicant Rejected
```

Task 5: Inclusiveness and Ethical Variable Design

Scenario

Inclusive coding practices avoid assumptions related to gender, identity, or roles and promote fairness in software design.

Task Description

Use an AI tool to generate a Python code snippet that processes user or employee details.

Analyze the code to identify:

- Gender-specific variables (e.g., male, female)
- Assumptions based on gender or identity
- Non-inclusive naming or logic

Modify or regenerate the code to:

• Use gender-neutral variable names

• Avoid gender-based conditions unless strictly required

• Ensure inclusive and respectful coding practices

Expected Output

• Original AI-generated code snippet

• Revised inclusive and gender-neutral code

• Brief explanation of:

o What was non-inclusive

o How inclusiveness was improved

Prompt:

Generate a Python program that processes employee details and gives a bonus based on gender and role.

Comments:

• The revised code uses gender-neutral variable names to avoid assumptions about identity.
• Bonus calculation is based on performance rating, not gender.
• The logic is inclusive, fair, and respectful to all users.

Explanation:

The original code was non-inclusive because it used gender-based conditions to calculate bonuses. Gender is not related to work performance, so this logic was unfair and exclusionary. The revised code removes gender variables and uses performance rating instead. Gender-neutral variable names are used to ensure respect and inclusiveness. This makes the code fair and suitable for all users.

Code & Output:

```python
def process_employee_details_inclusive(employee_name, department, performance_score):
    bonus_percentage = 0.0
    promotion_eligibility = "Not Eligible"
    feedback = []
    if performance_score >= 80:
        bonus_percentage = 0.12
        feedback.append("Received high performance bonus.")
    elif performance_score >= 60:
        bonus_percentage = 0.07
        feedback.append("Received moderate performance bonus.")
    else:
        feedback.append("Did not meet performance criteria for bonus.")
    if department.lower() == "marketing":
        if performance_score >= 80:
            promotion_eligibility = "Eligible for Promotion"
            feedback.append("Eligible for promotion in Marketing based on performance.")
        else:
            promotion_eligibility = "Not Eligible"
            feedback.append("Not eligible for promotion in Marketing based on performance.")
    elif department.lower() == "engineering":
        if performance_score >= 85:
            promotion_eligibility = "Eligible for Promotion"
            feedback.append("Eligible for promotion in Engineering based on performance.")
        else:
            promotion_eligibility = "Not Eligible"
            feedback.append("Not eligible for promotion in Engineering based on performance.")
    else:
        promotion_eligibility = "Not Applicable to Department"
    print(f"\n--- Employee Details for {employee_name} ---")
    print(f"Department: {department}")
    print(f"Performance Score: {performance_score}")
    print(f"Calculated Bonus Percentage: {bonus_percentage*100:.0f}%")
    print(f"Promotion Eligibility: {promotion_eligibility}")
    print(f"System Feedback: {' | '.join(feedback)}")
```

```python
process_employee_details_inclusive("Alice Smith", "Marketing", 85)
process_employee_details_inclusive("Bob Johnson", "Marketing", 80)
process_employee_details_inclusive("Charlie Brown", "Engineering", 90)
process_employee_details_inclusive("Diana Prince", "Engineering", 75)
process_employee_details_inclusive("Eve Adams", "Marketing", 70)
process_employee_details_inclusive("Frank White", "Engineering", 70)
```

```
--- Employee Details for Alice Smith ---
Department: Marketing
Performance Score: 85
Calculated Bonus Percentage: 12%
Promotion Eligibility: Eligible for Promotion
System Feedback: Received high performance bonus. | Eligible for promotion in Marketing based on performance.

--- Employee Details for Bob Johnson ---
Department: Marketing
Performance Score: 80
Calculated Bonus Percentage: 12%
Promotion Eligibility: Eligible for Promotion
System Feedback: Received high performance bonus. | Eligible for promotion in Marketing based on performance.

--- Employee Details for Charlie Brown ---
Department: Engineering
Performance Score: 90
Calculated Bonus Percentage: 12%
Promotion Eligibility: Eligible for Promotion
System Feedback: Received high performance bonus. | Eligible for promotion in Engineering based on performance.

--- Employee Details for Diana Prince ---
Department: Engineering
Performance Score: 75
Calculated Bonus Percentage: 7%
Promotion Eligibility: Not Eligible
System Feedback: Received moderate performance bonus. | Not eligible for promotion in Engineering based on performance.

--- Employee Details for Eve Adams ---
Department: Marketing
Performance Score: 70
Calculated Bonus Percentage: 7%
Promotion Eligibility: Not Eligible
System Feedback: Received moderate performance bonus. | Not eligible for promotion in Marketing based on performance.

--- Employee Details for Frank White ---
Department: Engineering
Performance Score: 70
Calculated Bonus Percentage: 7%
Promotion Eligibility: Not Eligible
System Feedback: Received moderate performance bonus. | Not eligible for promotion in Engineering based on performance.
```