AI ASSISTED CODING WEEK-5.3

2303A51642

B-28

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

Scenario: AI tools can sometimes generate insecure authentication logic.

AI-Generated Login Code (Insecure Example):

This simulates a typical first-pass output from an AI when asked for a "simple login system" without security constraints.

```
username = "admin"
password = "admin123"

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

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

Security Risk Analysis:

Hardcoded credentials: Username and password are written directly in the code, making them easy to expose.

Plain text passwords: Passwords are stored and compared without encryption, which is insecure.

No hashing: Passwords are not hashed, violating basic security practices.

No input validation: User input is accepted without checks, increasing misuse risk.

Not scalable: The design cannot securely handle multiple users.

Revised Secure Version of the Code:

```
import hashlib
users = {
  "admin": "240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9"
```

```
}
username = input("Enter username: ").strip()
password = input("Enter password: ").strip()

hashed_password = hashlib.sha256(password.encode()).hexdigest()

if username in users and users[username] == hashed_password:
    print("Login successful")
else:
    print("Invalid credentials")
```

Brief Explanation of Improvements:

- **Removed hardcoded plain text passwords**
  - Passwords are stored as SHA-256 hashes instead of readable text.
- **Used hashing for password comparison**
  - Even if data is exposed, original passwords cannot be easily recovered.
- **Basic input validation**
  - .strip() avoids accidental whitespace issues.
- **More realistic structure**
  - User credentials are stored in a data structure that can later be replaced with a database.

Task 2: Bias Detection in AI-Generated Decision Systems

Scenario: AI systems may unintentionally introduce bias.

AI-Generated Loan Approval Code (Biased Example):

```
name = input("Enter applicant name: ")
gender = input("Enter gender: ")
income = int(input("Enter annual income: "))
credit_score = int(input("Enter credit score: "))

if gender == "male" and income > 300000 and credit_score > 650:
    print("Loan Approved")
elif gender == "female" and income > 400000 and credit_score > 700:
    print("Loan Approved")
else:
    print("Loan Rejected")
```

Identification of Biased Logic:

- **Gender-based conditions**
    - Different income and credit score thresholds are applied based on gender.
- **Irrelevant personal attributes**
    - Gender is used in decision-making, even though it is not a valid financial factor.

Discussion on Fairness Issues:

This system violates fairness principles by:

- Treating applicants differently based on gender.
- Allowing non-financial attributes to influence approval decisions.
- Reinforcing stereotypes through unequal thresholds.
- Reducing trust and transparency in automated decision systems.

Such bias can lead to discrimination and may violate ethical and legal standards in real-world financial systems.

Mitigation Strategies:

- **Remove sensitive attributes**
    - Exclude gender, name, race, or religion from decision logic.
- **Use objective criteria only**
    - Base decisions on income, credit score, employment history, etc.
- **Apply uniform thresholds**
    - Ensure the same approval rules apply to all applicants.
- **Regular bias audits**
    - Test outcomes across diverse demographic groups.
- **Explainable logic**
    - Make decision rules transparent and reviewable.

Revised Fair Loan Approval Code (Bias-Reduced):

```
income = int(input("Enter annual income: "))
credit_score = int(input("Enter credit score: "))

if income > 350000 and credit_score > 680:
  print("Loan Approved")
else:
  print("Loan Rejected")
```

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.

Python Program: Recursive Binary Search (AI-Generated):

```
def binary_search(arr, low, high, target):
  # Base case: if the search space is invalid, element is not present
  if low > high:
    return -1
  # Find the middle index
  mid = (low + high) // 2
  # If the middle element is the target, return its index
  if arr[mid] == target:
    return mid
  # If target is smaller than middle element, search left sub-array
  elif target < arr[mid]:
    return binary_search(arr, low, mid - 1, target)

  # If target is greater than middle element, search right sub-array
  else:
    return binary_search(arr, mid + 1, high, target)

# Example usage
numbers = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
key = 23
result = binary_search(numbers, 0, len(numbers) - 1, key)

if result != -1:
  print("Element found at index", result)
else:
  print("Element not found")
```

AI-Generated Explanation of Recursive Logic:

Binary search works by repeatedly dividing the search range in half.
The **base case** occurs when the starting index becomes greater than the ending index, meaning the element is not present.
In each recursive step, the middle element is checked. If it matches the target, the search stops. If the target is smaller, the function recursively searches the left half; if larger, it searches the right half. This continues until the element is found or the base case is reached.

Student's Assessment on Clarity, Correctness, and Transparency:

The explanation clearly identifies both the base case and the recursive case, making the logic easy to follow. The inline comments accurately describe what each part of the code is doing and directly match the implementation. Overall, the code is well-structured, readable, and understandable for beginner-level students, making it transparent and easy to verify.

Task 4: Ethical Evaluation of AI-Based Scoring Systems

Scenario: AI-generated scoring systems can influence hiring decisions.

AI-Generated Job Applicant Scoring Code:

```
name = input("Enter name: ")
gender = input("Enter gender: ")
skills = int(input("Enter skills score (0-10): "))
experience = int(input("Enter years of experience: "))
education = input("Enter education level: ")

score = 0

if skills >= 7:
    score += 40
if experience >= 3:
    score += 30
if education.lower() == "masters":
    score += 20
if gender.lower() == "male":
    score += 10
```

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

Identification of Potential Bias:

- Gender directly affects the final score, giving preference to male applicants.
- Name and gender are collected even though they are unrelated to job performance.
- This introduces unfair advantage based on personal attributes rather than merit.

Ethical Analysis of the Scoring Logic:

The scoring system is ethically flawed because it allows non-job-related factors like gender to influence hiring outcomes. This violates fairness and equal opportunity principles, can reinforce discrimination, and reduces trust in automated hiring systems. Ethical AI systems should base decisions strictly on relevant qualifications such as skills, experience, and education, ensuring transparency and equal treatment for all applicants.

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.

Original AI-Generated Code Snippet (Non-Inclusive):

```
name = input("Enter employee name: ")
gender = input("Enter gender (male/female): ")
role = input("Enter job role: ")

if gender == "male":
  print(name, "is assigned to night shift")
else:
  print(name, "is assigned to day shift")
```

Issues Identified (Non-Inclusive Aspects):

- Uses **gender-specific variables** like male and female.
- Makes **assumptions based on gender**, assigning work shifts accordingly.
- Gender is used even though it is **irrelevant to job role or capability**.

Revised Inclusive and Gender-Neutral Code:

```
employee_name = input("Enter employee name: ")
job_role = input("Enter job role: ")
preferred_shift = input("Enter preferred shift (day/night): ")

print(employee_name, "is assigned to", preferred_shift, "shift")
```

Brief Explanation:

The original code was non-inclusive because it used gender-based variables and logic to make decisions unrelated to job requirements. The revised version removes gender entirely, uses neutral variable names, and bases decisions on user preference or role-related inputs, promoting fairness, inclusiveness, and respectful coding practices.