# AI-Assisted-Coding Week-5.3

**2303A51878**

**B-28**

## <u>Task 1: Privacy and Data Security in AI-Generated Code</u>

**<u>Scenario:-</u>** AI tools may generate authentication code that is functionally correct but insecure. This task demonstrates how such code can introduce privacy and data-security risks and how to improve it.

### 1) AI-Generated Login Code (Insecure):-

```
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"
```

### Output:-

**Case 1:** Correct credentials

```
Enter username: admin
Enter password: admin123
Login successful
```

**Case 2:** Wrong credentials

```
Enter username: admin
Enter password: 1234
Invalid credentials
```

## 2) Security Risks Identified

**Hardcoded Credentials**

- Username and password are directly written in the source code.
- Anyone with access to the file can read them.

**Plain Text Password Storage and Comparison**

- Passwords are stored and compared in plain text.
- If leaked, passwords can be immediately exploited.

**Insecure Authentication Logic**

- No hashing or encryption is used.
- No validation for empty or invalid input.
- Vulnerable to brute-force and credential-theft attacks.

## 3) Revised Secure Login Code

```
import hashlib
users = {
    "admin": hashlib.sha256("admin123".encode()).hexdigest()
}

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

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

```
if not username or not password:
    print("Username and password cannot be empty")
else:
    hashed_password = hash_password(password)
    if username in users and users[username] == hashed_password:
        print("Login successful")
    else:
        print("Invalid credentials")
```

## Output

**Case 1:** Correct credentials

```
Enter username: admin
Enter password: admin123
Login successful
```

**Case 2:** Empty input

```
Enter username:
Enter password:
Username and password cannot be empty
```

**Case 3:** Wrong password

```
Enter username: admin
Enter password: 1234
Invalid credentials
```

## 4) Explanation of Security Improvements

**Removed hardcoded plain-text passwords**: Passwords are stored as hashes.

**Used password hashing (SHA-256)**: Protects passwords even if data is exposed.

**Added input validation**: Prevents empty or invalid inputs.

**Improved authentication logic**: Compares hashed values instead of raw passwords.

## Task 2: Bias Detection in AI-Generated Decision Systems

**Scenario :-** AI systems can unintentionally introduce bias when decision logic includes irrelevant personal attributes such as **name** or **gender**. This task analyzes such bias in an AI-generated loan approval system.

### 1) AI-Generated Loan Approval Code (Biased Example)

```
name = input("Enter applicant name: ")
gender = input("Enter gender (Male/Female): ")
income = int(input("Enter monthly income: "))
credit_score = int(input("Enter credit score: "))

if gender == "Male" and income > 30000 and credit_score > 650:
    print("Loan Approved")
elif gender == "Female" and income > 40000 and credit_score > 700:
    print("Loan Approved")
else:
    print("Loan Rejected")
```

### Output

**Case 1: Male applicant**

```
Enter applicant name: Rahul
Enter gender (Male/Female): Male
Enter monthly income: 35000
Enter credit score: 680
Loan Approved
```

**Case 2: Female applicant (same details)**

```
Enter applicant name: Ananya
Enter gender (Male/Female): Female
Enter monthly income: 35000
Enter credit score: 680
Loan Rejected
```

## 2) Identification of Biased Logic

**Gender-Based Discrimination**

- Male applicants require **lower income and credit score**
- Female applicants face **stricter criteria**

**Irrelevant Attribute Usage**

- **Gender** is used for loan approval
- **Name** is collected but has no valid financial purpose

**Unfair Decision Rules**

- Approval criteria change based on gender
- Violates principles of fairness and equality

## 3) Fairness Issues Discussion

- Loan approval should be based on **financial eligibility**, not personal identity.
- Using gender introduces **systemic bias** and discrimination.
- Such systems may violate **ethical AI principles** and **anti-discrimination laws**.
- Biased systems reduce trust and transparency in AI decisions.

## 4) Revised Bias-Free Loan Approval Code

**Improved Python Code**

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

if income >= 35000 and credit_score >= 680:
    print("Loan Approved")
else:
    print("Loan Rejected")
```

## Output

**Case 1: Eligible applicant**

```
Enter monthly income: 40000
Enter credit score: 700
Loan Approved
```

**Case 2: Not eligible**

```
Enter monthly income: 30000
Enter credit score: 650
Loan Rejected
```

## 5) Mitigation Strategies to Reduce Bias

**Remove Sensitive Attributes**

- Do not use gender, name, race, or religion in decision logic

**Use Objective Criteria**

- Base decisions on income, credit score, employment history

**Fairness Testing**

- Test outcomes across different demographic groups

**Transparency**

- Clearly explain approval criteria to users

**Human Oversight**

- Allow manual review for borderline cases

## 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—especially beginners—to understand and verify. This task evaluates the clarity and explainability of AI-generated recursive code.

### 1) Python Program for Recursive Binary Search (AI-Generated)

**Python Code with Inline Comments**

```
def binary_search(arr, low, high, target):

    if low > high:

        return -1

    mid = (low + high) // 2

    if arr[mid] == target:

        return mid

    elif target < arr[mid]:

        return binary_search(arr, low, mid - 1, target)

    else:

        return binary_search(arr, mid + 1, high, target)

numbers = [2, 5, 8, 12, 16, 23, 38, 56]

target = 23

result = binary_search(numbers, 0, len(numbers) - 1, target)

if result != -1:

    print("Element found at index:", result)

else:

    print("Element not found")
```

## OUTPUT:-

**Case 1: Element Found**

Input (in code)

numbers = [2, 5, 8, 12, 16, 23, 38, 56]

target = 23

Output

Element found at index: 5

**Case 2: Element Not Found**

Input (in code)

numbers = [2, 5, 8, 12, 16, 23, 38, 56]

target = 15

Output

Element not found

## 2) AI-Generated Explanation of Recursive Logic

**Step-by-Step Explanation**

1. **Base Case**
   - If low > high, it means the element is not present in the list.
   - The function returns -1 to indicate failure.
2. **Middle Element Check**
   - The middle index of the list is calculated using (low + high) // 2.
3. **Recursive Case – Left Subarray**
   - If the target value is smaller than the middle element, the function recursively searches the **left half** of the list.
4. **Recursive Case – Right Subarray**
   - If the target value is greater than the middle element, the function recursively searches the **right half** of the list.
5. **Termination**
   - The recursion stops when the element is found or when the search range becomes invalid.

## 3) Student’s Assessment

**Clarity**

- The explanation clearly distinguishes between **base case** and **recursive case**.
- Each step of the recursion is explained in simple terms.

**Correctness**

- The binary search logic is correct and follows standard algorithm rules.
- Base case and recursive calls are implemented properly.

**Transparency**

- Inline comments match the actual code logic.
- Variable names (low, high, mid) are meaningful.
- Easy to trace how recursion works step by step.

**Beginner-Friendliness**

- Code is simple and readable.
- Comments help beginners understand why each step is needed.
- Output is clear and easy to interpret.

## Task 4: Ethical Evaluation of AI-Based Scoring Systems

**Scenario :-**AI-based scoring systems can significantly influence hiring decisions. If such systems include irrelevant personal attributes or biased logic, they may lead to unfair or unethical outcomes. This task evaluates an AI-generated job applicant scoring system for fairness and objectivity.

## 1) Python Scoring System Code (AI-Generated Example)

```
name = input("Enter applicant name: ")

gender = input("Enter gender (Male/Female): ")

skills = int(input("Enter skills score (out of 10): "))

experience = int(input("Enter years of experience: "))

education = input("Enter highest education (Bachelors/Masters/PhD): ")
```

```
score = 0
if skills >= 7:
    score += 40
if experience >= 3:
    score += 30
if education == "Masters":
    score += 20
elif education == "PhD":
    score += 30
if gender == "Male":
    score += 10
print("Final Applicant Score:", score)
```

## OUTPUTS:-

**Case 1: Male Applicant**

**Input**

```
Enter applicant name: Rahul
Enter gender (Male/Female): Male
Enter skills score (out of 10): 8
Enter years of experience: 4
Enter highest education (Bachelors/Masters/PhD): Masters
```

**Output**

```
Final Applicant Score: 100
```

**Case 2: Female Applicant (Same Qualifications)**

**Input**

Enter applicant name: Ananya

Enter gender (Male/Female): Female

Enter skills score (out of 10): 8

Enter years of experience: 4

Enter highest education (Bachelors/Masters/PhD): Masters

**Output**

Final Applicant Score: 90

**Case 3: Lower Qualifications**

**Input**

Enter applicant name: Kiran

Enter gender (Male/Female): Female

Enter skills score (out of 10): 6

Enter years of experience: 2

Enter highest education (Bachelors/Masters/PhD): Bachelors

**Output**

Final Applicant Score: 0

## 2) Identification of Potential Bias

- **Gender Bias**
    - Male applicants receive extra points regardless of qualifications.
- **Irrelevant Personal Attribute**
    - Gender is unrelated to job performance but still affects scoring.
- **Name Collected but Not Needed**
    - Name has no role in scoring and could encourage identity-based bias.

## 3) Ethical Analysis of the Scoring Logic

**Fairness**

- The scoring system is unfair because it favors one gender over another.
- Equal qualifications do not guarantee equal scores.

**Objectivity**

- Skills, experience, and education are valid evaluation criteria.
- Including gender introduces subjectivity and discrimination.

**Transparency**

- The logic is visible and understandable, but ethically flawed.
- Transparent bias is still unethical.

**Ethical Concerns**

- Violates principles of **equal opportunity** and **ethical AI usage**.
- May lead to discriminatory hiring decisions.
- Can reduce trust in AI-assisted recruitment systems.

## Task 5: Inclusiveness and Ethical Variable Design

**Scenario :-** Inclusive coding practices avoid assumptions related to gender, identity, or roles and promote fairness and respect in software design. This task evaluates AI-generated code for inclusiveness and improves it using gender-neutral design principles.

## 1) Original AI-Generated Code Snippet (Non-Inclusive)

```
name = input("Enter employee name: ")

gender = input("Enter gender (Male/Female): ")

salary = int(input("Enter salary: "))

if gender == "Male":

    bonus = salary * 0.10

elif gender == "Female":
    bonus = salary * 0.15
else:
    bonus = 0
print("Employee Name:", name)
print("Bonus Amount:", bonus)
```

## OUTPUTS:-

**Case 1: Male Employee**

Input

```
Enter employee name: Rahul
Enter gender (Male/Female): Male
Enter salary: 50000
```

Output

```
Employee Name: Rahul
Bonus Amount: 5000.0
```

**Case 2: Female Employee (Same Salary)**

Input

```
Enter employee name: Ananya
Enter gender (Male/Female): Female
Enter salary: 50000
```

Output

```
Employee Name: Ananya
Bonus Amount: 7500.0
```

**Case 3: Other / Unspecified Gender**

Input

```
Enter employee name: Kiran

Enter gender (Male/Female): Other

Enter salary: 50000
```

Output

```
Employee Name: Kiran

Bonus Amount: 0
```

## 2) Issues Identified in the Code

- Uses a gender-specific variable (gender).
- Applies different logic based on gender, which is unrelated to job performance.
- Assumes only binary gender options (Male/Female).
- Can lead to unfair and non-inclusive outcomes.

## 3) Revised Inclusive and Gender-Neutral Code

```
name = input("Enter employee name: ")

role = input("Enter employee role: ")

salary = int(input("Enter salary: "))

if role.lower() == "manager":

    bonus = salary * 0.15

else:

    bonus = salary * 0.10

print("Employee Name:", name)

print("Bonus Amount:", bonus)
```

## OUTPUTS:-

**Case 1: Manager Role**

Input

```
Enter employee name: Rahul

Enter employee role: Manager

Enter salary: 50000
```

Output

```
Employee Name: Rahul

Bonus Amount: 7500.0
```

**Case 2: Non-Manager Role**

Input

```
Enter employee name: Ananya

Enter employee role: Developer

Enter salary: 50000
```

Output

```
Employee Name: Ananya

Bonus Amount: 5000.0
```

**Case 3: Another Role**

Input

```
Enter employee name: Kiran

Enter employee role: Analyst

Enter salary: 40000
```

Output

```
Employee Name: Kiran

Bonus Amount: 4000.0
```

## 4) Brief Explanation

### What Was Non-Inclusive

- The original code used **gender as a decision factor**, even though gender is irrelevant to bonus calculation.
- It made assumptions about identity and treated employees differently based on gender.

### How Inclusiveness Was Improved

- Removed gender-based variables and conditions.
- Introduced **gender-neutral variables** such as role.
- Used **job-related attributes** to determine bonuses.
- Ensured equal and respectful treatment for all users.