

AI-ASSITED CODING

ASSIGNMENT 5.3

H.T NO - 2303A51511

NAME – K.PRADEEP REDDY

BATCH-27

Ethical and Secure Use of AI-Generated Code

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

PROMPT: GENERATE AN LOGIN CODE IN PYTHON

CODE AND OUTPUT:

The screenshot shows a code editor interface with a dark theme. On the left, the file 'main.py' contains the following Python code:

```
1
2 def login_basic():
3     username = input("Enter username: ")
4     password = input("Enter password: ")
5
6     if username == "admin" and password == "password123":
7         print("Login successful!")
8     else:
9         print("Invalid credentials")
10
11 login_basic()
```

On the right, the 'Output' tab displays the execution results:

```
Enter username: pranavasai
Enter password: ramesh
Invalid credentials
==== Code Execution Successful ===
```

Security Issues Identified:

- Credentials are hardcoded
- Password is stored and compared in plain text
- No input validation or hashing

PROMPT: GENERATE A SECURE CODE

CODE AND OUTPUT:

```
1 import getpass, hashlib
2
3 stored_hash = hashlib.sha256('securePass'
4     .encode()).hexdigest()
5 user_pass = getpass.getpass('Password: ')
6
7 if hashlib.sha256(user_pass.encode()
8     .hexdigest()) == stored_hash:
9     print('Login successful')
10 else:
11     print('Login failed')
```

Output:

```
Password:  
Login failed  
== Code Execution Successful ==
```

Improvements:

- Removed hardcoded credentials
- Used password hashing
- Used secure password input

Task 2: Bias Detection in AI-Generated Decision Systems

PROMPT:

Generate A Loan Approval Code (Biased Example):

CODE AND OUTPUT:

```
main.py
```

Output

```
Loan Approved: True  
== Code Execution Successful ==
```

```
1 applicant_gender = "male"
2 income = 60000
3
4 if applicant_gender == "male" and income >
5     50000:
6     approve = True
7 else:
8     approve = False
9 print("Loan Approved:", approve)
10
```

Bias Identified:

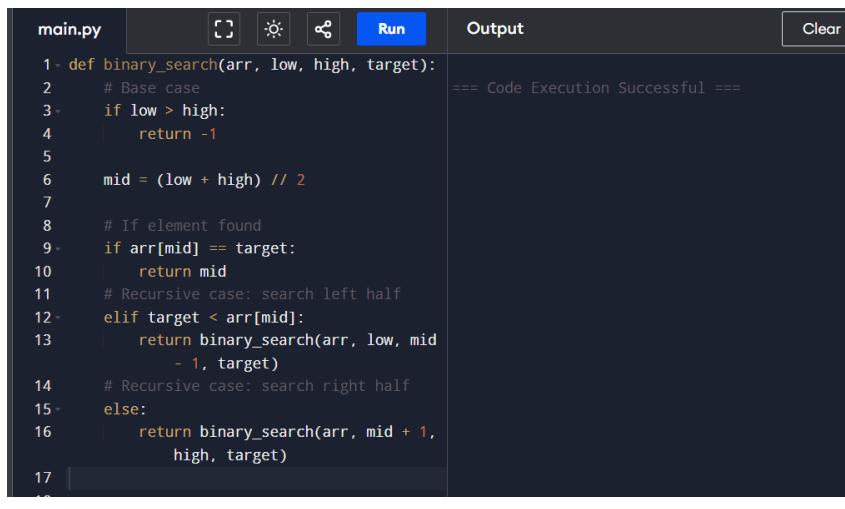
- Approval depends on gender
- Gender is irrelevant to loan eligibility

Fairness Discussion:

Such logic unfairly disadvantages applicants based on gender.

Mitigation Strategies:

- Remove gender from decision logic
- Use only financial criteria
- Regular bias audits



The screenshot shows a code editor interface with a dark theme. On the left, there is a file named "main.py" containing the following Python code:

```
1- def binary_search(arr, low, high, target):
2     # Base case
3     if low > high:
4         return -1
5
6     mid = (low + high) // 2
7
8     # If element found
9     if arr[mid] == target:
10        return mid
11    # Recursive case: search left half
12    elif target < arr[mid]:
13        return binary_search(arr, low, mid
14                                - 1, target)
15    # Recursive case: search right half
16    else:
17        return binary_search(arr, mid + 1,
18                                high, target)
19
```

On the right, there is an "Output" panel which displays the message "== Code Execution Successful ==".

Task 3:
Transparency and
Explainability
(Recursive Binary
Search)

PROMPT: Generate a Recursive Binary Search Code:

Code and Output:

Explanation:

- Base case stops recursion when search range is invalid

- Recursive case reduces problem size each call

Code is beginner-friendly, well-commented, and logically clear.

Task 4: Ethical Evaluation of AI-Based Scoring Systems

PROMPT: Generate a Scoring Code:

CODE AND OUTPUT:

Bias Analysis:

- No gender or name used

- Logic is objective and skill-based

Ethical Evaluation:

Scoring is fair as it relies on job-relevant attributes.

The screenshot shows a code editor interface with a dark theme. On the left, the code file is named "main.py". The code itself is a simple Python script that prompts the user for three scores (skills, experience, education) and calculates a final score based on a weighted sum. The "Run" button is highlighted in blue at the top right. To the right of the code, the "Output" pane displays the execution results. It shows the user inputs for each score followed by the calculated final score, and concludes with a message indicating successful code execution.

```
main.py [Run] ⌂ ⌓ ⌚ Run Output Clear
1 Enter skills score: 15
2 skills = float(input("Enter skills score:
      "))
3 Enter experience score: 34
4 experience = float(input("Enter experience
      score: "))
5 Enter education score: 45
6 education = float(input("Enter education
      score: "))
7
8
9 print("Final Score:", score)
10
```

Output:

```
Enter skills score: 15
Enter experience score: 34
Enter education score: 45
Final Score: 26.7
== Code Execution Successful ==
```

Task 5: Inclusiveness and Ethical Variable Design

PROMPT: generate a code for bonus based on if gender is male

CODE:

```
python

if gender == 'male':
    bonus = 1000
else:
    bonus = 500
```

Issues:

- Gender-based assumptions
- Non-inclusive logic

Revised Inclusive Code:

PROMPT: GENERATE A CODE BASED ON PERFORMANCE

CODE AND OUPUT:

```
1 # Input performance rating
2 performance_rating = float(input("Enter
    performance rating (1 to 5): "))
3
4 # Bonus calculation based on performance
5 if performance_rating >= 4:
6     bonus = 1000
7 else:
8     bonus = 500
9
10 # Output
11 print("Bonus Amount:", bonus)
12 |
13
```

Enter performance rating (1 to 5): 4
Bonus Amount: 1000
== Code Execution Successful ==

Improvements:

- Removed gender dependency
- Used performance-based logic