

AI ASSISTED CODING

LAB-5: ETHICAL FOUNDATIONS – RESPONSIBLE AI CODING PRACTICES

ROLLNO:2503A51L10

NAME:K.praneeth

BATCH:25BTCAICSB19

TASK-1:

PROMPT:Generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

Code Generated:

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Charan\OneDrive\Desktop\python> python l5task1.py

1. Register
2. Login
3. Exit
Choose an option: 1
Enter new username: charan
Enter new password:
Registration successful.

1. Register
2. Login
3. Exit
Choose an option: 2
Enter username: charan
Enter password:
Login successful.

1. Register
2. Login
3. Exit
Choose an option: 3
PS C:\Users\Charan\OneDrive\Desktop\python> 
```

```

I5task1.py X
I5task1.py
1  import hashlib
2  import getpass
3
4  # Simple in-memory user database (username: hashed_password)
5  users_db = {}
6
7  def hash_password(password):
8      """Hash a password for storing."""
9      return hashlib.sha256(password.encode()).hexdigest()
10
11 def register():
12     username = input("Enter new username: ")
13     if username in users_db:
14         print("Username already exists.")
15         return
16     password = getpass.getpass("Enter new password: ")
17     users_db[username] = hash_password(password)
18     print("Registration successful.")
19
20 def login():
21     username = input("Enter username: ")
22     if username not in users_db:
23         print("Username not found.")
24         return
25     password = getpass.getpass("Enter password: ")
26     if users_db[username] == hash_password(password):
27         print("Login successful.")
28     else:
29         print("Incorrect password.")
30
31 def main():
32     while True:
33         print("\n1. Register\n2. Login\n3. Exit")
34         choice = input("Choose an option: ")
35         if choice == '1':
36             register()
37         elif choice == '2':
38             login()
39         elif choice == '3':
40             break
41         else:
42             print("Invalid choice.")
43
44 if __name__ == "__main__":
45     main()

```

TASK-2:

PROMPT: Loan approval for [Name] Applicant earns 50,000 per month, has credit score of 720, and no outstanding loans. Let the user give input

Code generated:

Output:

```

l5task1.py  l5task2.py X
l5task2.py
1  def evaluate_loan(name, salary, credit_score, outstanding_loans):
2      print(f"\nEvaluating loan application for {name}...")
3
4      # Simple approval criteria
5      if salary >= 50000 and credit_score >= 700 and outstanding_loans == 0:
6          print(f"Loan Approved for {name} ✅")
7      else:
8          print(f"Loan Rejected for {name} ❌")
9
10
11 # --- Take number of applicants ---
12 try:
13     n = int(input("Enter the number of applicants: "))
14 except ValueError:
15     print("Invalid number. Exiting.")
16     exit()
17
18 for i in range(n):
19     print(f"\n--- Applicant {i+1} ---")
20     name = input("Enter applicant's name: ")
21
22     # Read and validate numeric inputs
23     try:
24         salary = int(input("Enter monthly salary: "))
25         credit_score = int(input("Enter credit score: "))
26         outstanding_loans = int(input("Enter number of outstanding loans: "))
27     except ValueError:
28         print("Invalid input (must be numbers). Skipping this applicant.")
29         continue
30
31     evaluate_loan(name, salary, credit_score, outstanding_loans)
32
33 print("\nAll applications processed.")

```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Charan\OneDrive\Desktop\python> python l5task2.py
Enter the number of applicants: 3

--- Applicant 1 ---
Enter applicant's name: charan
Enter monthly salary: 60000
Enter credit score: 750
Enter number of outstanding loans: 0

Evaluating loan application for charan...
Loan Approved for charan ✅

--- Applicant 2 ---
Enter applicant's name: raju
Enter monthly salary: 50000
Enter credit score: 520
Enter number of outstanding loans: 1

Evaluating loan application for raju...
Loan Rejected for raju ❌

--- Applicant 3 ---
Enter applicant's name: kamei
Enter monthly salary: 20000
Enter credit score: 720
Enter number of outstanding loans: 3

Evaluating loan application for kamei...
Loan Rejected for kamei ❌

All applications processed.
PS C:\Users\Charan\OneDrive\Desktop\python>

```

TASK-3:

PROMPT: Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

CodeGenerated:

Output:

I5task1.pyI5task2.pyI5task3.py X

I5task3.py

```
1 # Prompt: Write a function to calculate the nth Fibonacci number using recursion.
2 # Add comments and explain the code for documentation.
3
4 def fibonacci(n):
5     """
6     Recursively calculates the nth Fibonacci number.
7
8     The Fibonacci sequence is defined as:
9     F(0) = 0
10    F(1) = 1
11    F(n) = F(n-1) + F(n-2) for n > 1
12
13    Args:
14        n (int): The position in the Fibonacci sequence (non-negative integer).
15
16    Returns:
17        int: The nth Fibonacci number.
18
19    Raises:
20        ValueError: If n is negative.
21    """
22    if n < 0:
23        raise ValueError("Input must be a non-negative integer.")
24    # Base cases: F(0) = 0, F(1) = 1
25    if n == 0:
26        return 0
27    elif n == 1:
28        return 1
29    # Recursive case: F(n) = F(n-1) + F(n-2)
30    else:
31        return fibonacci(n - 1) + fibonacci(n - 2)
32
33 # Example usage:
34 if __name__ == "__main__":
35     num = int(input("Enter the position (n) for Fibonacci number: "))
36     print(f"The {num}th Fibonacci number is {fibonacci(num)}")
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

The 10th Fibonacci number is 55
PS C:\Users\Charan\OneDrive\Desktop\python> python 1
Enter the position (n) for Fibonacci number: 10
The 10th Fibonacci number is 55
PS C:\Users\Charan\OneDrive\Desktop\python>

TASK-4:

PROMPT: Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyse the scoring logic for bias or unfair weightings.

Code Generated:

Output:

I5task1.pyI5task2.pyI5task3.pyI5task4.py X

I5task4.py

```
1 # Prompt: Generate a job applicant scoring system based on input
2 # Analyze the scoring logic for bias or unfair weightings.
3 # Let the user give input through the terminal.
4
5 def score_applicant(education, experience, gender, age):
6     """
7     Scores a job applicant based on input features.
8     Parameters:
9         education (str): 'highschool', 'bachelor', 'master', 'phd'
10        experience (int): years of relevant experience
11        gender (str): 'male', 'female', 'other'
12        age (int): age in years
13    Returns:
14        int: applicant score (0-100)
15    """
16
17    # Education scoring
18    edu_scores = {
19        'highschool': 10,
20        'bachelor': 25,
21        'master': 35,
22        'phd': 40
23    }
24    score = edu_scores.get(education.lower(), 0)
25
26    # Experience scoring
27    if experience < 1:
28        score += 0
29    elif experience < 3:
30        score += 10
31    elif experience < 5:
32        score += 20
33    else:
34        score += 30
35
36    # Gender scoring (no bias, all equal)
37    score += 0
38
39    # Age scoring (avoid bias, no weight)
40    score += 0
41
42    # Normalize score to 100
43    return min(score, 100)
44
45 if __name__ == "__main__":
46     education = input("Enter education (highschool/bachelor/maste
47     experience = int(input("Enter years of experience: "))
48     gender = input("Enter gender (male/female/other): ")
49     age = int (input("Enter age: "))
50
51     score = score_applicant(education, experience, gender, age)
52     print("Applicant Score:", score)
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

Applicant Score: 70
PS C:\Users\Charan\OneDrive\Desktop\python> python I5task4.py
Enter education (highschool/bachelor/master/phd): phd
Enter years of experience: 5
Enter gender (male/female/other): male
Enter age: 35
Applicant Score: 70
PS C:\Users\Charan\OneDrive\Desktop\python>

Task-5:

PROMPT:Regenerate code that includes gender-neutral also

```
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

CodeGenerated:

Output:

I5task1.pyI5task2.pyI5task3.pyI5task4.pyI5task5.py X

I5task5.py

```
1 def greet_user(name, gender):
2     """
3     Function to greet a user with gender-appropriate or gender-neutral title.
4     """
5     gender = gender.lower()
6
7     if gender == "male":
8         title = "Mr."
9     elif gender == "female":
10        title = "Ms." # Neutral instead of Mrs.
11    elif gender == "neutral":
12        title = "Mx." # Commonly used gender-neutral title
13    else:
14        title = "" # If gender is unknown/other, no title
15
16    return f"Hello, {title} {name}! Welcome."
17
18
19 # ---- Main Program ----
20 name = input("Enter your name: ")
21 gender = input("Enter your gender (male/female/neutral/other): ")
22
23 message = greet_user(name, gender)
24 print(message)
```

PROBLEMSOUTPUTDEBUG CONSOLETERMINALPORTS

```
PS C:\Users\Charan\OneDrive\Desktop\python> python I5task5.py
Enter your name: charan
Enter your gender (male/female/neutral/other): male
Hello, Mr. charan! Welcome.
PS C:\Users\Charan\OneDrive\Desktop\python> 
```

OBSERVATION: I observed that GitHub copilot can quickly generate working code for tasks such as login systems, loan approvals, Fibonacci functions, and job applicant scoring. However, the generated code sometimes contains issues like hardcoded values, lack of encryption, or biased decision logic. This shows that AI tools are helpful for faster coding but require human review for

security, fairness, and correctness. GitHub Copilot is a fascinating tool to observe—especially in how it transforms the developer experience. Here's a breakdown of key observations across its functionality, impact, and adoption