

D.HAVISH

BATCH – 42

2303A52253

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

Prompt:

Generate a simple login system python code in which can register and login users details manually and runs until we exist

CODE:

```
▶ import hashlib
users = {}
while True:
    print("\n1. Registration")
    print("2. Login")
    print("3. Exit")
    choice = input("Enter your choice: ")
    # ----- REGISTRATION -----
    if choice == "1":
        print("\n--- Registration ---")
        username = input("Create username: ").strip()
        password = input("Create password: ").strip()

        if username == "" or password == "":
            print("Username and password cannot be empty")
        elif username in users:
            print("Username already exists")
        else:
            hashed_password = hashlib.sha256(password.encode()).hexdigest()
            users[username] = hashed_password
            print("Registration successful")
    # ----- LOGIN -----
    elif choice == "2":
        print("\n--- Login ---")
        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:
```

```
        if username in users and users[username] == hashed_password:
            print("Login successful")
        else:
            print("Invalid username or password")
# ----- EXIT -----
elif choice == "3":
    print("Exiting program")
    break

else:
    print("Invalid choice, try again")
```

OUTPUT:

```
1. Registration
2. Login
3. Exit
...
Enter your choice: 1

--- Registration ---
Create username: naziya
Create password: 1234
Registration successful

1. Registration
2. Login
3. Exit
Enter your choice: 2

--- Login ---
Enter username: naziya
Enter password: 4567
Invalid username or password

1. Registration
2. Login
3. Exit
Enter your choice: 2

--- Login ---
Enter username: naziya
Enter password: 1234
Login successful

1. Registration
2. Login
3. Exit
```

```
1. Registration
2. Login
3. Exit
Enter your choice: 3
Exiting program
```

IMPROVEMENTS:

```

import re
import time

USERNAME_RE = re.compile(r"^[A-Za-z0-9_]{3,20}$")

def validate_username(username: str) -> tuple[bool, str]:
    if not USERNAME_RE.fullmatch(username):
        return False, "Username must be 3-20 chars: letters, numbers, underscore only."
    return True, ""

def validate_password(password: str) -> tuple[bool, str]:
    if len(password) < 8:
        return False, "Password must be at least 8 characters."
    if not any(c.isalpha() for c in password) or not any(c.isdigit() for c in password):
        return False, "Password must contain at least 1 letter and 1 digit."
    return True, ""

failed = 0

# inside the while loop, in the login branch:
ok, msg = login(username, password)
print(msg)
if not ok:
    failed += 1
    if failed >= 5:
        print("Too many failed attempts. Waiting 10 seconds...")
        time.sleep(10)
        failed = 0
else:
    failed = 0

```

OUTPUT:

```

Simple Login System
-----
Choose an option:
1) Register
2) Login
3) Exit
> 2
Username: Sikindhar
Password:
Welcome, Sikindhar!

```

```
Choose an option:  
1) Register  
2) Login  
3) Exit  
> 2  
Username: Testbeta  
Password:  
Invalid username or password.
```

JUSTIFICATION:

This program implements a basic registration and login system using user input without hardcoding any credentials. Usernames and passwords are stored dynamically during runtime, ensuring better privacy and flexibility. The system continuously runs until the user chooses to exit.

Task 2: Bias Detection in AI-Generated Decision Systems

Prompt:

Create a loan approval system based on user input

CODE

```
▶ def loan_approval_system(age, income, credit_score, loan_amount):  
    if age < 18:  
        return "Loan Denied: Applicant must be at least 18 years old."  
  
    if income < 20000:  
        return "Loan Denied: Income must be at least $20,000."  
  
    if credit_score < 600:  
        return "Loan Denied: Credit score must be at least 600."  
  
    if loan_amount > income * 5:  
        return "Loan Denied: Loan amount exceeds allowable limit."  
  
    return "Loan Approved!"  
  
name = input("Enter your name: ")  
age = int(input("Enter your age: "))  
income = float(input("Enter your annual income: "))  
credit_score = int(input("Enter your credit score: "))  
loan_amount = float(input("Enter desired loan amount: "))  
  
result = loan_approval_system(age, income, credit_score, loan_amount)  
print(result)
```

OUTPUT:

```
... Enter your name: naziya  
Enter your age: 20  
Enter your annual income: 20000  
Enter your credit score: 3  
Enter desired loan amount: 3000  
Loan Denied: Credit score must be at least 600.
```

Justification:

By this code we can get to know whether the loan is approved or denied, we can clearly see that loan approval decision is not at all depending on name, caste, gender and any other irrelevant aspects which shows the transparency and fairness of the code generated by A.I.

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

Binary search on a sorted list using recursion with comment

Code:

```

▶ def binary_search(arr, low, high, target):
    # Base condition: if range is valid
    if low <= high:
        mid = (low + high) // 2

        # If target is found at mid
        if arr[mid] == target:
            return mid

        # If target is smaller, search left half
        elif arr[mid] > target:
            return binary_search(arr, low, mid - 1, target)

        # If target is larger, search right half
        else:
            return binary_search(arr, mid + 1, high, target)

    # Target not found
    return -1

arr = [10, 20, 30, 40, 50]
target = int(input("Enter element to search: "))
result = binary_search(arr, 0, len(arr) - 1, target)

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

```

OUTPUT:

```

...
... Enter element to search: 30
      Element found at index 2

```

Justification:

In the above generated code we are searching the number in a sorted array using binary search on a sorted list. We are performing by recursion which helps in reusing of code and also having step by step explanation(comments in code) which plays a key role in easy understanding of the code.

Task 4: Ethical Evaluation of AI-Based Scoring Systems

Prompt:

Job application scoring system based on users input

Code:

```
▶  experience = int(input("Enter years of experience: "))
skills = int(input("Enter skill rating (1-10): "))
education = int(input("Enter education score (1-10): "))

score = (experience * 2) + (skills * 3) + (education * 2)

print("Application Score:", score)

if score >= 50:
    print("Candidate Selected")
else:
    print("Candidate Not Selected")
```

Output:

```
... Enter years of experience: 2
Enter skill rating (1-10): 5
Enter education score (1-10): 8
Application Score: 35
Candidate Not Selected
```

Justification:

In the above generated code we are calculating the job application score based on experience, skills, and education. The score helps in deciding whether the candidate is selected or not. Only relevant parameters are considered, which plays a key role in ensuring fairness and transparency in the job selection process.

Task 5: Inclusiveness and Ethical Variable Design**Prompt:**

Create a Python employee management system that stores employee details including name, gender (male/female), job title, and salary. Add a function that gives different benefits: maternity leave for females and paternity leave for males. Use Mr./Mrs. titles based on gender when displaying employee information. user input should be taken for adding employees and displaying their details along with the benefits they are entitled to.

Code:

```
employees = []
while True:
    print("\n1. Add Employee")
    print("2. Display Employees")
    print("3. Exit")
    choice = input("Enter choice: ")
    if choice == "1":
        name = input("Enter name: ")
        gender = input("Enter gender (male/female): ").lower()
        job = input("Enter job title: ")
        salary = int(input("Enter salary: "))

        employees.append((name, gender, job, salary))
        print("Employee added successfully")

    elif choice == "2":
        for emp in employees:
            name, gender, job, salary = emp

            if gender == "female":
                title = "Mrs."
            else:
                title = "Mr."

            print("\nEmployee Details")
            print("Name:", title, name)
            print("Job Title:", job)
            print("Salary:", salary)
```

```
elif choice == "3":
    print("Exiting program")
    break

else:
    print("Invalid choice")
```

Output:

...

```
1. Add Employee
2. Display Employees
3. Exit
Enter choice: 1
Enter name: naziya
Enter gender (male/female): female
Enter job title: web dev
Enter salary: 50000
Employee added successfully
```

```
1. Add Employee
2. Display Employees
3. Exit
Enter choice: 2
```

```
Employee Details
Name: Mrs. naziya
Job Title: web dev
Salary: 50000
```

```
1. Add Employee
2. Display Employees
3. Exit
Enter choice: 3
Exiting program
```

Justification:

In the above generated code we are managing employee details such as name, gender, job title, and salary. The system provides benefits like maternity or paternity leave based on gender and displays employee information using appropriate titles. This approach plays a key role in promoting inclusiveness, ethical design, and easy understanding of employee benefits.