AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
"""

Date: January 28, 2026

"""


# ============================================================================

# TASK 1: Privacy and Data Security in AI-Generated Code

# ============================================================================

# PROMPT: This task demonstrates the importance of secure password storage.

# Compare insecure (plain text) vs secure (hashed) password systems.

# Learn how proper password hashing protects user data from security breaches.

# ============================================================================


import hashlib


# Storage for registered users (in-memory for demo purposes)

insecure_users = {}  # Plain text passwords - BAD

secure_users = {}    # Hashed passwords - GOODn


def hash_password(password):

    """Hash password using SHA-256"""

    return hashlib.sha256(password.encode()).hexdigest()
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
def is_password_strong(password):
    """
    Check if password meets strong password criteria
    Requirements:
    - At least 8 characters long
    - Contains at least one uppercase letter
    - Contains at least one lowercase letter
    - Contains at least one digit
    - Contains at least one special character
    """
    if len(password) < 8:
        return False, "Password must be at least 8 characters long"

    has_upper = any(c.isupper() for c in password)
    has_lower = any(c.islower() for c in password)
    has_digit = any(c.isdigit() for c in password)
    has_special = any(c in "!@#$%^&*()_+-=[]{}|;:,.<>?" for c in password)

    if not has_upper:
        return False, "Password must contain at least one uppercase letter"
    if not has_lower:
        return False, "Password must contain at least one lowercase letter"
    if not has_digit:
        return False, "Password must contain at least one digit"
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    if not has_special:
        return False, "Password must contain at least one special character (!@#$%^&* etc.)"

    return True, "Password is strong"

def insecure_register():
    """Insecure registration - stores plain text passwords"""
    print("\n--- INSECURE REGISTRATION ---")
    print("Issues: Stores passwords in plain text\n")

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

    if not username or not password:
        print("✗ Username and password cannot be empty!")
        return

    if username in insecure_users:
        print("✗ Username already exists!")
        return

    # SECURITY RISK: Storing password in plain text
    insecure_users[username] = password
    print(f"✓ Registration successful! (Password stored as: {password}) ⚠️ INSECURE!")
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
def insecure_login():
    """Insecure login system"""
    print("\n--- INSECURE LOGIN SYSTEM ---")
    print("Issues: Plain text password comparison\n")

    if not insecure_users:
        print("✗ No users registered! Please register first.")
        return

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

    # Plain text comparison - SECURITY RISK
    if username in insecure_users and insecure_users[username] == password:
        print("✓ Login successful!")
        print(f" ⚠️ Your password '{password}' is visible in database!")
    else:
        print("✗ Login failed! Username or password incorrect.")

def secure_register():
    """Secure registration - stores hashed passwords"""
    print("\n--- SECURE REGISTRATION ---")
    print("Improvements: Password hashing, strong password validation\n")
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

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

# Input validation
if not username or not password:
    print("✗ Username and password cannot be empty!")
    return

# Check password strength
is_strong, message = is_password_strong(password)
if not is_strong:
    print(f"✗ Password is not strong enough! {message}")
    print("\nPassword Requirements:")
    print("  • At least 8 characters long")
    print("  • At least one uppercase letter (A-Z)")
    print("  • At least one lowercase letter (a-z)")
    print("  • At least one digit (0-9)")
    print("  • At least one special character (!@#$%^&* etc.)")
    return

if username in secure_users:
    print("✗ Username already exists!")
    return
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    # SECURE: Store hashed password
    secure_users[username] = hash_password(password)
    print("✓ Registration successful! Password stored securely (hashed).")

def secure_login():
    """Secure login system with hashed passwords"""
    print("\n--- SECURE LOGIN SYSTEM ---")
    print("Improvements: Password hashing, input validation\n")

    if not secure_users:
        print("✗ No users registered! Please register first.")
        return

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

    # Input validation
    if not username or not password:
        print("✗ Username and password cannot be empty!")
        return

    # Compare hashed passwords - SECURE
    if username in secure_users and secure_users[username] ==
hash_password(password):
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
        print("✓ Login successful!")
        print("✓ Your password is securely hashed in database!")
    else:
        print("✗ Invalid credentials! Username or password incorrect.")


def task1_demo():
    print("=" * 70)
    print("TASK 1: Privacy and Data Security")
    print("=" * 70)

    print("\n[1] Insecure System (Plain Text)")
    print("[2] Secure System (Hashed)")
    system_choice = input("\nSelect system (1/2): ")

    if system_choice == "1":
        print("\n[A] Register")
        print("[B] Login")
        action = input("\nSelect action (A/B): ").upper()

        if action == "A":
            insecure_register()
        elif action == "B":
            insecure_login()
        else:
            print("Invalid choice!")
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
elif system_choice == "2":
    print("\n[A] Register")
    print("[B] Login")
    action = input("\nSelect action (A/B): ").upper()

    if action == "A":
        secure_register()
    elif action == "B":
        secure_login()
    else:
        print("Invalid choice!")
else:
    print("Invalid choice!")

print("\nSecurity Explanation:")
print("- Insecure: Plain text passwords (visible in database)")
print("- Secure: Hashed passwords (SHA-256), input validation")
```

OUTPUT :

OUTPUT :

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

OUTPUT :

```
TASK 1: Privacy and Data Security
================================================================

[1] Insecure System (Plain Text)
[2] Secure System (Hashed)

Select system (1/2): 1

[A] Register
[B] Login

Select action (A/B): A

--- INSECURE REGISTRATION ---
Issues: Stores passwords in plain text

Enter username: Dinesh
Enter password: dinesh123
✓ Registration successful! (Password stored as: dinesh123) ⚠INSECURE!

Security Explanation:
- Insecure: Plain text passwords (visible in database)
- Secure: Hashed passwords (SHA-256), input validation
```

JUSTIFICATION :

This task emphasizes the necessity of safeguarding sensitive user information by demonstrating why passwords should never be stored in plain text. It reinforces responsible data management practices and the role of security measures in maintaining user trust.

```
================================================================
======

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

#
================================================================
======

# PROMPT: This task illustrates how AI systems can contain hidden biases.

# Compare a biased loan approval system (using gender/name) vs a fair system.

# Learn to identify and eliminate discriminatory factors in decision-making.

#
================================================================
======
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
def biased_loan_approval(name, gender, income, credit_score):
    """Biased loan system - CONTAINS BIAS"""
    approved = False

    # BIAS: Gender-based approval
    if gender == "male" and income > 30000 and credit_score > 650:
        approved = True
    elif gender == "female" and income > 35000 and credit_score > 700:
        # Higher requirements for females - UNFAIR BIAS
        approved = True

    return approved

def fair_loan_approval(income, credit_score, employment_years):
    """Fair loan system based only on relevant financial factors"""
    # Gender-neutral, name-neutral evaluation
    if income > 30000 and credit_score > 650 and employment_years >= 2:
        return True
    return False

def task2_demo():
    print("\n" + "=" * 70)
    print("TASK 2: Bias Detection in Loan Approval System")
    print("=" * 70)
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
choice = input("\nTest [1] Biased System or [2] Fair System? (1/2): ")


if choice == "1":
    print("\n--- BIASED LOAN APPROVAL SYSTEM ---")
    name = input("Enter applicant name: ")
    gender = input("Enter gender (male/female): ").lower()
    income = int(input("Enter annual income: "))
    credit_score = int(input("Enter credit score: "))

    result = biased_loan_approval(name, gender, income, credit_score)
    print(f"\nLoan Approval: {'✓ APPROVED' if result else '✗ DENIED'}")
    print("\nBIAS WARNING: This system uses different criteria for different genders!")

elif choice == "2":
    print("\n--- FAIR LOAN APPROVAL SYSTEM ---")
    income = int(input("Enter annual income: "))
    credit_score = int(input("Enter credit score: "))
    employment_years = int(input("Enter years of employment: "))

    result = fair_loan_approval(income, credit_score, employment_years)
    print(f"\nLoan Approval: {'✓ APPROVED' if result else '✗ DENIED'}")
    print("\nThis system uses only relevant financial criteria (no bias)")
else:
    print("Invalid choice!")
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
print("\nMitigation: Use only relevant financial criteria, no gender/name bias")
```

OUTPUT :

```
Select task (1-6): 2

======================================================================
TASK 2: Bias Detection in Loan Approval System
======================================================================

Test [1] Biased System or [2] Fair System? (1/2): 1

--- BIASED LOAN APPROVAL SYSTEM ---
Enter applicant name: Dinesh
Enter gender (male/female): male
Enter annual income: 500002
Enter credit score: 65

Loan Approval: ✗ DENIED

BIAS WARNING: This system uses different criteria for different genders!

Mitigation: Use only relevant financial criteria, no gender/name bias
```

JUSTIFICATION :

This task illustrates how incorporating personal attributes such as gender or names into automated decisions can lead to discriminatory outcomes. It underlines the importance of designing AI systems that promote fairness and avoid biased judgments.

```
================================================================================
```

```
# TASK 3: Transparency and Explainability - Recursive Binary Search
#
================================================================================
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
# PROMPT: This task emphasizes the importance of transparent, explainable code.
# See how a binary search algorithm can be made clear with step-by-step output.
# Learn to write code that is easy to understand and debug for all developers.
#
================================================================================

def binary_search_recursive(arr, target, left, right, depth=0):
    """
    Recursive Binary Search with detailed explanations

    Parameters:
    - arr: sorted list of numbers
    - target: element to search for
    - left: left boundary index
    - right: right boundary index
    - depth: recursion depth for visualization

    Returns:
    - Index of target if found, -1 otherwise
    """

    # STEP 1: BASE CASE - Search space exhausted
    if left > right:
        print(f"{'  ' * depth}Base case: left > right, element not found")
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    return -1

  # STEP 2: Calculate middle index
  mid = (left + right) // 2
  print(f"{'  ' * depth}Depth {depth}: Checking middle index {mid}, value = {arr[mid]}")

  # STEP 3: BASE CASE - Element found
  if arr[mid] == target:
    print(f"{'  ' * depth}Found! Target {target} at index {mid}")
    return mid

  # STEP 4: RECURSIVE CASE - Search left half
  elif arr[mid] > target:
    print(f"{'  ' * depth}Target {target} < {arr[mid]}, search LEFT half")
    return binary_search_recursive(arr, target, left, mid - 1, depth + 1)

  # STEP 5: RECURSIVE CASE - Search right half
  else:
    print(f"{'  ' * depth}Target {target} > {arr[mid]}, search RIGHT half")
    return binary_search_recursive(arr, target, mid + 1, right, depth + 1)

def task3_demo():
  print("\n" + "=" * 70)
  print("TASK 3: Transparent Recursive Binary Search")
  print("=" * 70)
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
sorted_list = [2, 5, 8, 12, 16, 23, 38, 45, 56, 67, 78]
print(f"\nSorted list: {sorted_list}")

target = int(input("Enter number to search: "))

print("\nStep-by-step execution:")
result = binary_search_recursive(sorted_list, target, 0, len(sorted_list) - 1)

if result != -1:
    print(f"\n✓ Result: Element {target} found at index {result}")
else:
    print(f"\n✗ Result: Element {target} not found in list")

print("\nTransparency Assessment:")
print("✓ Clear comments explaining each step")
print("✓ Base cases and recursive cases explained")
print("✓ Visualization of recursion depth")
print("✓ Easy to understand for beginners")
```

OUTPUT :

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

OUTPUT :

```
Select task (1-6): 3

======================================================================
TASK 3: Transparent Recursive Binary Search
======================================================================

Sorted list: [2, 5, 8, 12, 16, 23, 38, 45, 56, 67, 78]
Enter number to search: 78

Step-by-step execution:
Depth 0: Checking middle index 5, value = 23
Target 78 > 23, search RIGHT half
  Depth 1: Checking middle index 8, value = 56
  Target 78 > 56, search RIGHT half
    Depth 2: Checking middle index 9, value = 67
    Target 78 > 67, search RIGHT half
      Depth 3: Checking middle index 10, value = 78
      Found! Target 78 at index 10

✓ Result: Element 78 found at index 10

Transparency Assessment:
✓ Clear comments explaining each step
✓ Base cases and recursive cases explained
✓ Visualization of recursion depth
✓ Easy to understand for beginners
```

JUSTIFICATION :

This task highlights the value of transparent system design by showing how clear explanations and documentation help users understand how decisions are made, increasing accountability and confidence in AI systems.

==========

TASK 4: Ethical Evaluation of AI-Based Scoring Systems

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
#
=============================================================================
======

# PROMPT: This task explores fairness in automated scoring/ranking systems.

# Compare biased scoring (using name/gender) vs fair scoring (merit-based only).

# Learn to design evaluation systems that judge people on relevant criteria only.

#
=============================================================================
======


def biased_applicant_score(name, gender, skills, experience, education):

    """Biased scoring - UNETHICAL"""

    score = 0


    # Skill scoring

    score += skills * 20

    score += experience * 10

    score += education * 15


    # BIAS: Gender-based bonus - UNETHICAL

    if gender == "male":

        score += 10  # Unfair advantage


    # BIAS: Name-based assumption - UNETHICAL

    if name in ["John", "Michael", "David"]:

        score += 5  # Cultural/name bias
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    return score


def fair_applicant_score(skills_rating, years_experience, education_level,
                         certifications, portfolio_quality):
    """
    Fair scoring based only on relevant professional factors
    All personal attributes (name, gender, age, etc.) are excluded
    """
    score = 0

    # Skills (0-10 scale)
    score += skills_rating * 20  # Max: 200 points

    # Experience (years)
    score += min(years_experience * 10, 100)  # Max: 100 points

    # Education level (1=High School, 2=Bachelor, 3=Master, 4=PhD)
    score += education_level * 15  # Max: 60 points

    # Certifications count
    score += certifications * 10  # Max: varies

    # Portfolio quality (0-10 scale)
    score += portfolio_quality * 15  # Max: 150 points
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    return score


def task4_demo():
    print("\n" + "=" * 70)
    print("TASK 4: Ethical Job Applicant Scoring System")
    print("=" * 70)

    choice = input("\nTest [1] Biased Scoring or [2] Fair Scoring? (1/2): ")

    if choice == "1":
        print("\n--- BIASED SCORING SYSTEM ---")
        name = input("Enter applicant name: ")
        gender = input("Enter gender (male/female): ").lower()
        skills = int(input("Enter skills rating (0-10): "))
        experience = int(input("Enter years of experience: "))
        education = int(input("Enter education level (1-4): "))

        score = biased_applicant_score(name, gender, skills, experience, education)
        print(f"\n✗ Total Score: {score}")
        print("BIAS DETECTED: Gender and name influence score!")

    elif choice == "2":
        print("\n--- FAIR SCORING SYSTEM ---")
        skills = int(input("Enter skills rating (0-10): "))
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    experience = int(input("Enter years of experience: "))
    education = int(input("Enter education level (1=HS, 2=Bach, 3=Mast, 4=PhD): "))
    certs = int(input("Enter number of certifications: "))
    portfolio = int(input("Enter portfolio quality (0-10): "))

    score = fair_applicant_score(skills, experience, education, certs, portfolio)
    print(f"\n✓ Total Score: {score}")
    print("\nEthical Analysis:")
    print("✓ No gender, name, or identity factors")
    print("✓ Only relevant professional criteria")
  else:
    print("Invalid choice!")
```

OUTPUT :

```
Select task (1-6): 4

================================================================================
TASK 4: Ethical Job Applicant Scoring System
================================================================================

Test [1] Biased Scoring or [2] Fair Scoring? (1/2): 1

--- BIASED SCORING SYSTEM ---
Enter applicant name: DINESH
Enter gender (male/female): MALE
Enter skills rating (0-10): 9
Enter years of experience: 6
Enter education level (1-4): 3

✗ Total Score: 295
BIAS DETECTED: Gender and name influence score!
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

JUSTIFICATION :

This task demonstrates the impact of ethical evaluation methods by comparing biased and unbiased scoring approaches. It stresses that fair AI systems must assess individuals solely on merit and relevant performance criteria.

```
================================================================================
```

```
# TASK 5: Inclusiveness and Ethical Variable Design
#
================================================================================
# PROMPT: This task demonstrates inclusive and respectful software design.
# Learn how to create gender-neutral systems that respect all users equally.
# See how fair compensation should be based on role and experience, not identity.
#
================================================================================


def calculate_salary(position, experience):
    """Fair salary calculation based on role and experience"""
    base_salaries = {
        "Developer": 45000,
        "Senior Developer": 65000,
        "Lead Developer": 85000
    }
    base = base_salaries.get(position, 40000)
    experience_bonus = experience * 2000
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    return base + experience_bonus


def task5_demo():
    print("\n" + "=" * 70)
    print("TASK 5: Inclusive and Ethical Variable Design")
    print("=" * 70)

    print("\nNON-INCLUSIVE VERSION:")
    print("❌ Separates employees by gender (male_employees, female_employees)")
    print("❌ Different salaries based on gender")
    print("❌ Assumes titles (Mr./Ms.)")

    print("\nINCLUSIVE VERSION:")
    print("✓ Gender-neutral variables")
    print("✓ Salary based on position and experience only")
    print("✓ Optional user-specified titles")

    print("\n--- ADD EMPLOYEE (INCLUSIVE SYSTEM) ---")
    name = input("Enter employee name: ")
    position = input("Enter position (Developer/Senior Developer/Lead Developer): ")
    experience = int(input("Enter years of experience: "))
    title = input("Enter preferred title (Mr./Ms./Mx./Dr. or leave blank): ").strip()

    salary = calculate_salary(position, experience)
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
print("\n--- EMPLOYEE INFORMATION ---")
if title:
    print(f"Name: {title} {name}")
else:
    print(f"Name: {name}")
print(f"Position: {position}")
print(f"Experience: {experience} years")
print(f"Salary: ${salary:,}")


print("\n✓ This system is inclusive and fair!")
print("✓ No gender assumptions or bias")
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

OUTPUT :

```
Select task (1-6): 5

======================================================================
TASK 5: Inclusive and Ethical Variable Design
======================================================================

NON-INCLUSIVE VERSION:
❌ Separates employees by gender (male_employees, female_employees)
❌ Different salaries based on gender
❌ Assumes titles (Mr./Ms.)

INCLUSIVE VERSION:
✓ Gender-neutral variables
✓ Salary based on position and experience only
✓ Optional user-specified titles

--- ADD EMPLOYEE (INCLUSIVE SYSTEM) ---
Enter employee name: DINESH
Enter position (Developer/Senior Developer/Lead Developer): Developer
Enter years of experience: 6
Enter preferred title (Mr./Ms./Mx./Dr. or leave blank): Mr.

--- EMPLOYEE INFORMATION ---
Name: Mr. DINESH
Position: Developer
Experience: 6 years
Salary: $57,000

✓ This system is inclusive and fair!
✓ No gender assumptions or bias
```

JUSTIFICATION :

This task focuses on the importance of inclusive system design by promoting the use of neutral variables. It ensures AI applications respect diversity and provide equal treatment to all users.

#
==========
======

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
# MAIN MENU
#
============================================================================
======


def main():
    while True:
        print("\n" + "=" * 70)
        print("LAB 5: Ethical Foundations – Responsible AI Coding Practices")
        print("=" * 70)
        print("\n1. Task 1: Privacy and Data Security")
        print("2. Task 2: Bias Detection in Loan Approval")
        print("3. Task 3: Transparent Recursive Binary Search")
        print("4. Task 4: Ethical Job Applicant Scoring")
        print("5. Task 5: Inclusive Variable Design")
        print("6. Exit")

        choice = input("\nSelect task (1-6): ")

        if choice == "1":
            task1_demo()
        elif choice == "2":
            task2_demo()
        elif choice == "3":
            task3_demo()
```

AI – ASSISTANT CODING - LAB 5.3

Lab 5: Ethical Foundations – Responsible AI Coding Practices

NAME : K. Dinesh
BATCH : 42
ROLL NO : 2303A52288

```
    elif choice == "4":
        task4_demo()
    elif choice == "5":
        task5_demo()
    elif choice == "6":
        print("\nLab completed. Goodbye!")
        break
    else:
        print("Invalid choice! Please select 1-6.")


if __name__ == "__main__":
    main()
```