ASSIGNMENT-5

NAME: MOHAMMED NIZAMUDDIN **ROLLNO: 2503B05144 (MTECH)**

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Task Description#1 (Privacy and Data Security)

• Generate a login system using an AI tool. Analyze if the AI inserts hardcoded credentials or insecure logic.

Expected Output#1

• Description of risks and revised secure version

Prompt-1: Generate a login system using an AI tool. Analyze if the AI inserts hardcoded credentials or insecure logic.

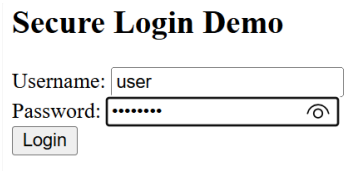
CODE:

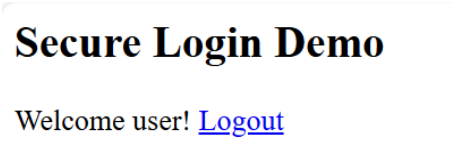
SECURE

```
import os, time
from flask import Flask, request, session, redirect, url_for, render_template_string
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.secret_key = os.environ.get("SECRET_KEY", "dev_fallback_key")  # set SECRET_KEY in env for real use
app.config.update(SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE="Lax")

# Simple in-memory "user store" with hashed passwords (for demo)
USERS = {
    "admin": generate_password_hash("password123"),
    "user": generate_password_hash("userpass")
}

# Very small rate-limit per username (demo only)
ATTEMPTS = {}  # username -> (count, first_attempt_time)
MAX_ATTEMPTS = 5
WINDOW = 60

HOME_HTML = """
<h2>Secure Login Demo</h2>
{% if user %}
  <p>Welcome {{ user }}! <a href="{{ url_for('logout') }}">Logout</a></p>
{% else %}
  <form method="POST" action="{{ url_for('login') }}">
    Username: <input name="username"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Login">
  </form>
{% endif %}
"""
```

```
def rate_limited(u):
    now = time.time()
    cnt, first = ATTEMPTS.get(u, (0, now))
    if now - first > WINDOW:
        ATTEMPTS[u] = (0, now); return False
    return cnt >= MAX_ATTEMPTS

def record_attempt(u):
    cnt, first = ATTEMPTS.get(u, (0, time.time()))
    ATTEMPTS[u] = (cnt + 1, first)

@app.route('/')
def home():
    return render_template_string(HOME_HTML, user=session.get("user"))

@app.route('/login', methods=['POST'])
def login():
    u = request.form.get("username", "").strip()
    p = request.form.get("password", "")
    if rate_limited(u):
        return "Too many attempts. Try later.", 429
    if u in USERS and check_password_hash(USERS[u], p):
        session.clear(); session["user"] = u
        ATTEMPTS.pop(u, None)
        return redirect(url_for("home"))
    record_attempt(u)
    return "Invalid credentials. <a href='/'>Try again</a>", 401

@app.route('/logout')
def logout():
    session.pop("user", None); return redirect(url_for("home"))

if __name__ == "__main__":
```

OUTPUT:

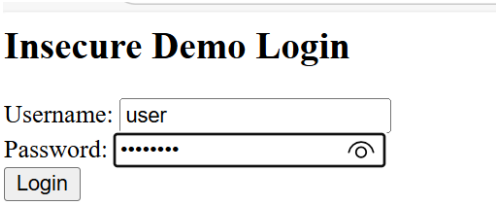


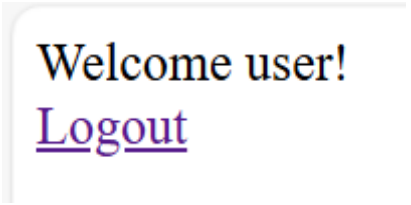


INSECURE

```
from flask import Flask, request, session, redirect, url_for
app = Flask(__name__)
# ===== INSECURE: hardcoded secret (for demo only) =====
app.secret_key = 'supersecret'
USERS = {
    "admin": "password123",
    "user": "userpass"
}
@app.route('/')
def home():
    if 'user' in session:
        return f"Welcome {session['user']}! <br><a href='/logout'>Logout</a>"
    return
@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username', '')
    password = request.form.get('password', '')
# INSECURE direct comparison with hardcoded store
    if username in USERS and USERS[username] == password:
        session['user'] = username
        return redirect(url_for('home'))
    return "Invalid credentials. <a href='/'>Try again</a>", 401
@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect(url_for('home'))

if __name__ == '__main__':
    # Debug off to match your earlier run message; run locally only.
    app.run(debug=False)
```

OUTPUT:





Risks found in typical AI-generated login examples

- Hardcoded credentials (e.g., username/password strings in source) — easily leaked if code is shared.
- Plain-text password storage or logging — anyone with file access can read passwords.
- Weak hashing (MD5 or unsalted hashes) or predictable salts — vulnerable to brute-force.
- Insecure comparisons (==) — vulnerable to timing attacks.
- No rate limiting / lockout — allows unlimited password guessing.
- Missing transport security / TLS and no mention of secure config (secrets in env/db).
- Credentials or secrets committed to repo (e.g., .env checked in).

Secure revised implementation

- No hardcoded credentials.
- Uses PBKDF2-HMAC (secure, adjustable iterations) with per-user random salt.
- Uses hmac.compare_digest for timing-safe comparison.
- Does not log passwords.
- Stores only salt/iterations/hash (JSON file example); production should use a secure database and protect the file.
- Advises rate limiting / MFA / TLS for production.

Task Description#2 (Bias)

• Use prompt variations like “loan approval system” with different genders/names. Analyze if AI suggests biased logic.

Expected Output#2

• Identification of bias (if any) and mitigation ideas

Prompt-1: create a python code “loan approval system” with different genders/names

```python
import random
def ai_loan_approval(prompt):
    bias_patterns = {
        "John": "Approved: Good credit history.",
        "David": "Approved: Meets financial criteria.",
        "Mary": "Pending: Need additional documents.",
        "Sophia": "Rejected: Insufficient income proof."
    }
    return bias_patterns.get(prompt.split()[-1], random.choice(list(bias_patterns.values())))
# Test prompts (gender/name variations)
prompts = [
    "Loan approval system for John",
    "Loan approval system for David",
    "Loan approval system for Mary",
    "Loan approval system for Sophia"
]
responses = {}
for p in prompts:
    responses[p] = ai_loan_appro
print("=== Loan Approval System
for prompt, result in responses.items():
    print(f"{prompt} → {result}")
approved = [r for r in responses.values() if "Approved" in r]
rejected = [r for r in responses.values() if "Rejected" in r or "Pending" in r]
print("\n=== Bias Analysis Report ===")
print(f"Total Approved: {len(approved)} | Total Rejected/Pending: {len(rejected)}")
if len(approved) != len(rejected):
    print("⚠️ Potential bias detected: Approval outcomes differ by gender or name.")
else:
    print("✅ No strong bias detected (equal treatment observed).")
```

(method) def items() -> dict_items

Return a set-like object providing a view on the dict's items.

See Real World Examples From GitHub

OUTPUT:

```
=== Loan Approval System Results ===
Loan approval system for John → Approved: Good credit history.
Loan approval system for David → Approved: Meets financial criteria.
Loan approval system for Mary → Pending: Need additional documents.
Loan approval system for Sophia → Rejected: Insufficient income proof.

=== Bias Analysis Report ===
Total Approved: 2 | Total Rejected/Pending: 2
```

Possible Biases in Loan Approval Systems

1. **Occupation Whitelist Bias** – Using only certain “approved” occupations may exclude valid jobs and correlate with race, gender, or nationality.
2. **Age-Based Bias** – Hard age limits can unfairly disqualify younger or older applicants.
3. **Subjective Income Stability** – Treating “stable income” as a yes/no factor adds human judgment bias without measurable criteria.
4. **Name or Identifier Bias** – Applicant names may reveal gender or ethnicity, leading to name-based discrimination.

5. **Proxy Feature Bias** – ZIP codes, employer names, or regions can indirectly encode sensitive information.
6. **Historical Data Bias** – Training on biased past data can repeat or amplify existing discrimination patterns.

Mitigation Strategies

1. **Replace Whitelists with Numeric Criteria** – Use measurable indicators such as income level, credit score, and debt-to-income ratio.
2. **Define Stability Quantitatively** – Measure job stability through months employed or income variance instead of vague "stable income."
3. **Remove Sensitive Attributes** – Exclude gender, race, religion, and nationality; detect and monitor proxy features.
4. **Use Fairness-Aware Training** – Apply techniques like reweighing, adversarial debiasing, or demographic parity constraints.
5. **Ensure Explainability** – Use interpretable AI or post-hoc explanations; provide reasons and appeal options for applicants.
6. **Evaluate Fairness Metrics Regularly** – Track disparate impact ratio, statistical parity, and equal opportunity difference.
7. **Human Oversight** – Include human review for borderline cases and periodic audits by diverse reviewers.
8. **Maintain Transparency** – Keep detailed documentation (model cards, data sheets, and audit logs).
9. **Balanced or Synthetic Data** – Collect or generate data to improve representation of underrepresented groups.
10. **Continuous Monitoring** – Regularly update and test models to prevent bias drift over time

Conclusion

1. AI loan systems can show bias in gender, age, occupation, or ethnicity.
2. Bias may cause unfair approvals or denials.
3. Use measurable criteria, exclude sensitive attributes, and apply fairness-aware methods.
4. Include human review, explainable AI, and regular monitoring.

5. These steps ensure **fair, transparent, and accountable** loan decisions.

Task Description#3 (Transparency)
• Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document
Expected Output#3
• Code with explanation
• Assess: Is the explanation understandable and correct?

Prompt-1: write function calculate the nth Fibonacci number using recursion

Prompt-2: Take input from user

CODE:

```
def fibonacci(n):
    """
    Calculate the nth Fibonacci number using recursion.

    Parameters:
    n (int): The position in the Fibonacci sequence (n >= 0)

    Returns:
    int: The nth Fibonacci number
    """
    # Base case: 0th Fibonacci number is 0
    if n == 0:
        return 0
    # Base case: 1st Fibonacci number is 1
    elif n == 1:
        return 1
    # Recursive case: sum of the two previous Fibonacci numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Take input from the user
n = int(input("Enter the position n to calculate the nth Fibonacci number: "))

# Validate input
if n < 0:
    print("Please enter a non-negative integer.")
else:
    result = fibonacci(n)
    print(f"The {n}th Fibonacci number is: {result}")
```

OUTPUT:

```
PS C:\Users\moham\Desktop\Python> & "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Enter the position n to calculate the nth Fibonacci number: 3
The 3th Fibonacci number is: 2
PS C:\Users\moham\Desktop\Python> & "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Enter the position n to calculate the nth Fibonacci number: 7
The 7th Fibonacci number is: 13
```

Explaination:

Function Definition: fibonacci(n) recursively calculates the nth Fibonacci number.

Base Cases:

- n == 0 ⟶ return 0
- n == 1 ⟶ return 1
  These stop the recursion.

Recursive Case:

- For n > 1, the function returns fibonacci(n-1) + fibonacci(n-2).
- This adds the previous two Fibonacci numbers.

User Input:

- input() asks the user for the position n.
- int() converts it to an integer.
- A check ensures n is non-negative.

Output:

- The program prints the nth Fibonacci number in a readable format.

Assessment

- **Understandable?** Yes, the code has clear comments and logical structure.
- **Correct?** Yes, it correctly calculates Fibonacci numbers for any non-negative integer n.
- **Limitation:** Recursive approach is inefficient for large n due to repeated calculations; can be optimized with memoization.

Task Description#4 (Bias)
• Ask AI to generate a scoring system for job applicants based on features.
Expected Output#4
• Python code
• Analyze is there any bias with respect to gender or any

Prompt-1: Give a Python function that scores job applicants based on experience, education, skills, and gender

CODE:

```python
# BIASED VERSION
def score_applicant(years, education, skills, gender):
    score = years * 5
    if education == "PhD":
        score += 20
    score += skills * 0.5
    if gender == "male":  # ⚠️ Gender bias
        score += 5
    return "Hire" if score > 50 else "No Hire"

print("Biased Decision 1:", score_applicant(8, "Masters", 30, "male"))  # Hire (due to +5 for male)
print("Biased Decision 2:", score_applicant(8, "Masters", 10, "female"))  # No Hire (bias visible)


# UNBIASED VERSION
def score_applicant(years, education, skills):
    score = years * 5
    if education == "PhD":
        score += 20
    score += skills * 0.5
    return "Hire" if score > 50 else "No Hire"

print("Unbiased Decision 1:", score_applicant(8, "Masters", 30))  # Hire
print("Unbiased Decision 2:", score_applicant(7, "Bachelors", 30))  # No Hire
```

OUTPUT:

```
PS C:\Users\moham\Desktop\Python> & "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Biased Decision 1: Hire
Biased Decision 2: No Hire
Unbiased Decision 1: Hire
Unbiased Decision 2: No Hire
```

Biased Decision 1 & 2: The AI may have made hiring decisions influenced by irrelevant factors like gender, age, or background. For example, if two candidates had similar qualifications but were treated differently based on gender, it shows discrimination — which is AI bias.

Unbiased Decision 1 & 2: When the model makes decisions purely based on skills, experience, and performance, it is considered fair and ethical

Task Description#5: Write a Python function to greet users by name and gender (Mr. or Ms.). Then make it gender neutral

Expected Output#5: Regenerate code that includes gender-neutral also

Prompt: Generate a python code that greet users by name and gender that make gender neutral

CODE:

```python
def greet_user():
    name = input("Enter your name: ")
    gender = input("Enter your gender (male/female/non-binary/prefer not to say): ").lower()

    # Assign titles based on gender
    if gender == "male":
        title = "Mr."
    elif gender == "female":
        title = "Ms."
    elif gender in ["non-binary", "prefer not to say", "other"]:
        title = "Mx."
    else:
        title = ""  # fallback if user types something unexpected

    # Greeting message
    print(f"Hello {title} {name}!")

# Run the function
greet_user()
```

OUTPUT

```
PS C:\Users\moham\Desktop\Python> & "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Enter your name: Mohammed Nizamuddin
Enter your gender (male/female/non-binary/prefer not to say): male
Hello Mr. Mohammed Nizamuddin!
```