

Vyshnavi Parisha

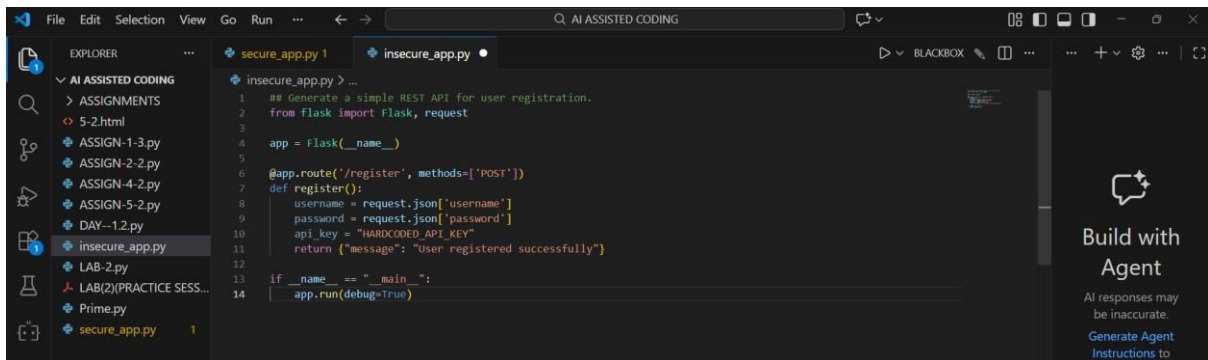
2403a51l34

B-52

Lab 5: Ethical Foundations – Responsible AI Coding Practices

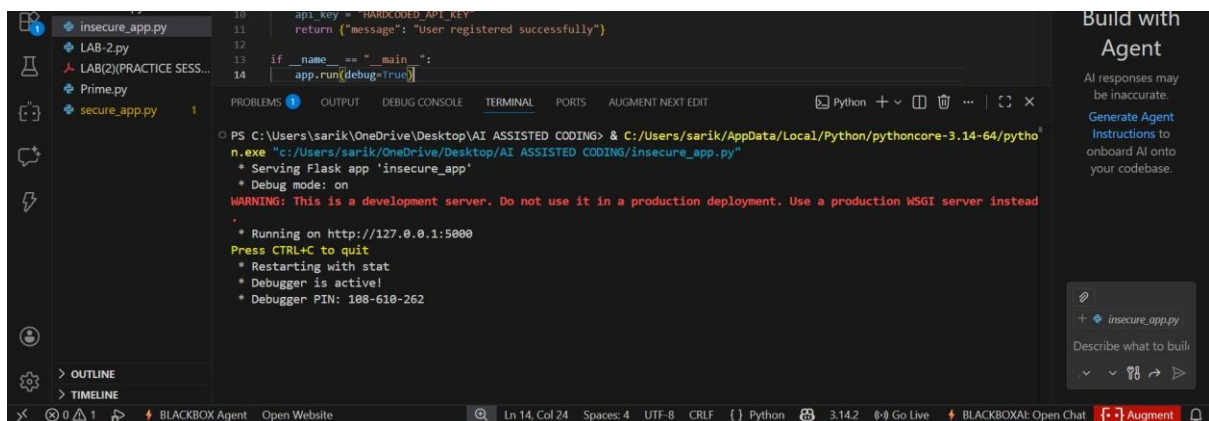
Task Description – 1: Secure API Usage

Prompt: Generate a simple REST API for user registration.

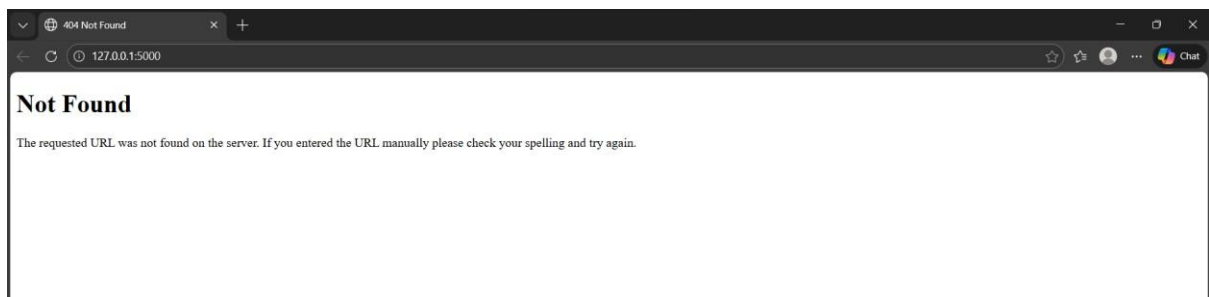


```
1  ## Generate a simple REST API for user registration.
2  from flask import Flask, request
3
4  app = Flask(__name__)
5
6  @app.route('/register', methods=['POST'])
7  def register():
8      username = request.json["username"]
9      password = request.json["password"]
10     api_key = "HARDCODED_API_KEY"
11     return {"message": "User registered successfully"}
12
13 if __name__ == "__main__":
14     app.run(debug=True)
```

OUTPUT:



```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "C:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/insecure_app.py"
* Serving Flask app 'insecure_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 108-610-262
```



The screenshot shows a VS Code editor with a file explorer on the left containing files like DAY-1.2.py, insecure_app.py, LAB-2.py, LAB(2)(PRACTICE SESS..., Prime.py, and secure_app.py. The main editor displays the code for insecure_app.py, which has a hardcoded API key 'HARDCODED_API_KEY'. The terminal at the bottom shows the command to run the application and the output, which includes a 404 error for the root path (/).

```

9 password = request.json['password']
10 api_key = "HARDCODED_API_KEY"
11 return {"message": "User registered successfully"}
12
13 if __name__ == "__main__":
14     app.run(debug=True)

```

```

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/insecure_app.py"
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 188-618-262
127.0.0.1 - - [28/Jan/2026 21:46:17] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [28/Jan/2026 21:46:17] "GET /favicon.ico HTTP/1.1" 404 -

```

Explanation: You got 404 error because your Flask app does not have a home (/) route, so the browser cannot find that page.

Identified Security Flaws:

1. API key is **hardcoded**, exposing sensitive credentials
2. No authentication or authorization mechanism
3. No input validation (password strength, missing fields)
4. Password stored/used in **plain text**
5. No token-based access control

Corrected Secure Version (Token-Based Authentication):

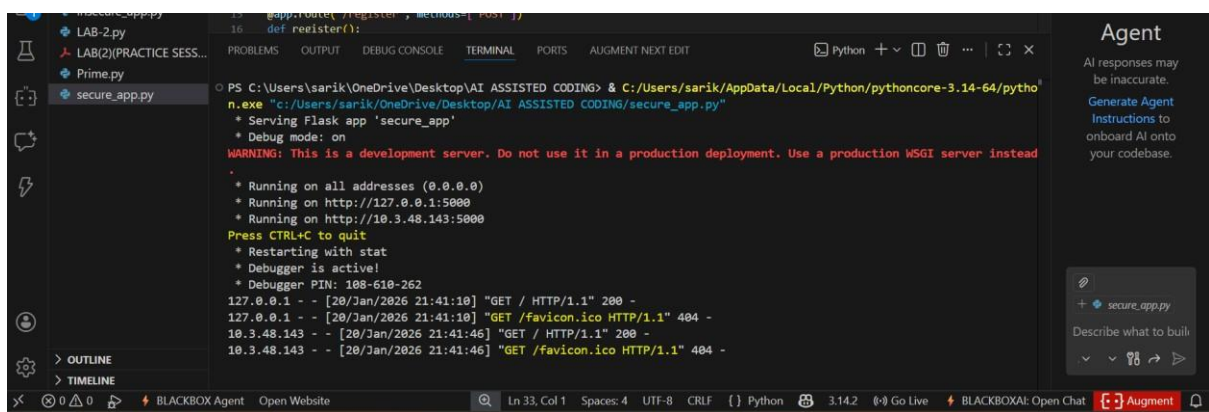
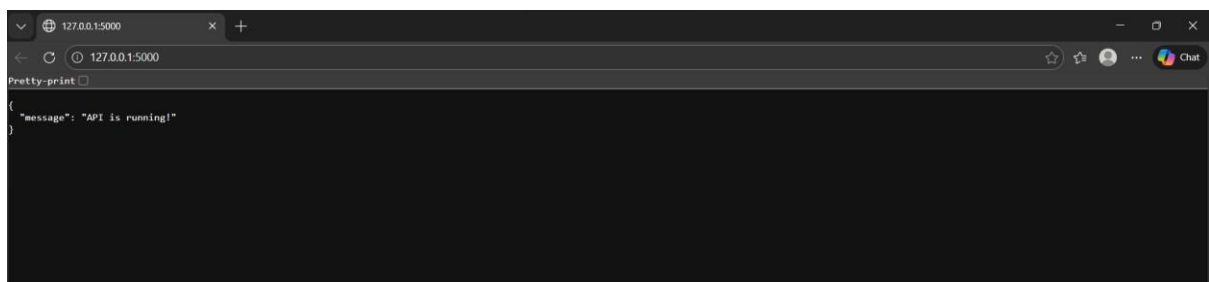
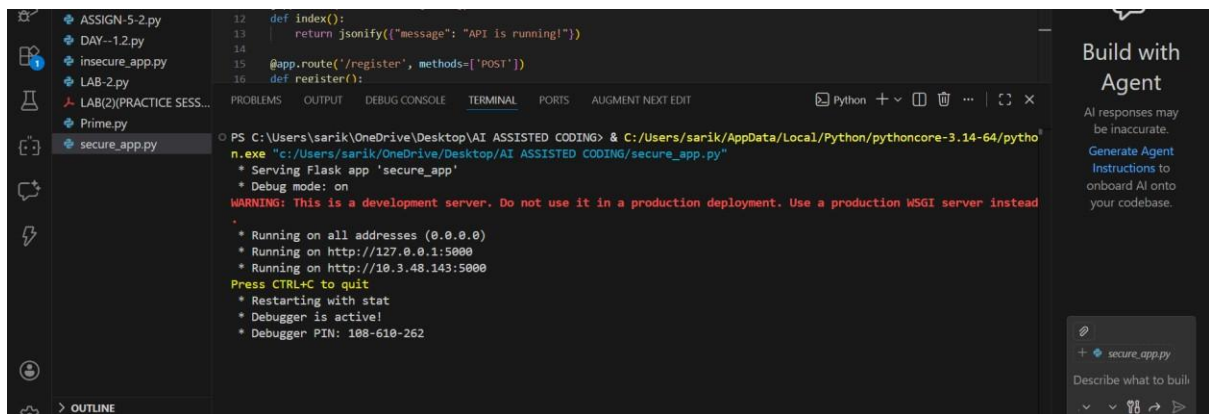
The screenshot shows a VS Code editor with a file explorer on the left containing files like ASSIGN-1-3.py, ASSIGN-2-2.py, ASSIGN-4-2.py, ASSIGN-5-2.py, DAY-1.2.py, insecure_app.py, LAB-2.py, LAB(2)(PRACTICE SESS..., Prime.py, and secure_app.py. The main editor displays the code for secure_app.py, which implements token-based authentication using JWT. The terminal at the bottom shows the command to run the application and the output, which includes a 404 error for the root path (/).

```

1  ## Secure API (Corrected - Token-Based Authentication)
2  > from flask import Flask, request, jsonify
3
4  app = Flask(__name__)
5  app.config['SECRET_KEY'] = os.getenv("SECRET_KEY", "mysecretkey")
6
7  @app.route('/', methods=['GET'])
8  def index():
9      return jsonify({"message": "API is running!"})
10
11 @app.route('/register', methods=['POST'])
12 def register():
13     data = request.get_json()
14     if not data or not data.get('username') or not data.get('password'):
15         return jsonify({"error": "Invalid input"}), 400
16     hashed_password = generate_password_hash(data['password'])
17     token = jwt.encode(
18         {
19             'user': data['username'],
20             'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=1)
21         },
22         app.config['SECRET_KEY'],
23         algorithm="HS256"
24     )
25     return jsonify({"token": token})
26
27 if __name__ == "__main__":
28     app.run(debug=True, host="0.0.0.0", port=5000)

```

OUTPUT:



Observations: The initial API code is insecure because it uses a hardcoded API key and does not protect user data. The corrected version improves security by validating inputs, hashing passwords, and using token-based authentication for safer access control.

Task Description – 2: Fair Decision Logic

Prompt: Generate a scholarship eligibility checker based on academic score, family income, and location.

AI-Generated Code:

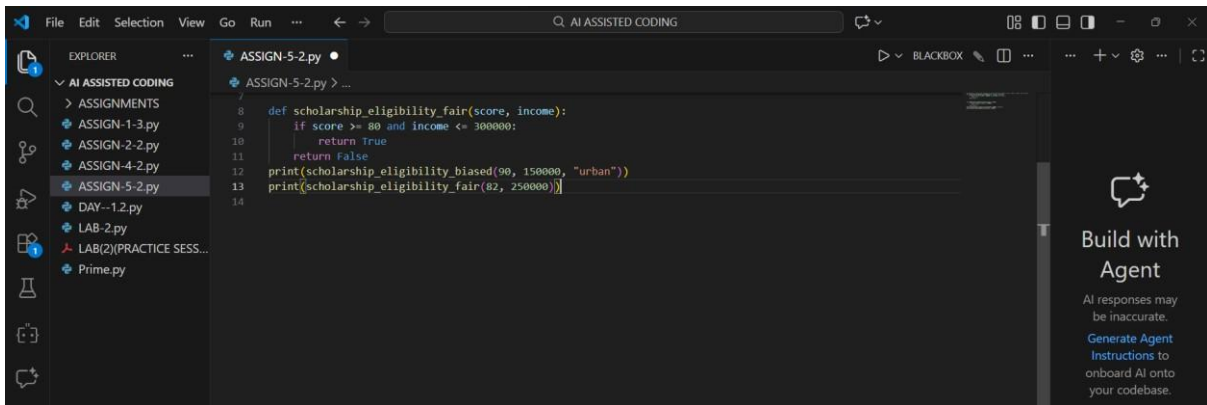


The screenshot shows a Python file named `ASSIGN-5-2.py` in VS Code. The code defines a function `scholarship_eligibility_biased` that checks for eligibility based on score, income, and location. The logic is biased towards urban students.

```
1  ## Generate a scholarship eligibility checker based on academic score, family income, and location.
2  def scholarship_eligibility_biased(score, income, location):
3      if score > 85 and income < 200000 and location == "urban":
4          return True
5      return False
6
7
```

Observations:

1. The logic unfairly favors urban students
 2. Rural or semi-urban students are excluded
 3. No flexibility or weighted scoring approach
- Improved Version:**

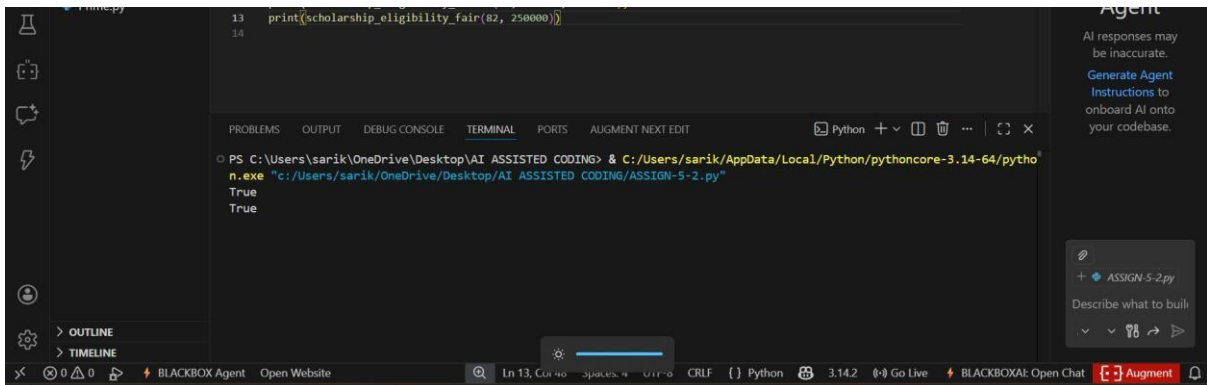


The screenshot shows the same file `ASSIGN-5-2.py` but with an improved, fairer function `scholarship_eligibility_fair`. It uses a weighted scoring system based on score and income, and includes a comment about the bias in the original function.

```
8  def scholarship_eligibility_fair(score, income):
9      if score >= 80 and income <= 300000:
10         return True
11     return False
12
13     print(scholarship_eligibility_biased(90, 150000, "urban"))
14     print(scholarship_eligibility_fair(82, 250000))

```

OUTPUT:



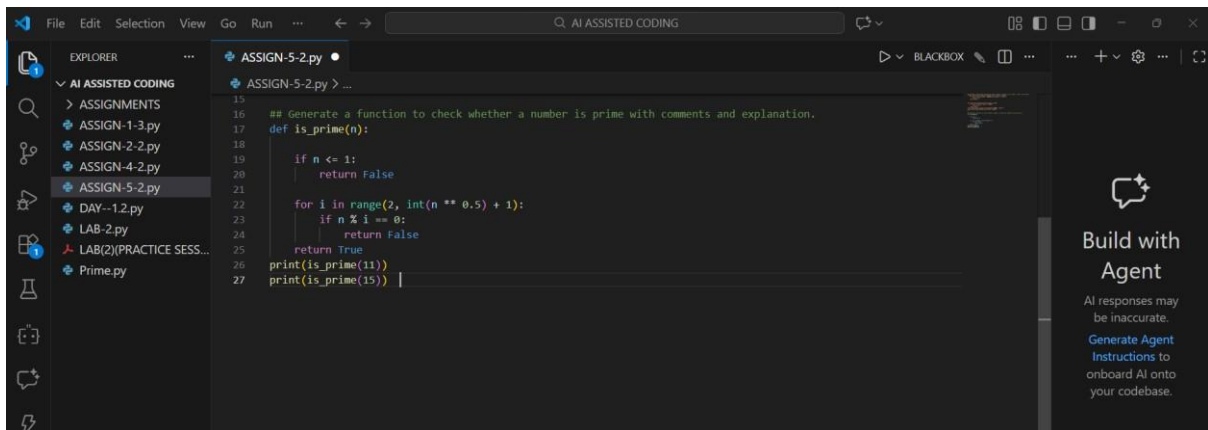
The screenshot shows the output of the improved function in the terminal. The output is:

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-5-2.py"
True
True
```

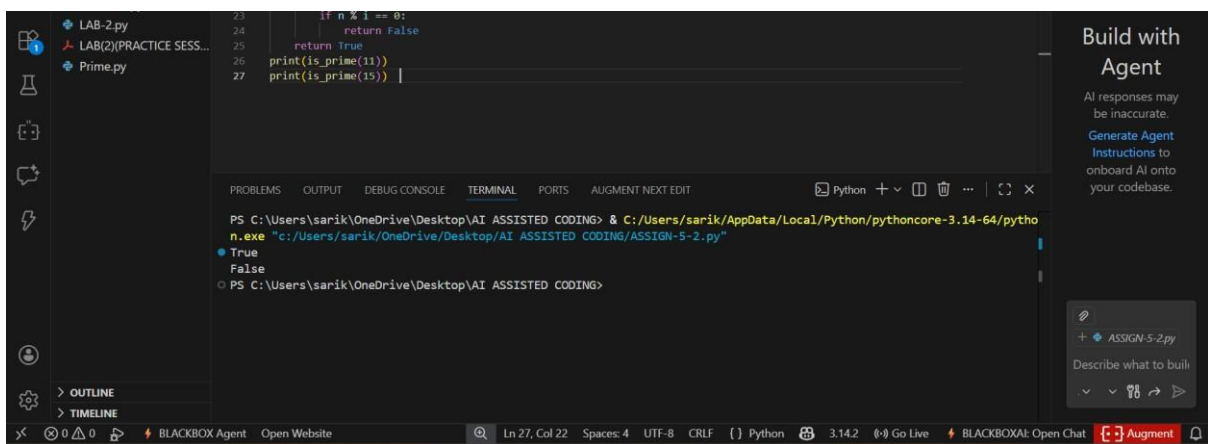
Explanation: The original logic introduces geographic bias by favoring urban students. Location should not be a deciding factor unless justified by policy. A fair system focuses on merit and economic need. Weighted or threshold-based criteria help ensure equitable access.

Task Description – 3: Explainability

Prompt: Generate a function to check whether a number is prime with comments and explanation.



OUTPUT:

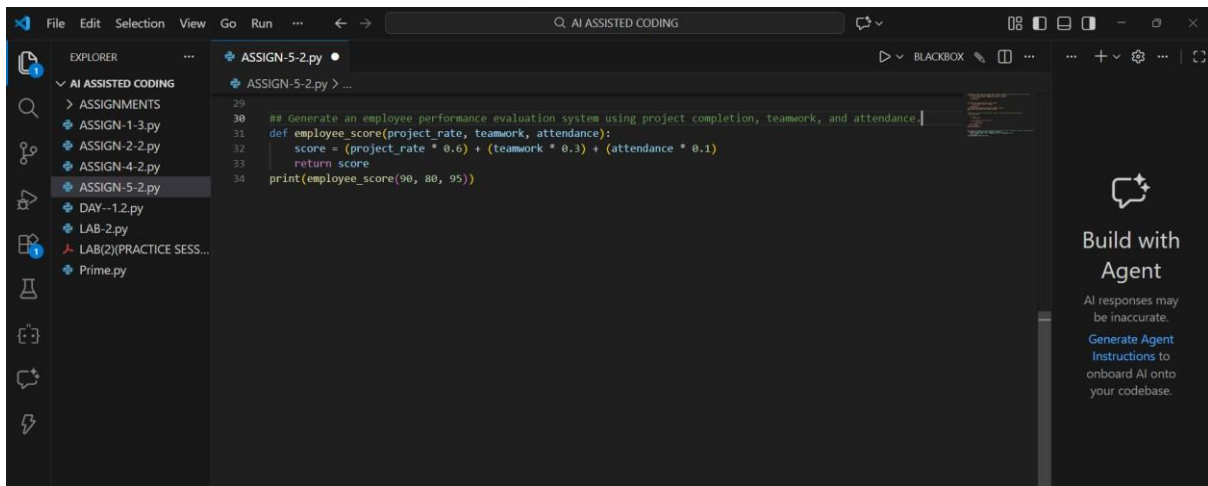


Explanation: The function first checks if the number is greater than 1. It then tests divisibility from 2 up to the square root of the number to reduce computation. If any divisor is found, the number is not prime; otherwise, it is prime.

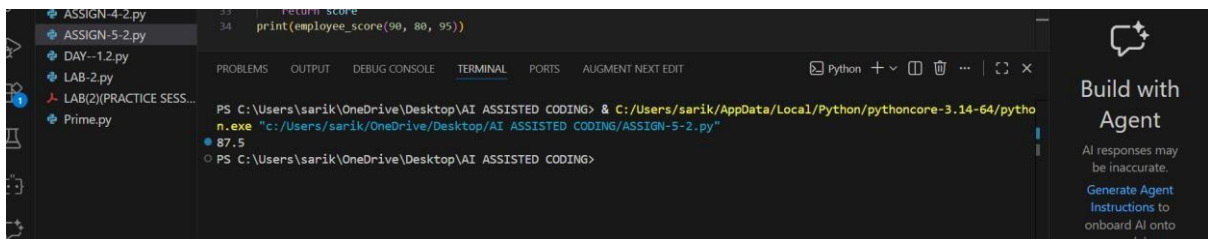
The explanation is clear, correct, and efficient. Inline comments improve readability and help beginners understand the logic easily.

Task Description – 4: Ethical Scoring System

Prompt: Generate an employee performance evaluation system using project completion, teamwork, and attendance.



OUTPUT:



Observations:

1. Heavy weight on project completion may disadvantage collaborative roles
2. Attendance weighting may penalize employees with health or caregiving needs
3. Teamwork score depends on subjective evaluation

The criteria are reasonable but require transparency and flexibility. Ethical systems should allow contextual review and avoid over-reliance on single metrics.

Task Description – 5: Accessibility and Inclusiveness Prompt:

Generate a user feedback form application.


```
5-2.html
File Edit View

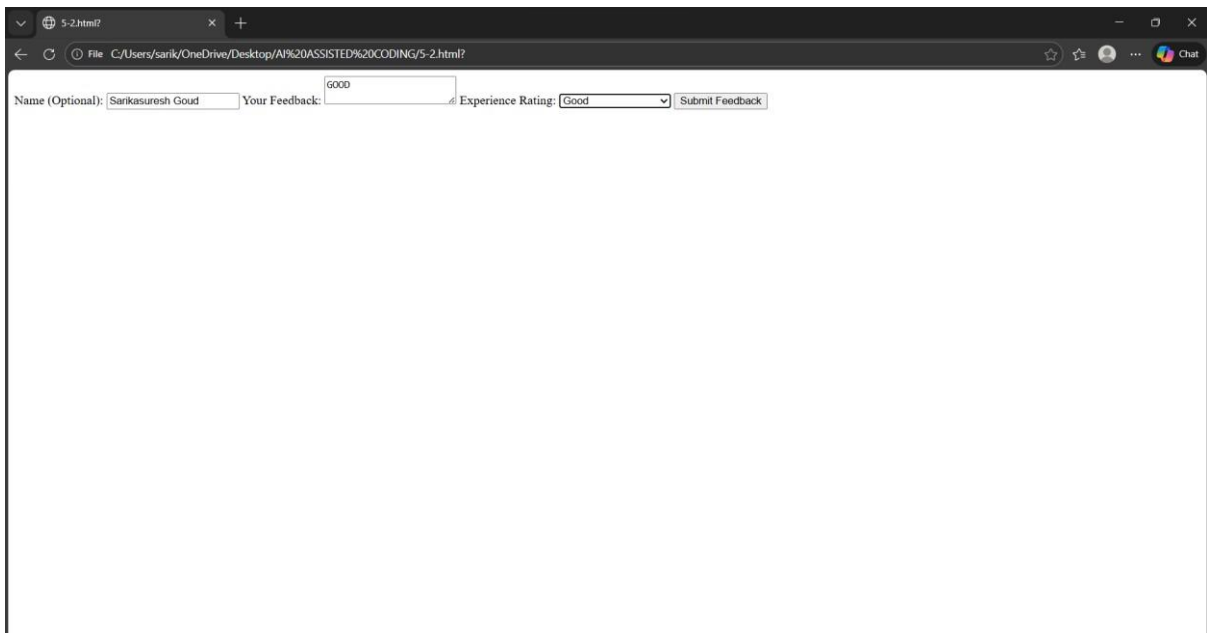
<form aria-label="User Feedback Form">
  <label for="name">Name (Optional):</label>
  <input type="text" id="name" aria-required="false">

  <label for="feedback">Your Feedback:</label>
  <textarea id="feedback" aria-required="true"></textarea>

  <label for="rating">Experience Rating:</label>
  <select id="rating">
    <option>Very Good</option>
    <option>Good</option>
    <option>Neutral</option>
    <option>Needs Improvement</option>
  </select>

  <button type="submit">Submit Feedback</button>
</form>
```

OUTPUT:



Observations: The feedback form uses neutral and inclusive language to avoid exclusion of any user group. Accessibility is enhanced through ARIA labels, optional fields, and simple input options for diverse users.