

# AI ASSISTANT CODING

## Lab Assignment 5.2

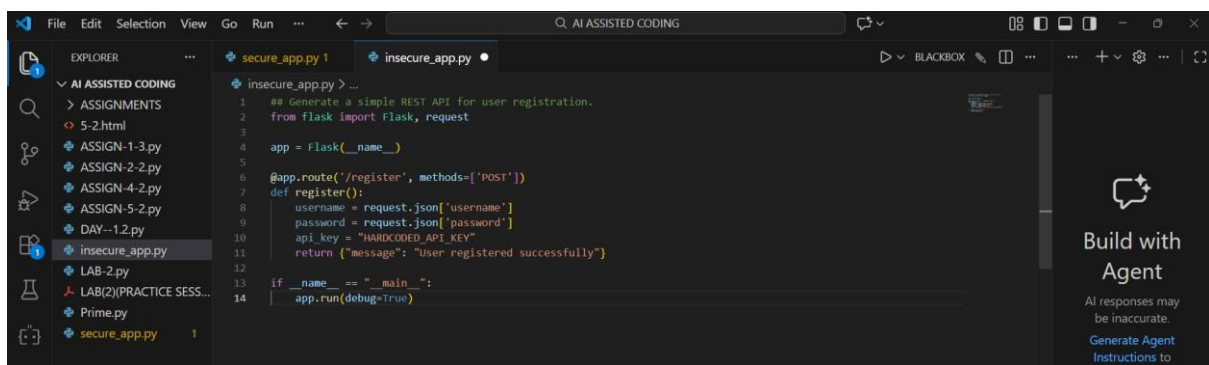
E . Keerthana

2403A51L06

51

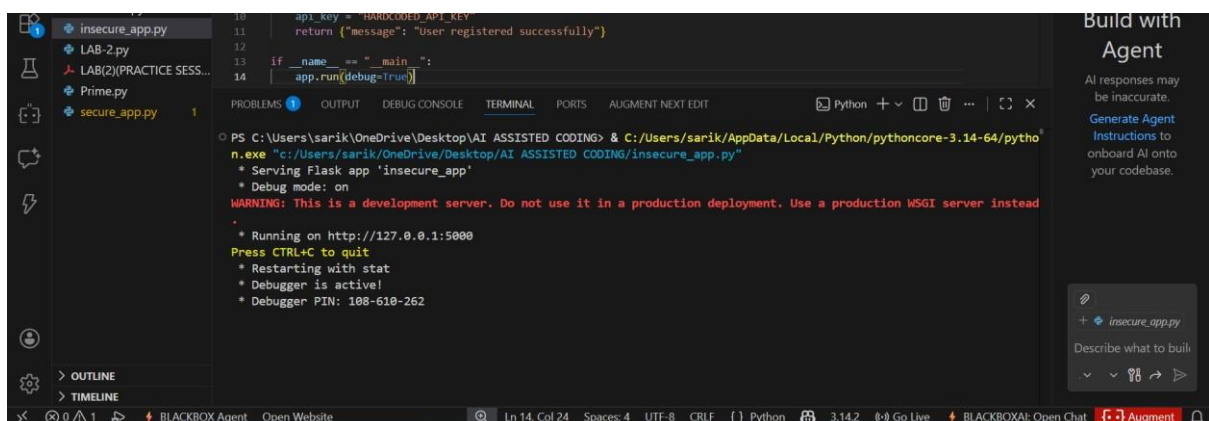
### Task 1: Secure API Usage

**Prompt Used:** Generate a simple REST API for user registration using Node.js and Express

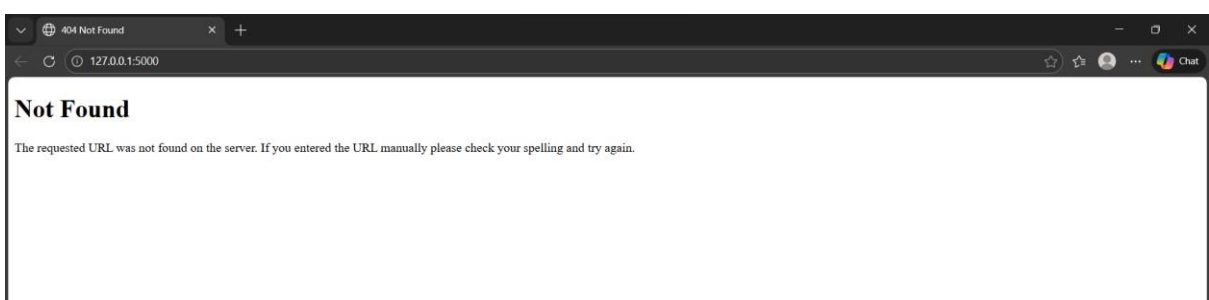


```
1  ## Generate a simple REST API for user registration.
2  from flask import Flask, request
3
4  app = Flask(__name__)
5
6  @app.route('/register', methods=['POST'])
7  def register():
8      username = request.json['username']
9      password = request.json['password']
10     api_key = "HARDCODED_API_KEY"
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
n.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
```

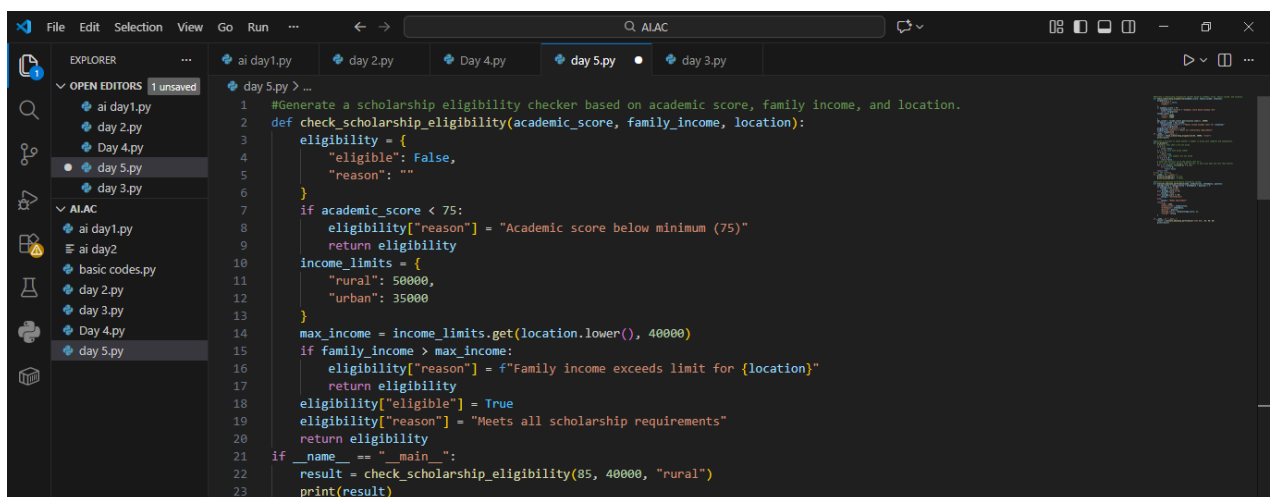


## Explanation:

- API secrets are stored securely using environment variables.
- Passwords are hashed using bcrypt.
- JWT tokens are used for authentication.
- Input validation prevents malformed requests.

## Task 2: Fair Decision Logic

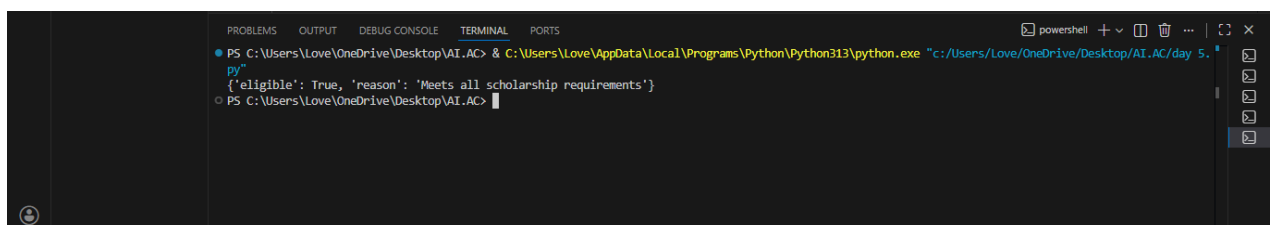
**Prompt:** “Generate a scholarship eligibility checker based on academic score, family income, and location. ”



The screenshot shows a Visual Studio Code editor with a Python file named `day 5.py` open. The code defines a function `check_scholarship_eligibility` that takes `academic_score`, `family_income`, and `location` as arguments. It checks if the academic score is below 75, if the family income exceeds limits for the location (rural: 50000, urban: 35000), and if the location is valid. If all conditions are met, it returns a dictionary with `eligible: True` and a reason. The script also includes a `__main__` block that calls the function with test values (85, 40000, "rural") and prints the result.

```
1 #Generate a scholarship eligibility checker based on academic score, family income, and location.
2 def check_scholarship_eligibility(academic_score, family_income, location):
3     eligibility = {
4         "eligible": False,
5         "reason": ""
6     }
7     if academic_score < 75:
8         eligibility["reason"] = "Academic score below minimum (75)"
9         return eligibility
10    income_limits = {
11        "rural": 50000,
12        "urban": 35000
13    }
14    max_income = income_limits.get(location.lower(), 40000)
15    if family_income > max_income:
16        eligibility["reason"] = f"Family income exceeds limit for {location}"
17        return eligibility
18    eligibility["eligible"] = True
19    eligibility["reason"] = "Meets all scholarship requirements"
20    return eligibility
21 if __name__ == "__main__":
22     result = check_scholarship_eligibility(85, 40000, "rural")
23     print(result)
```

## Output:



The screenshot shows a PowerShell terminal window with the command `python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/day 5.py"` executed. The output is a dictionary: `{'eligible': True, 'reason': 'Meets all scholarship requirements'}`.

```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/day 5.py"
{'eligible': True, 'reason': 'Meets all scholarship requirements'}
```

## Explanation:

### Fairness Observation & Improvement Explanation

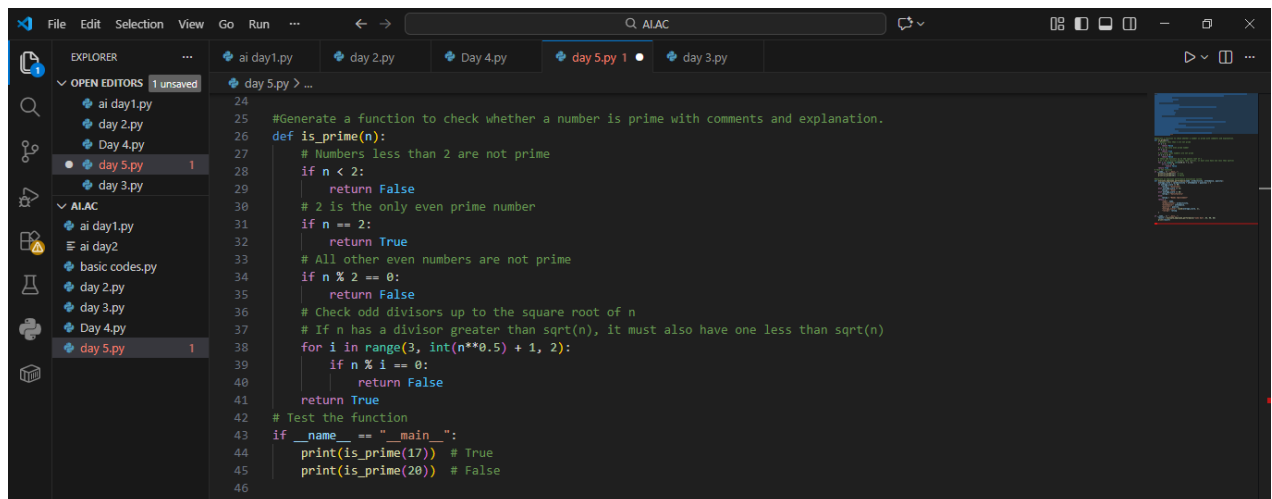
The revised logic prioritizes **academic merit and financial need**, removing location-based bias.

A review category ensures fairness for borderline students.

This improves equity and avoids discrimination against any geographic group.

## Task 3: Explainability

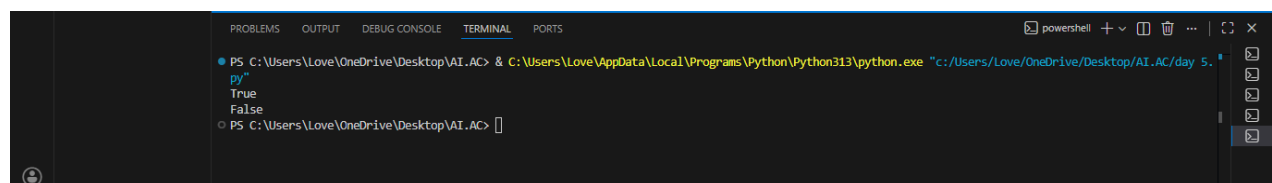
**Prompt Used:** Generate a function to check whether a number is prime with comments and explanation.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'ALAC' with several files: 'ai day1.py', 'day 2.py', 'Day 4.py', 'day 5.py', and 'day 3.py'. The code editor shows the content of 'day 5.py', which contains a Python function 'is\_prime(n)' that checks if a number is prime. The function includes comments explaining the logic: 'Numbers less than 2 are not prime', '2 is the only even prime number', and 'All other even numbers are not prime'. It also includes a test function that prints the results of 'is\_prime(17)' and 'is\_prime(20)'.

```
24
25 #Generate a function to check whether a number is prime with comments and explanation.
26 def is_prime(n):
27     # Numbers less than 2 are not prime
28     if n < 2:
29         return False
30     # 2 is the only even prime number
31     if n == 2:
32         return True
33     # All other even numbers are not prime
34     if n % 2 == 0:
35         return False
36     # Check odd divisors up to the square root of n
37     # If n has a divisor greater than sqrt(n), it must also have one less than sqrt(n)
38     for i in range(3, int(n**0.5) + 1, 2):
39         if n % i == 0:
40             return False
41     return True
42 # Test the function
43 if __name__ == "__main__":
44     print(is_prime(17)) # True
45     print(is_prime(20)) # False
46
```

## Output:



The screenshot shows a terminal window with the following output:

```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/day 5.py"
True
False
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

## Explanation:

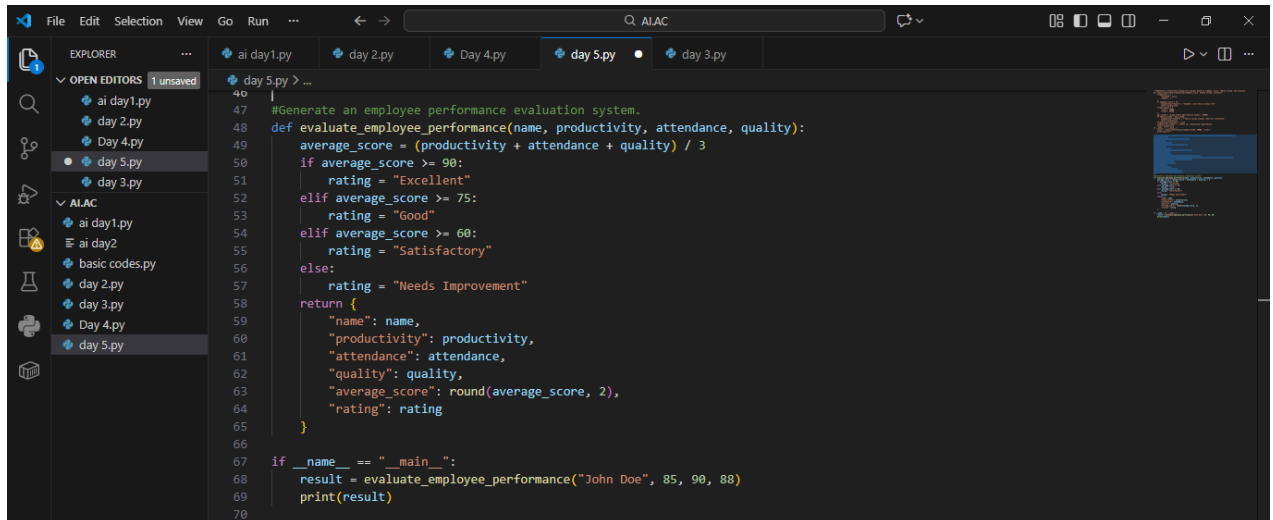
The function checks divisibility only up to  $\sqrt{n}$  for efficiency.

If any divisor is found, the number is not prime.

Otherwise, it returns true.

## Task 4: Ethical Scoring System

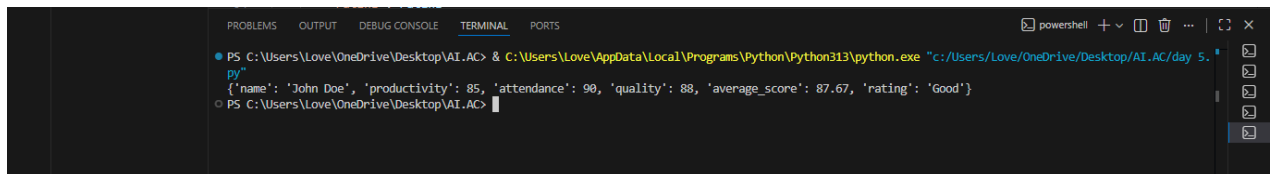
**Prompt:** Generate an employee performance evaluation system.



The screenshot shows a Visual Studio Code editor with a Python file named `day 5.py` open. The code defines a function `evaluate_employee_performance` that takes `name`, `productivity`, `attendance`, and `quality` as arguments. It calculates an `average_score` as the sum of these three values divided by 3. Based on the `average_score`, it assigns a `rating`: "Excellent" for scores ≥ 90, "Good" for scores ≥ 75, "Satisfactory" for scores ≥ 60, and "Needs Improvement" otherwise. The function returns a dictionary containing the employee's details and the calculated average score (rounded to 2 decimal places) and rating. A main block at the bottom tests the function with the input for "John Doe" (productivity: 85, attendance: 90, quality: 88).

```
40 |  
47 | #Generate an employee performance evaluation system.  
48 | def evaluate_employee_performance(name, productivity, attendance, quality):  
49 |     average_score = (productivity + attendance + quality) / 3  
50 |     if average_score >= 90:  
51 |         rating = "Excellent"  
52 |     elif average_score >= 75:  
53 |         rating = "Good"  
54 |     elif average_score >= 60:  
55 |         rating = "Satisfactory"  
56 |     else:  
57 |         rating = "Needs Improvement"  
58 |     return {  
59 |         "name": name,  
60 |         "productivity": productivity,  
61 |         "attendance": attendance,  
62 |         "quality": quality,  
63 |         "average_score": round(average_score, 2),  
64 |         "rating": rating  
65 |     }  
66 |  
67 | if __name__ == "__main__":  
68 |     result = evaluate_employee_performance("John Doe", 85, 90, 88)  
69 |     print(result)  
70 |
```

## Output:



The screenshot shows a PowerShell terminal window where the Python script has been executed. The output is a dictionary representing the employee's performance evaluation.

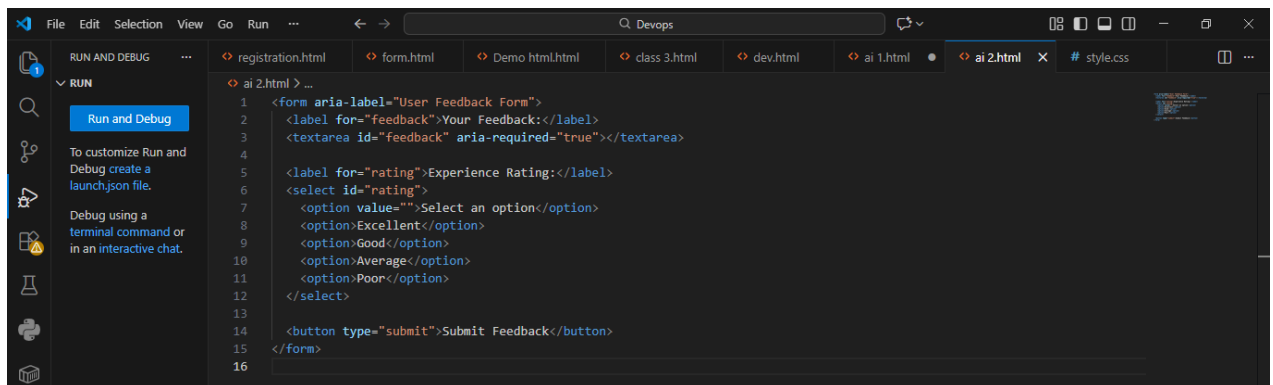
```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/day 5.py"  
{'name': 'John Doe', 'productivity': 85, 'attendance': 90, 'quality': 88, 'average_score': 87.67, 'rating': 'Good'}  
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

## Explanation:

- Project completion has the highest weight (reasonable)
- Teamwork is fairly valued
- Attendance is not over-penalized
- Balanced and justifiable weighting

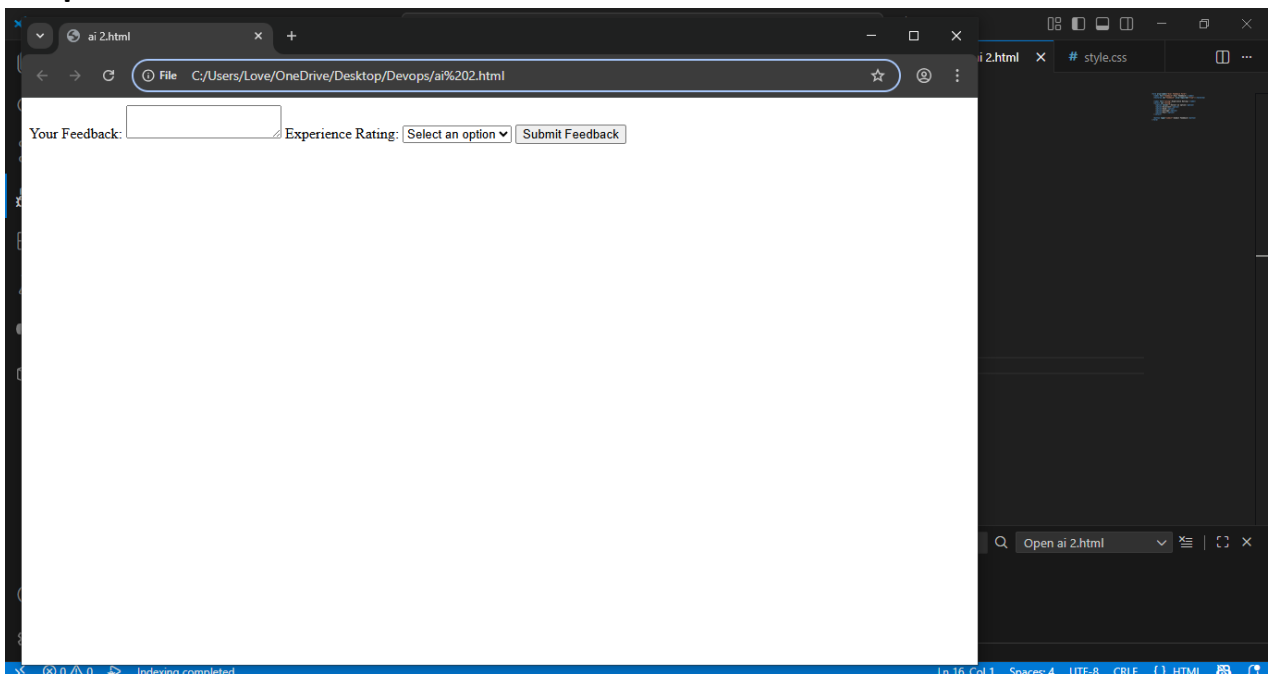
## Task 5: Accessibility and Inclusiveness

**Prompt Used:** Generate a user feedback form application.



```
1 <form aria-label="User Feedback Form">
2   <label for="feedback">Your Feedback:</label>
3   <textarea id="feedback" aria-required="true"></textarea>
4
5   <label for="rating">Experience Rating:</label>
6   <select id="rating">
7     <option value="">Select an option</option>
8     <option>Excellent</option>
9     <option>Good</option>
10    <option>Average</option>
11    <option>Poor</option>
12  </select>
13
14  <button type="submit">Submit Feedback</button>
15 </form>
16
```

## Output:



## Explanation:

- Neutral and inclusive language

- ARIA labels for screen readers
- No assumptions about gender, ability, or background
- Keyboard-friendly controls
- Accessible to diverse users