

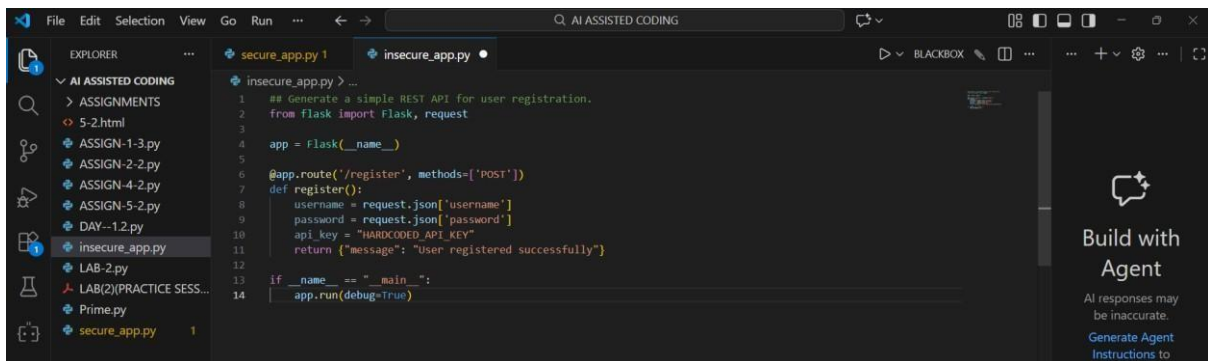
2403a51L40

B-52

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Task Description – 1: Secure API Usage

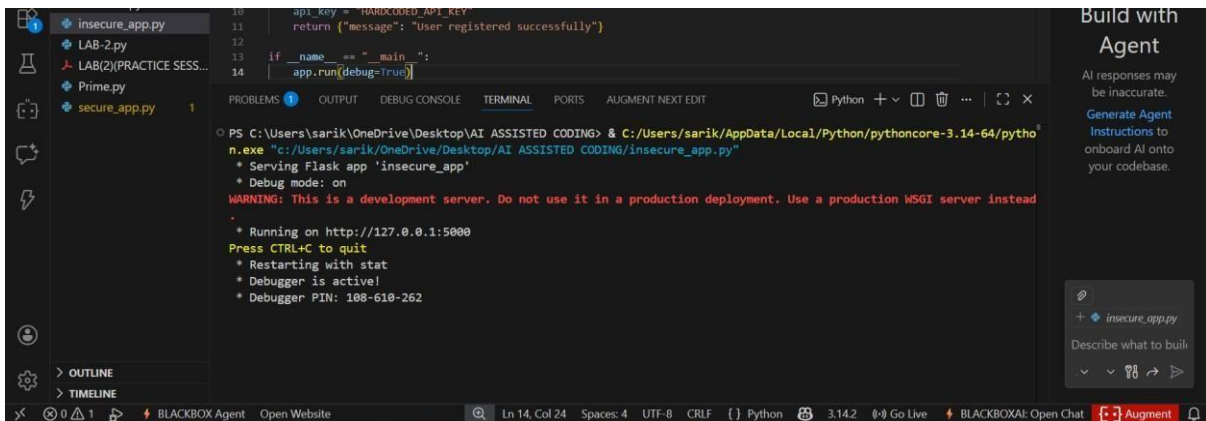
Prompt: Generate a simple REST API for user registration.



The screenshot shows the Visual Studio Code interface with the 'AI ASSISTED CODING' sidebar open. The main editor displays a Python file named 'insecure_app.py' containing the following code:

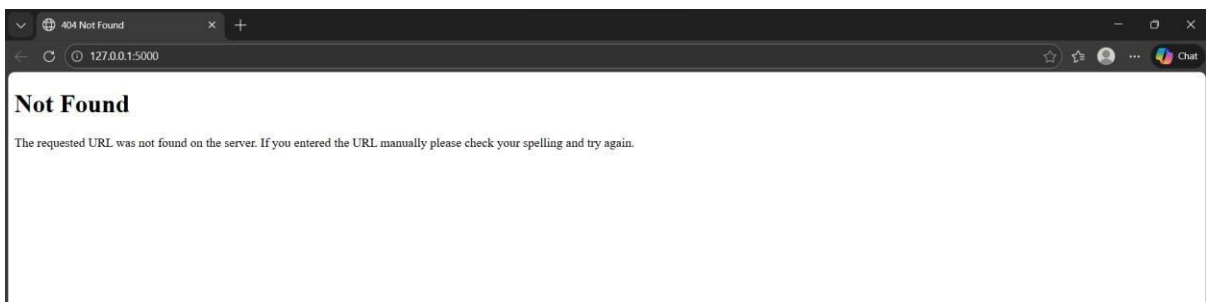
```
1 # Generate a simple REST API for user registration.
2 from flask import Flask, request
3
4 app = Flask(__name__)
5
6 @app.route('/register', methods=['POST'])
7 def register():
8     username = request.json['username']
9     password = request.json['password']
10    api_key = "HARDCODED_API_KEY"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

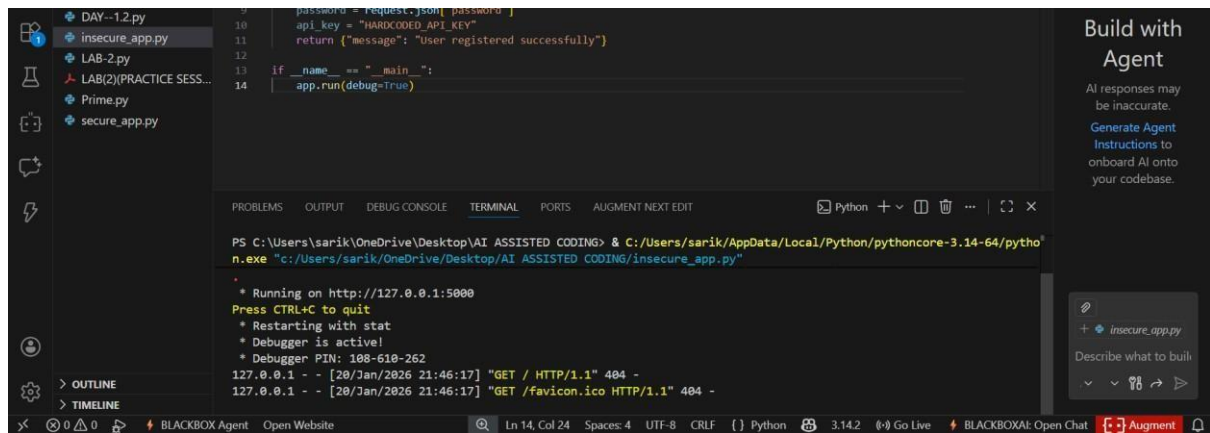
OUTPUT:



The screenshot shows the Visual Studio Code interface with the 'AI ASSISTED CODING' sidebar open. The main editor displays the output of the REST API, showing the following message:

```
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
```





```
password = request.json['password']
api_key = "HARDCODED_API_KEY"
return {"message": "User registered successfully"}

if __name__ == "__main__":
    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"

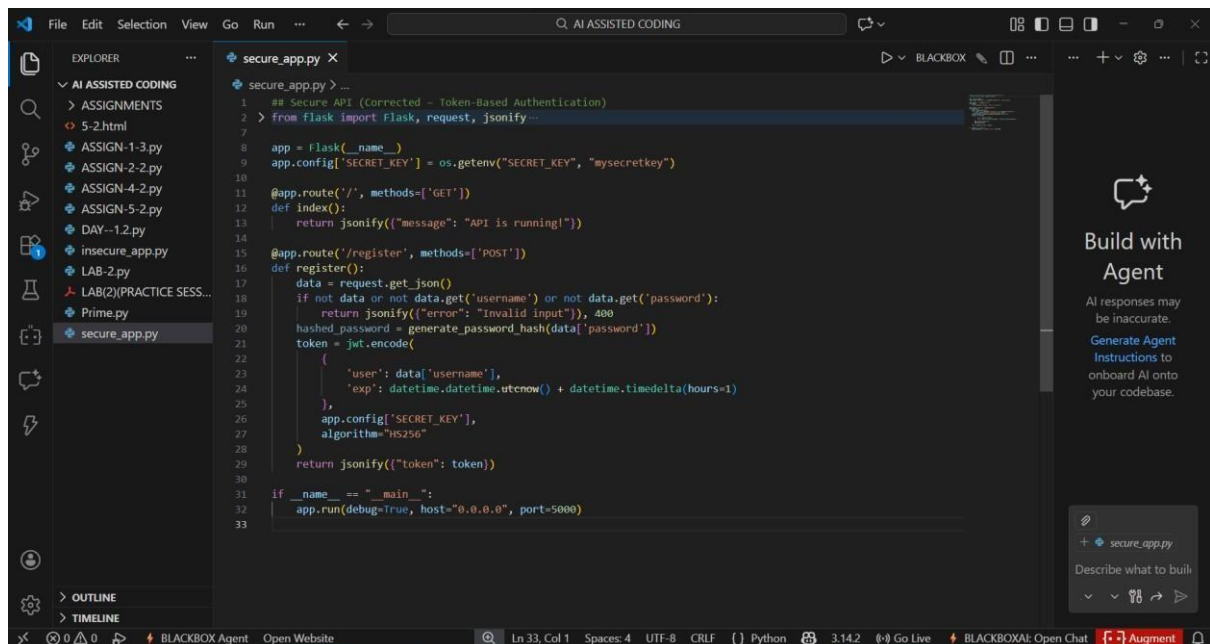
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 188-610-262
127.0.0.1 - - [20/Jan/2026 21:46:17] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [20/Jan/2026 21:46:17] "GET /favicon.ico HTTP/1.1" 404 -
```

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):



```
# Secure API (Corrected - Token-Based Authentication)
from flask import Flask, request, jsonify

app = Flask(__name__)
app.config['SECRET_KEY'] = os.getenv("SECRET_KEY", "mysecretkey")

@app.route('/', methods=['GET'])
def index():
    return jsonify({"message": "API is running!"})

@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    if not data or not data.get('username') or not data.get('password'):
        return jsonify({"error": "Invalid input"}), 400
    hashed_password = generate_password_hash(data['password'])
    token = jwt.encode(
        {
            'user': data['username'],
            'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=1)
        },
        app.config['SECRET_KEY'],
        algorithm="HS256"
    )
    return jsonify({"token": token})

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
```

OUTPUT:

The screenshot shows a VS Code editor with a file explorer on the left containing files like 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 insecure_app.py, which includes a simple index route and a register route. The terminal shows the command to run the application, the output of the Flask server, and a warning about using a development server.

```
def index():
    return jsonify({"message": "API is running!"})

@app.route('/register', methods=['POST'])
def register():
```

Terminal output:

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/pytho
n.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/secure_app.py"
* Serving Flask app 'secure_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 all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.3.48.143:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 108-610-262
```

The screenshot shows a web browser window with the address bar set to 127.0.0.1:5000. The page content displays a JSON response from the API: {"message": "API is running!"}.

The screenshot shows the VS Code editor with the file explorer on the left. The main editor displays the code for secure_app.py, which includes a more complex register route that validates inputs, hashes passwords, and uses token-based authentication. The terminal shows the command to run the application, the output of the Flask server, and a warning about using a development server. The bottom status bar indicates the application is running on 127.0.0.1:5000.

```
@app.route('/register', methods=['POST'])
def register():
    # Validate inputs
    if not request.json or not 'username' in request.json or not 'password' in request.json:
        return jsonify({'message': 'Invalid input'}), 400
    # Hash password
    password_hash = bcrypt.hashpw(request.json['password'].encode('utf-8'), bcrypt.gensalt())
    # Save to database
    db = get_db()
    cursor = db.cursor()
    cursor.execute('INSERT INTO users (username, password_hash) VALUES (%s, %s)',
                  (request.json['username'], password_hash))
    db.commit()
    cursor.close()
    return jsonify({'message': 'User registered successfully'}), 201
```

Terminal output:

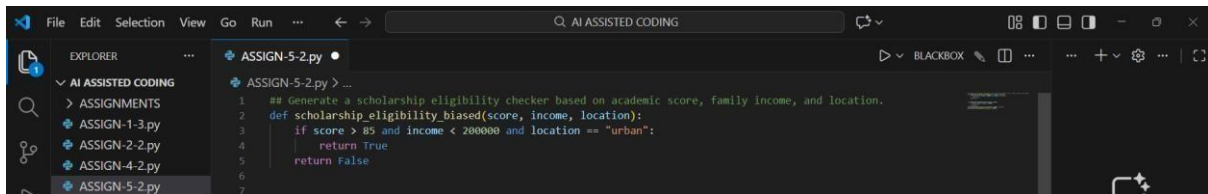
```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/pytho
n.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/secure_app.py"
* Serving Flask app 'secure_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.3.48.143:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 108-610-262
127.0.0.1 - - [20/Jan/2026 21:41:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Jan/2026 21:41:10] "GET /favicon.ico HTTP/1.1" 404 -
10.3.48.143 - - [20/Jan/2026 21:41:46] "GET / HTTP/1.1" 200 -
10.3.48.143 - - [20/Jan/2026 21:41:46] "GET /favicon.ico HTTP/1.1" 404 -
```

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:

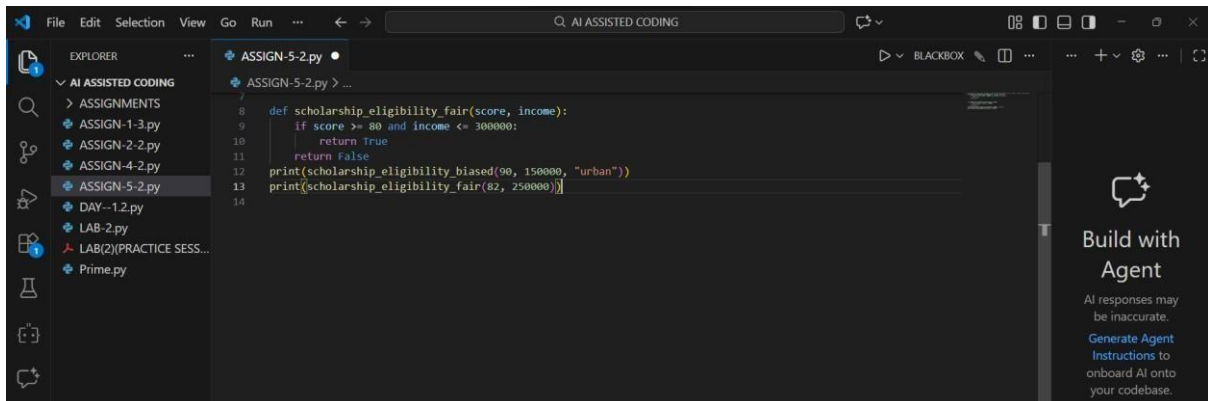


```
1  # Generate a scholarship eligibility checker based on academic score, family income, and location.
2  def scholarship_eligibility_biased(score, income, location):
3      if score > 85 and income < 200000 and location == "urban":
4          return True
5      return False
6
7
```

Observations:

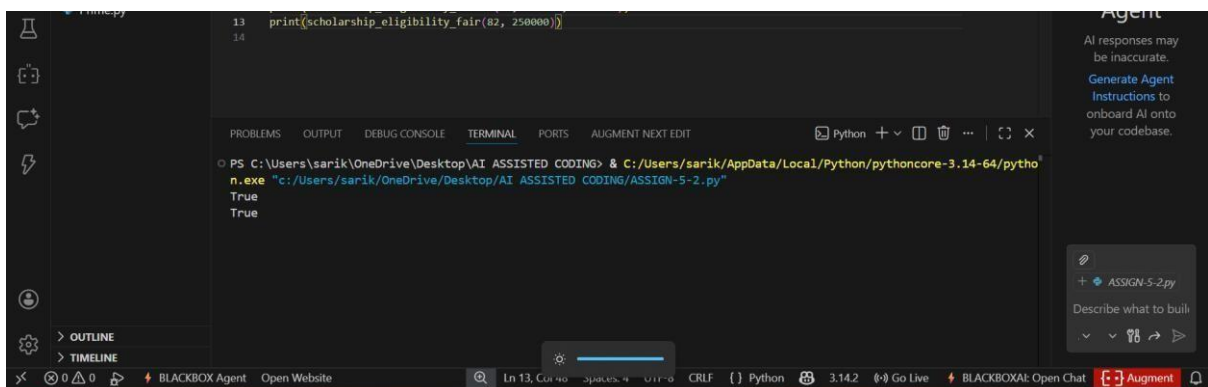
1. The logic unfairly favors urban students
2. Rural or semi-urban students are excluded
3. No flexibility or weighted scoring approach

Improved Version:



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

OUTPUT:

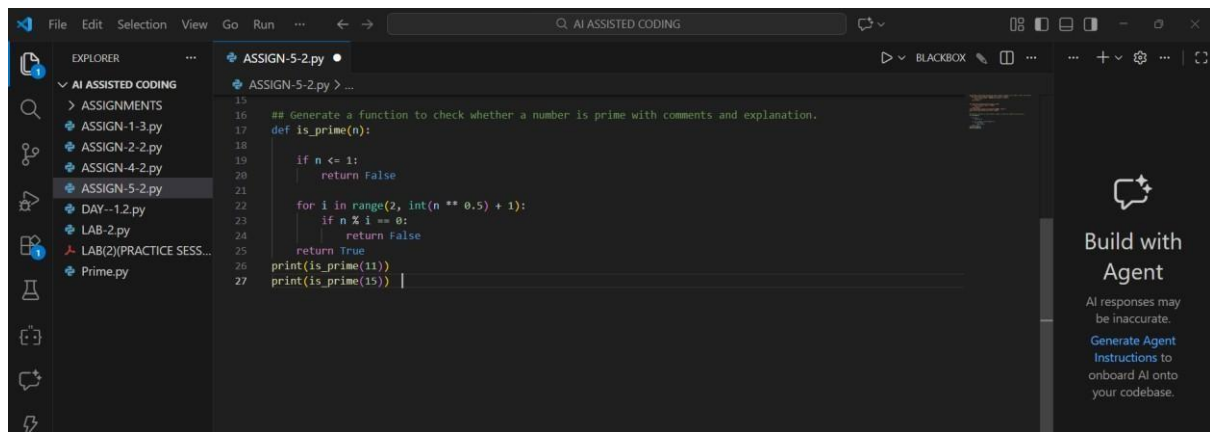


```
13     print(scholarship_eligibility_fair(82, 250000))
14
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & c:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-5-2.py"
True
True
```

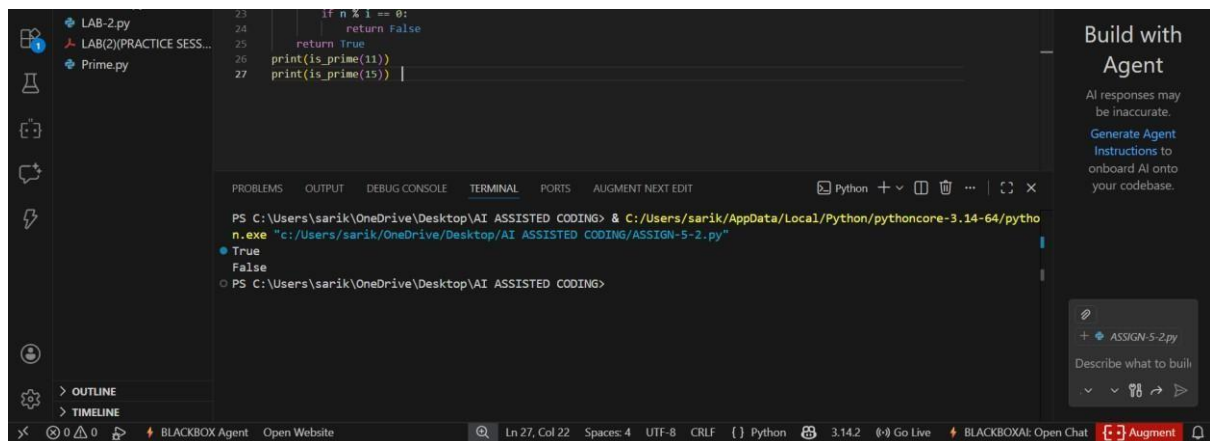
Explanation: The original logic introduces geographic bias by favoring urban students. Location should not be a deciding factor unless justified by policy. A fair system focuses on merit and economic need. Weighted or threshold-based criteria help ensure equitable access.

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.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files under 'AI ASSISTED CODING', including 'ASSIGN-5-2.py'. The main editor window displays the code for 'ASSIGN-5-2.py':

```
29
30 ## Generate an employee performance evaluation system using project completion, teamwork, and attendance
31 def employee_score(project_rate, teamwork, attendance):
32     score = (project_rate * 0.6) + (teamwork * 0.3) + (attendance * 0.1)
33     return score
34 print(employee_score(90, 80, 95))
```

On the right side, there is a 'Build with Agent' panel with a warning: 'AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.'

OUTPUT:

The screenshot shows the Visual Studio Code interface with the 'TERMINAL' panel active. The terminal displays the command to run the script and its 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/ASSIGN-5-2.py"
87.5
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

On the right side, there is a 'Build with Agent' panel with a warning: 'AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.'

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

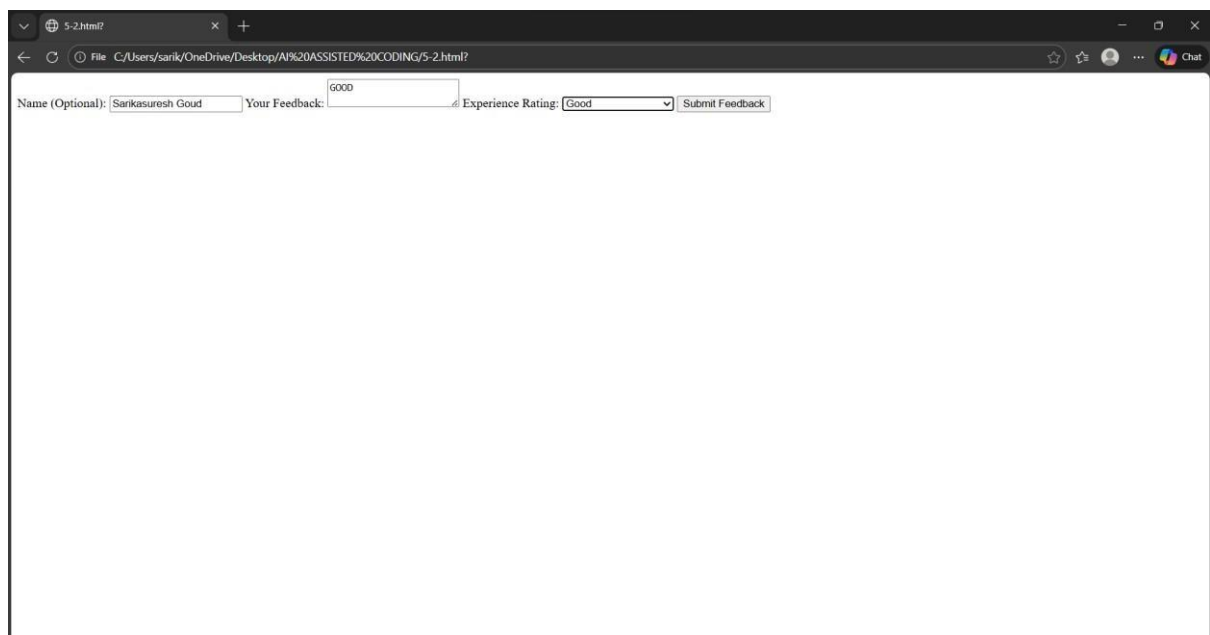
<form aria-label="User Feedback Form">
  <label for="name">Name (Optional):</label>
  <input type="text" id="name" aria-required="false">

  <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:



The screenshot shows a web browser window with the URL 'C:/Users/sarik/OneDrive/Desktop/AI%20ASSISTED%20CODING/5-2.html?'. The rendered form is as follows:

Name (Optional): Your Feedback: Experience Rating:

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.