

Bharath Samala

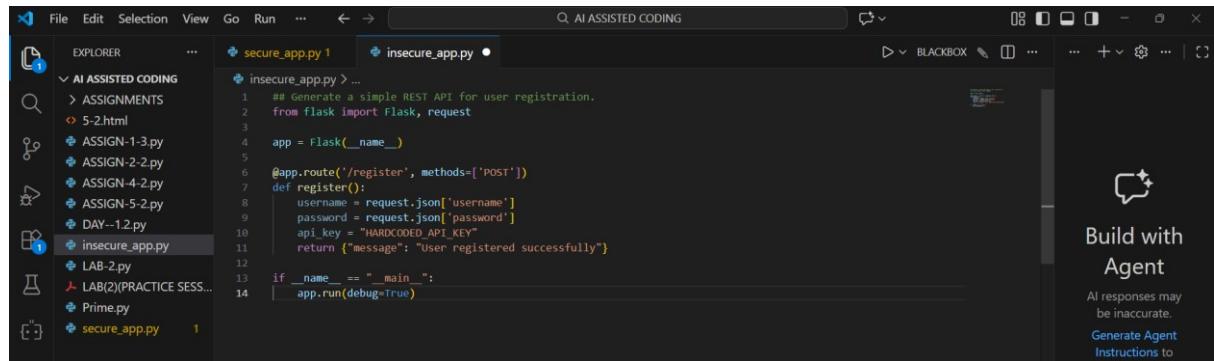
2403a51l17

B-51

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Task Description – 1: Secure API Usage

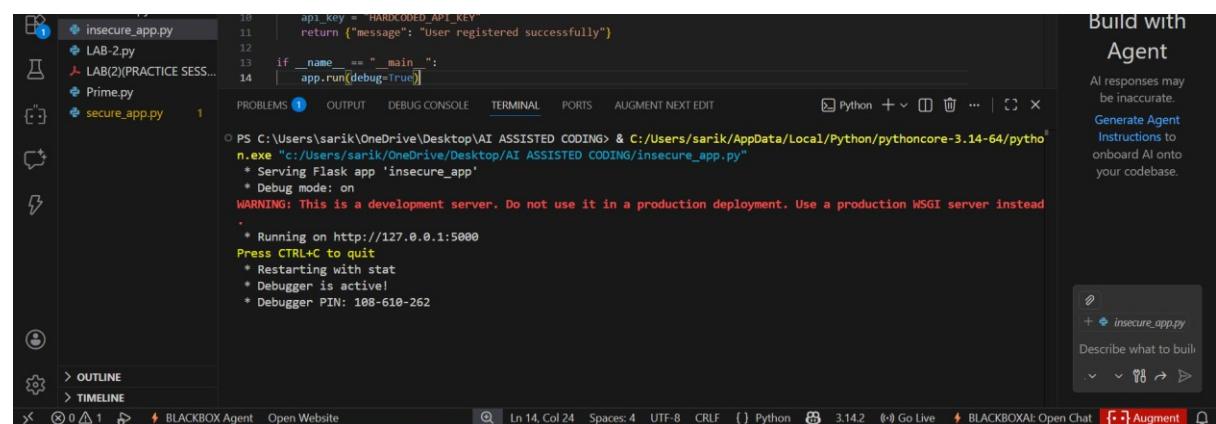
Prompt: Generate a simple REST API for user registration.



The screenshot shows a code editor interface with a dark theme. In the center is a code editor window displaying Python code for a REST API. The code defines a Flask application with a route for user registration. It includes a hard-coded API key and a warning about using a development server in production. The code editor has a sidebar on the left showing a file tree with various Python files and a terminal tab at the bottom.

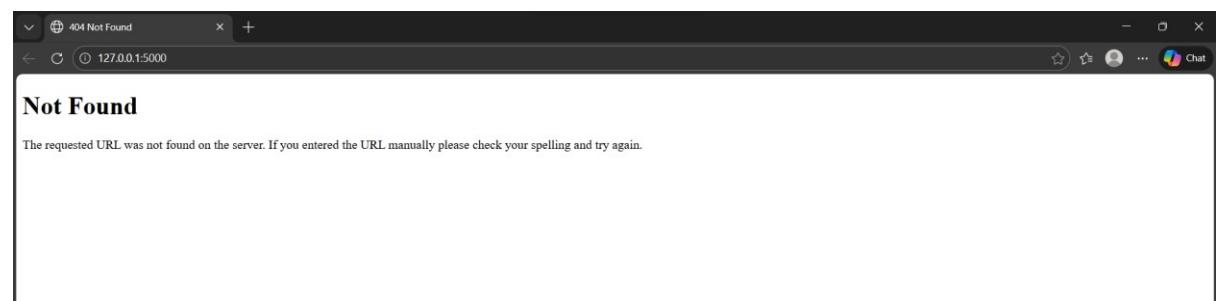
```
# Generate a simple REST API for user registration.
from flask import Flask, request
app = Flask(__name__)
@app.route('/register', methods=['POST'])
def register():
    username = request.json['username']
    password = request.json['password']
    api_key = "HARDCODED_API_KEY"
    return {"message": "User registered successfully"}
if __name__ == "__main__":
    app.run(debug=True)
```

OUTPUT:



The screenshot shows the same code editor interface after running the code. The terminal tab at the bottom displays the command to run the application and the resulting output. The output shows the application is running on port 5000 with a warning about using it in production. The code editor interface remains the same with its sidebar and bottom navigation.

```
PS C:\Users\sarik\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64\python
n.exe "C:/Users/sarik/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
```



```

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT

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"

•

* Running on http://127.0.0.1:5000
Press CTRL+C to quit

* Restarting with stat

* Debugger is active!

* Debugger PIN: 108-610-262

127.0.0.1 - - [20/Jan/2026 21:46:17] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [20/Jan/2026 21:46:17] "GET /favicon.ico HTTP/1.1" 404 -

Ln 14, Col 24 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 (i) Go Live BLACKBOXAI: Open Chat [?] Augment

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):

```

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
28
29
30
31 if __name__ == "__main__":
32     app.run(debug=True, host="0.0.0.0", port=5000)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT

Q AI ASSISTED CODING

EXPLORER secure_app.py > ...

AI ASSISTED CODING > ASSIGNMENTS > 5-2.html

ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py ASSIGN-5-2.py DAY-12.py insecure_app.py LAB-2.py LAB(2)(PRACTICE SESSION...) Prime.py secure_app.py

OUTLINE TIMELINE

Ln 33, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 (i) Go Live BLACKBOXAI: Open Chat [?] Augment

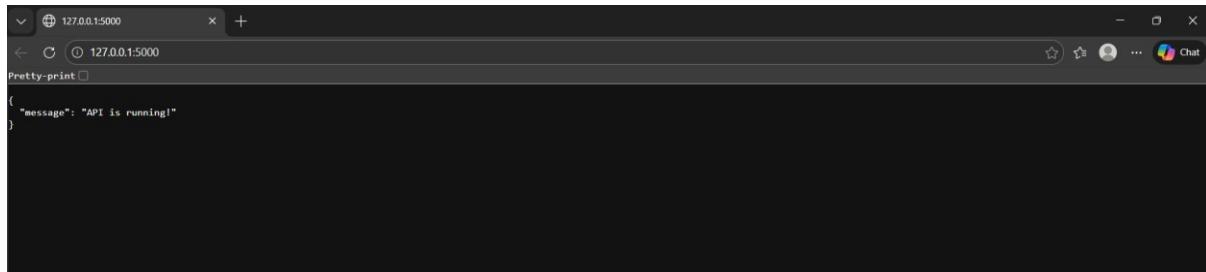
OUTPUT:

```

12     def index():
13         return jsonify({"message": "API is running!"})
14
15     @app.route('/register', methods=['POST'])
16     def register():
17
18     if __name__ == '__main__':
19         app.run()

```

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/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



```

12     def index():
13         return jsonify({"message": "API is running!"})
14
15     @app.route('/register', methods=['POST'])
16     def register():
17
18     if __name__ == '__main__':
19         app.run()

```

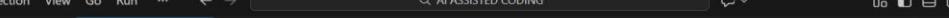
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64\python
n.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/secure_app.py"
* Serving Flask app 'secure_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production 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 - - [28/Jan/2026 21:41:10] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/Jan/2026 21:41:10] "GET /favicon.ico HTTP/1.1" 404 -
10.3.48.143 - - [28/Jan/2026 21:41:46] "GET / HTTP/1.1" 200 -
10.3.48.143 - - [28/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:



The screenshot shows the Visual Studio Code interface with the 'AI ASSISTED CODING' extension active. The left sidebar has an 'EXPLORER' tab selected, showing a tree view with 'ASSIGN-5-2.py' expanded. The main editor area displays Python code for a scholarship checker. The status bar at the bottom right indicates the file is 100% complete.

```
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
```

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 Visual Studio Code interface with the following details:

- File Explorer:** On the left, under the "AI ASSISTED CODING" category, the file "ASSIGN-5-2.py" is selected. Other files listed include "ASSIGNMENTS", "ASSIGN-1-3.py", "ASSIGN-2-2.py", "ASSIGN-4-2.py", "ASSIGN-5-2.py", "DAY--1-2.py", "LAB-2.py", "LAB(2)(PRACTICE SESSION)", and "Prime.py".
- Code Editor:** The main area displays the following Python code:

```
def scholarship_eligibility_fair(score, income):
    if score >= 80 and income <= 300000:
        return True
    return False

print(scholarship_eligibility_biased(90, 150000, "urban"))
print(scholarship_eligibility_fair(82, 250000))
```
- Right Panel:** A "Build with Agent" panel is open, featuring a speech bubble icon and the text "Build with Agent". Below it, a note says "All responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."

OUTPUT:

The screenshot shows a Visual Studio Code interface with the following details:

- Code Editor:** The code being run is:

```
13 print(scholarship_eligibility.fair(82, 250000))
14
```
- Terminal:** The terminal output shows:

```
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/ASSIGN-5-2.py"
True
True
```
- Status Bar:** The status bar at the bottom indicates: Line 13, Col 10, spaces: 4, 011-0, CRLF, Python 3.14.2, Go Live, BLACKBOXAI: Open Chat, and Augment.

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.

The screenshot shows a code editor interface with a dark theme. In the Explorer sidebar, there are several files listed under 'AI ASSISTED CODING': ASSIGN-1-3.py, ASSIGN-2-2.py, ASSIGN-4-2.py, DAY--1.2.py, LAB-2.py, LAB(2)(PRACTICE SESSION).py, and Prime.py. The main editor area displays the content of 'ASSIGN-5-2.py'. The code defines a function 'is_prime(n)' that checks if a number is prime. It includes comments explaining the logic, such as generating a function to check primality with comments and explanation, and testing divisibility from 2 up to the square root of the number. The right-hand sidebar has a 'Build with Agent' section with a note that AI responses may be inaccurate and a button to 'Generate Agent Instructions'.

```
15
16     ## Generate a function to check whether a number is prime with comments and explanation.
17     def is_prime(n):
18
19         if n <= 1:
20             return False
21
22         for i in range(2, int(n ** 0.5) + 1):
23             if n % i == 0:
24                 return False
25         return True
26
27     print(is_prime(11))
28     print(is_prime(15))
```

OUTPUT:

The screenshot shows a terminal window within a code editor. The terminal output shows the execution of 'ASSIGN-5-2.py'. The code defines a function 'is_prime(n)' that returns True if n is prime and False otherwise. It prints the results for 11 and 15. The terminal also shows the path 'C:\Users\srak\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/srak/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/srak/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-5-2.py"' and the results 'True' and 'False'. The right-hand sidebar has a 'Build with Agent' section with a note that AI responses may be inaccurate and a button to 'Generate Agent Instructions'.

```
23         if n % 1 == 0:
24             return False
25         return True
26     print(is_prime(11))
27     print(is_prime(15))
```

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

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 a code editor interface with the title bar "AI ASSISTED CODING". The left sidebar is titled "EXPLORER" and lists several Python files: ASSIGN-1-3.py, ASSIGN-2-2.py, ASSIGN-4-2.py, ASSIGN-5-2.py, DAY-12.py, LAB-2.py, LAB(2)(PRACTICE SESS..., and Prime.py. The main editor area displays the following Python code:

```
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
34     print(employee_score(90, 80, 95))
```

A sidebar on the right is titled "Build with Agent" with the sub-instruction "Generate Agent Instructions to onboard AI onto your codebase".

OUTPUT:

The screenshot shows a terminal window with the title bar "AI ASSISTED CODING". The left sidebar is identical to the one in the code editor. The terminal tab is active, showing the command "python ASSIGN-5-2.py" 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>
```

A sidebar on the right is titled "Build with Agent" with the sub-instruction "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 title "5-2.html". The address bar indicates the file is located at "C:/Users/sarik/OneDrive/Desktop/API%20ASSISTED%20CODING/5-2.html". The page displays a user feedback form. It contains three input fields: "Name (Optional)" with the value "Sarikasuresh Goud", "Your Feedback" with the value "Good", and "Experience Rating" with the value "Good". A "Submit Feedback" button is also present.

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.