

Harika Jupaka

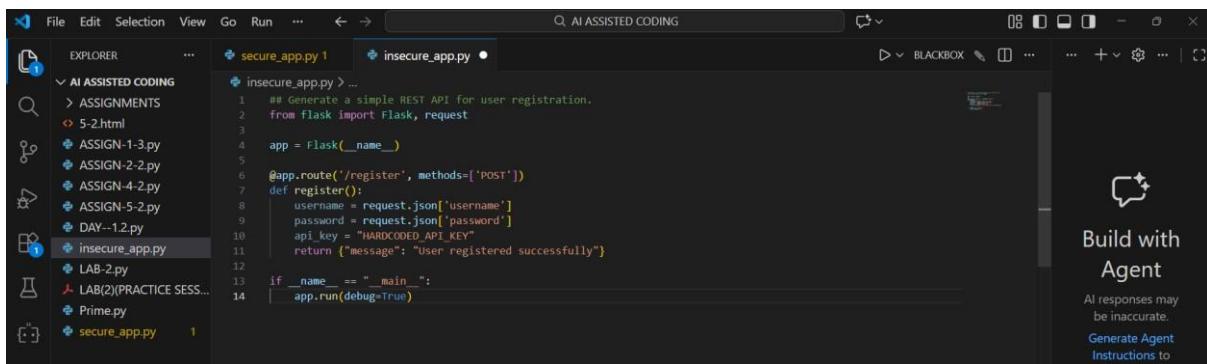
2403a51l31

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.

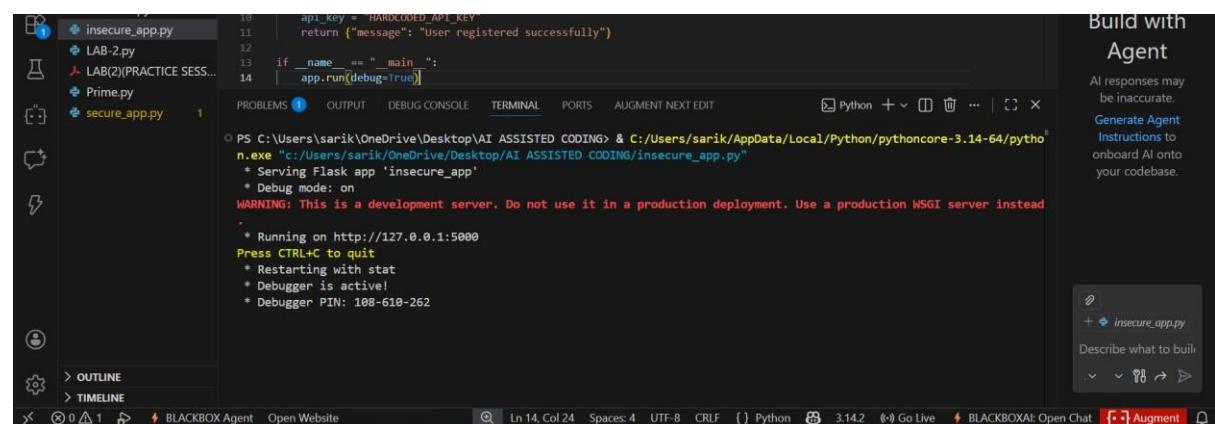


The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files like `insecure_app.py`, `secure_app.py`, and several assignment files (e.g., `ASSIGN-1-3.py`, `ASSIGN-2-2.py`, etc.).
- Code Editor:** Displays the `insecure_app.py` file with the following code:

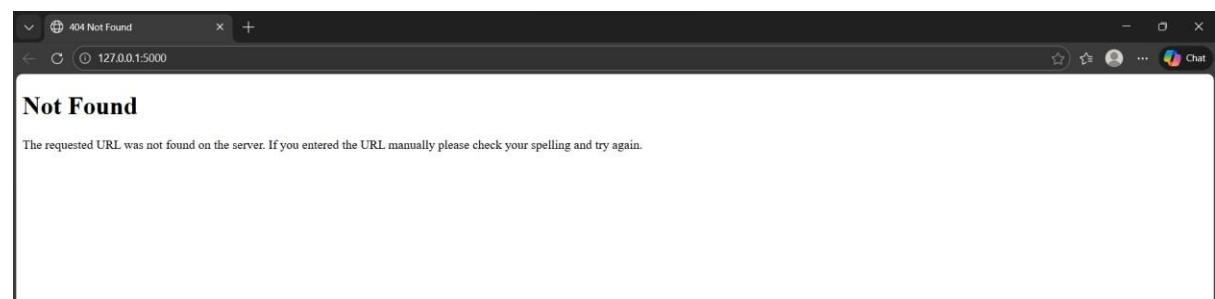
```
# Generate a simple REST API for user registration.
from flask import Flask, request
app = Flask(__name__)
@app.route('/register', methods=['POST'])
def register():
    username = request.json['username']
    password = request.json['password']
    api_key = "HARDCODED_API_KEY"
    return {"message": "User registered successfully"}
if __name__ == "__main__":
    app.run(debug=True)
```
- AI Assistant Panel:** On the right, there's a panel titled "Build with Agent" with the message "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."

OUTPUT:



The screenshot shows the output of running the `insecure_app.py` script in a terminal window:

```
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
```



```

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

if __name__ == "__main__":
    app.run(debug=True)

```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

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 -

< OUTLINE > TIMELINE BLACKBOX Agent Open Website

Ln 14, Col 24 Spaces: 4 UTF-8 CRLF Python 3.14.2 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):

```

    ## 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)

```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

EXPLORER AI ASSISTED CODING secure_app.py > ... secure_app.py > ...

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 SESS... Prime.py secure_app.py

< OUTLINE > TIMELINE BLACKBOX Agent Open Website

Ln 33, Col 1 Spaces: 4 UTF-8 CRLF Python 3.14.2 Go Live BLACKBOXAI: Open Chat Augment

OUTPUT:

```

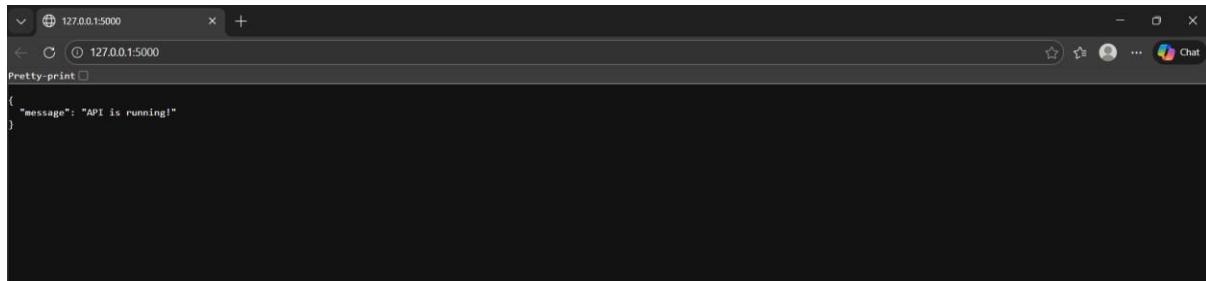
12 def index():
13     return jsonify({"message": "API is running!"})
14
15 @app.route('/register', methods=['POST'])
16 def register():

```

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.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

OUTLINE



```

12 app.route('/register', methods=['POST'])
13
14 def register():

```

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.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 -

OUTLINE TIMELINE

BLACKBOX Agent Open Website

Ln 33, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.2 ⚡ Go Live BLACKBOXAI: Open Chat ⚡ Augment

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:

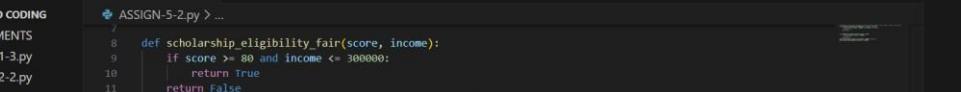


The screenshot shows the Visual Studio Code interface with the 'AI ASSISTED CODING' extension active. The left sidebar has a tree view under 'EXPLORER' with items like 'ASSIGNMENTS', 'ASSIGN-1-3.py', 'ASSIGN-2-2.py', 'ASSIGN-4-2.py', and 'ASSIGN-5-2.py'. The main editor area displays Python code for a scholarship checker:

```
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 "AI ASSISTED CODING" extension active. The "EXPLORER" sidebar on the left lists various Python files. The "ASSIGN-5-2.py" file is currently selected and open in the main editor area. The code defines a function to check if a student is eligible for a scholarship based on their score and income. The "AI ASSISTED CODING" status bar at the bottom right indicates that AI responses may be inaccurate and provides options to generate agent instructions or onboard AI onto the codebase.

```
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))
```

OUTPUT:

The screenshot shows a Visual Studio Code interface. The top bar has tabs for 'Home' and 'Agent'. The code editor on the left has icons for file operations like New, Open, Save, and Delete. The main area shows a Python script with two lines of code: `print(scholarship_eligibility_fair(82, 250000))` and a closing brace. Below the editor is a toolbar with 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and 'AUGMENT NEXT EDIT' buttons. A dropdown menu for 'Python' is open. The terminal below shows command-line output in a Windows environment (PS C:\...). The command run was `python.exe "c:/Users/sarik/Desktop/AI ASSISTED CODING/ASSIGN-5-2.py"`. The output shows two 'True' responses. At the bottom, there's an 'OUTLINE' view, a timeline view, and a status bar with file paths, line numbers (Ln 13, Col 10), and other settings.

```
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/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.

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' that checks if a number is prime by testing divisibility from 2 up to the square root of the number. It includes inline comments explaining the logic. A sidebar on the right is titled 'Build with Agent' and contains a note that AI responses may be inaccurate, along with 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         if n <= 1:
19             return False
20
21         for i in range(2, int(n ** 0.5) + 1):
22             if n % i == 0:
23                 return False
24
25         return True
26
27 print(is_prime(11))
28 print(is_prime(15))
```

OUTPUT:

This screenshot shows the same code editor environment. The terminal tab is active, displaying the command 'python ASSIGN-5-2.py' being run. The output shows the program printing 'True' for 11 and 'False' for 15. The terminal window also shows the current working directory as 'C:\Users\...'. The sidebar on the right remains the same, with the 'Build with Agent' section visible.

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

```
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
○ False
○ PS C:\Users\sarik\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--1.2.py, LAB-2.py, LAB(2)(PRACTICE SESS..., and Prime.py. The main editor area displays the following Python code:

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

A right-hand sidebar titled "Build with Agent" contains the message: "AI responses may be inaccurate. 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: "87.5". The right-hand sidebar "Build with Agent" is also present.

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 consists of 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.