

A.NagaKoushik

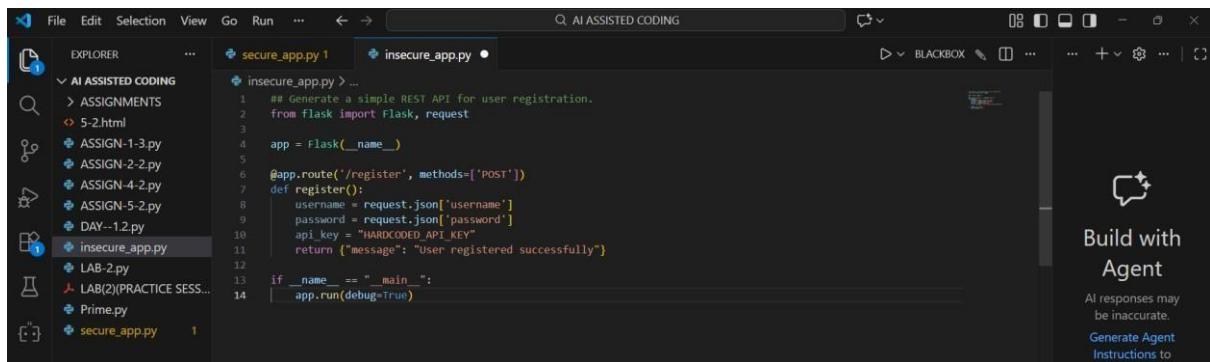
2403a51l22

Batch-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 the following code in `insecure_app.py`:

```
## Generate a simple REST API for user registration.
from Flask import Flask, request

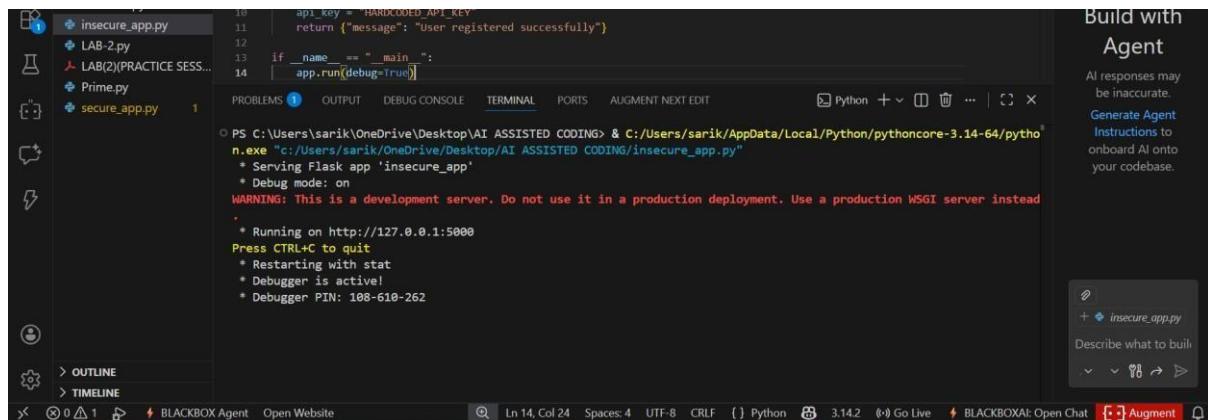
app = Flask(__name__)

@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)
```

The code defines a `register` endpoint that expects JSON input for `username` and `password`, and returns a success message. It also runs the Flask application with `debug=True`.

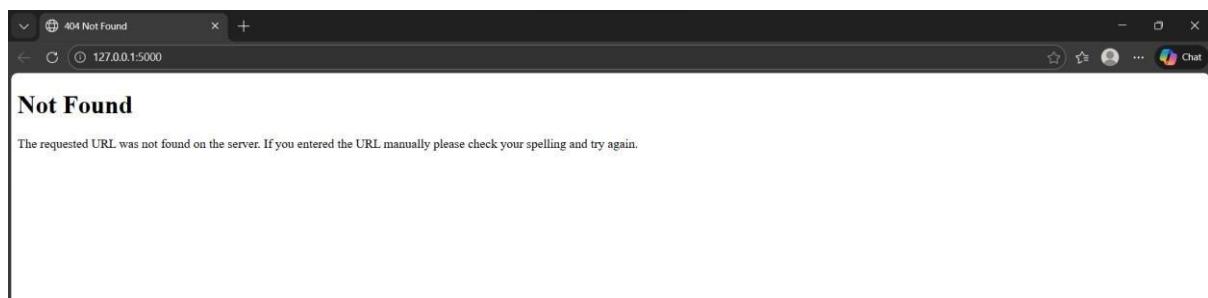
OUTPUT:



The screenshot shows the terminal output of running the script:

```
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 application is running on port 5000.



The screenshot shows a Python script named `insecure_app.py` in a code editor. The script contains code to handle a POST request for password registration. A terminal window below shows the script running and listening on port 5000. The terminal output includes a debugger PIN and two log entries indicating successful HTTP requests.

```
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.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: 108-618-262

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 -

> 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):

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

- File Explorer:** Shows a tree view with various files and folders, including 'secure_app.py' which is currently selected.
- Code Editor:** Displays the content of 'secure_app.py'. The code implements a secure API using Flask, token-based authentication, and JSON responses. It includes routes for index, register, and running status checks.
- AI ASSISTED CODING:** A sidebar on the right provides AI assistance, including a 'Build with Agent' section with a note about potential inaccuracy and instructions for onboard AI.

```
secure_app.py

# Secure API (Corrected - Token-Based Authentication)
from flask import Flask, request, jsonify
import os
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 Python development environment with the following components:

- File Explorer:** Shows files like `ASSIGN-5-2.py`, `DAY-12.py`, `insecure_app.py`, `LAB-2.py`, `LAB(2)(PRACTICE SESSION).py`, `Prime.py`, and `secure_app.py`.
- Terminal:** Displays command-line output for running `secure_app.py`. It shows the app is running on multiple ports (0.0.0.0 and 127.0.0.1) and provides a debugger PIN.
- Output:** Shows the JSON response from the application: `{"message": "API is running!"}`.
- Browser Preview:** A screenshot of a browser window showing the same JSON response.
- AI Integration:** A sidebar titled "Build with Agent" includes a message about AI responses being inaccurate, a button to "Generate Agent Instructions", and a panel for describing what to build, currently set to `secure_app.py`.
- Bottom Bar:** Includes icons for BLACKBOX Agent, Open Website, and Augment, along with status information like Ln 33, Col 1, and Python 3.14.2.

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 'ASSIGNMENTS' expanded, containing files like 'ASSIGN-1-3.py', 'ASSIGN-2-2.py', 'ASSIGN-4-2.py', and 'ASSIGN-5-2.py'. The main editor area displays a Python script named 'ASSIGN-5-2.py' with the following code:

```
## Generate a scholarship eligibility checker based on academic score, family income, and location.
def scholarship_eligibility_biased(score, income, location):
    if score > 85 and income < 200000 and location == "urban":
        return True
    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 a code editor interface with the following details:

- File Explorer:** Shows a tree view of files under "AI ASSISTED CODING". The file "ASSIGN-5-2.py" is selected.
- Code Editor:** 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))
```
- Toolbar:** Includes standard icons for File, Edit, Selection, View, Go, Run, and various document-related functions.
- Header:** Shows "AI ASSISTED CODING" and other UI elements like a search bar and a refresh icon.
- Bottom Right:** A callout bubble says "Build with Agent" and "AI responses may be inaccurate." It also includes a "Generate Agent Instructions" button.

OUTPUT:

The screenshot shows a Python script named `scholarship_eligibility_fair.py` being run in a terminal window. The terminal output indicates that the function `scholarship_eligibility_fair` returns `True` for two different input pairs: `(82, 250000)` and `(82, 2500000)`.

```
print(scholarship_eligibility_fair(82, 250000))
print(scholarship_eligibility_fair(82, 2500000))

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.

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

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

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/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 following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Toolbar:** Back, Forward, Home, AI ASSISTED CODING button, Refresh, Save, Copy, Paste, Find, Replace, Select All, Undo, Redo, New, Open, Save, Save As, Save All, Close, Exit.
- SIDE BAR:** Explorer, AI ASSISTED CODING, ASSIGNMENTS, ASSIGN-1-3.py, ASSIGN-2-2.py, ASSIGN-4-2.py, ASSIGN-5-2.py (highlighted), DAY-1-2.py, LAB-2.py, LAB(2)(PRACTICE SESS...), Prime.py.
- CODE EDITOR:** The file `ASSIGN-5-2.py` is open. The code defines a function `employee_score` that calculates a score based on project rate, teamwork, and attendance. The code is as follows:

```
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, 88, 95))
```

RIGHT SIDE: A sidebar titled "Build with Agent" contains a message: "AI responses may be inaccurate." Below it are buttons for "Generate Agent Instructions" and "onboard AI onto your codebase." There is also a "Feedback" icon.

OUTPUT:



The screenshot shows a Windows taskbar at the bottom with icons for File Explorer, Task View, Start, and a search bar. The main area is a code editor in VS Code. On the left sidebar, there are icons for file navigation and a list of files: ASSIGN-4-2.py, ASSIGN-5-2.py, DAY-1.2.py, LAB-2.py, LAB(2) (PRACTICE SESSION), and Prime.py. The file 'ASSIGN-5-2.py' is currently selected. The code in the editor is:

```
33 |     return score
34 | print(employee_score(90, 80, 95))
```

The terminal tab is active, showing the command line 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 floating message bubble in the top right corner says 'Build with Agent'. A status bar at the bottom right indicates 'All responses may be inaccurate. Generate Agent Instructions to onboard AI onto'.

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 content is a user feedback form. It contains the following elements:

- A text input field labeled "Name (Optional)" with the value "Sarikasuresh Goud".
- A text area labeled "Your Feedback" containing the text "GOOD".
- A select menu labeled "Experience Rating" with the value "Good".
- A submit button labeled "Submit Feedback".

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.