

Varshini Girugula

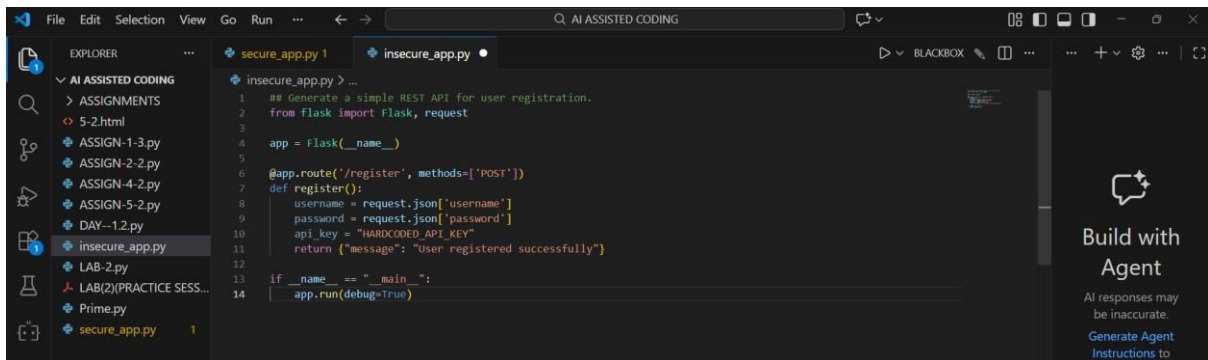
2403a51114

Batch-51

## Lab 5: Ethical Foundations – Responsible AI Coding Practices

### Task Description – 1: Secure API Usage

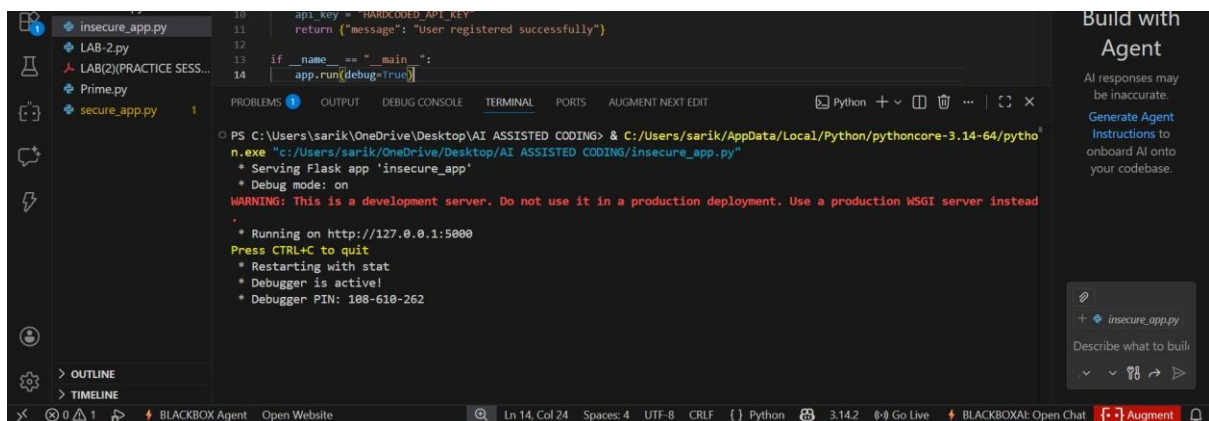
**Prompt:** Generate a simple REST API for user registration.



The screenshot shows the Visual Studio Code editor with the file `insecure_app.py` open. The code is a simple Flask application for user registration. The left sidebar shows the Explorer view with a list of files including `insecure_app.py`. The right sidebar shows the 'Build with Agent' panel.

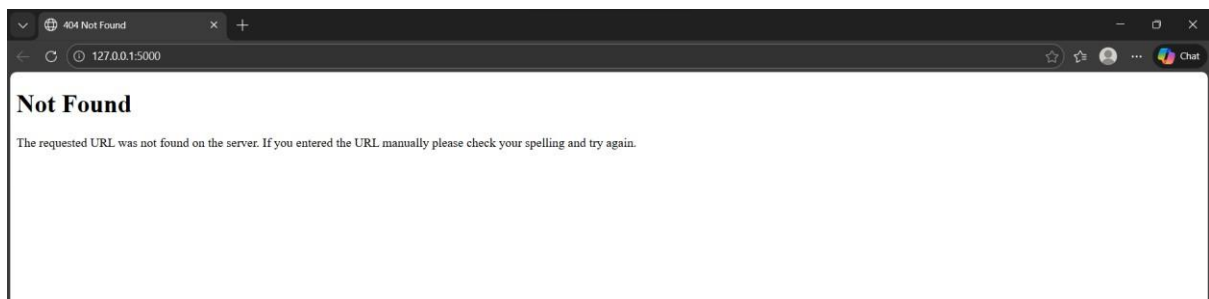
```
1  ## Generate a simple REST API for user registration.
2  from flask import Flask, request
3
4  app = Flask(__name__)
5
6  @app.route('/register', methods=['POST'])
7  def register():
8      username = request.json["username"]
9      password = request.json["password"]
10     api_key = "HARDCODED_API_KEY"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

### OUTPUT:



The screenshot shows the Visual Studio Code editor with the file `insecure_app.py` open. The terminal window at the bottom shows the output of the application. The left sidebar shows the Explorer view with a list of files including `insecure_app.py`. The right sidebar shows the 'Build with Agent' panel.

```
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
```



The screenshot shows a VS Code editor with a Python file named `insecure_app.py`. The code is a simple Flask application with a single route `/` that returns a success message. The terminal output shows the application running on `http://127.0.0.1:5000`. A 404 error is displayed in the terminal, indicating that the browser cannot find the requested resource. The error message is: `127.0.0.1 - - [28/Jan/2026 21:46:17] "GET / HTTP/1.1" 404 -`. The status bar at the bottom indicates the application is running on Python 3.14.2.

**Explanation:** You got 404 error because your Flask app does not have a home (`/`) route, so the browser cannot find that page.

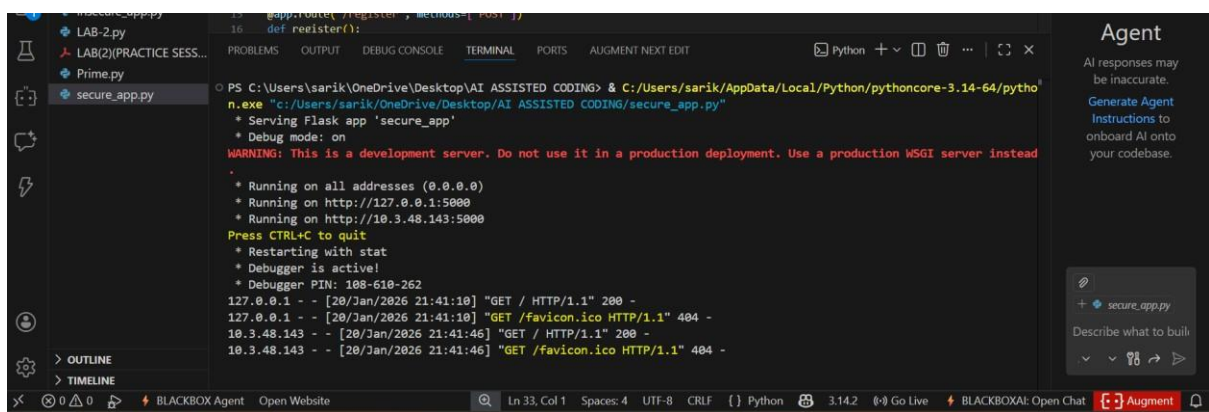
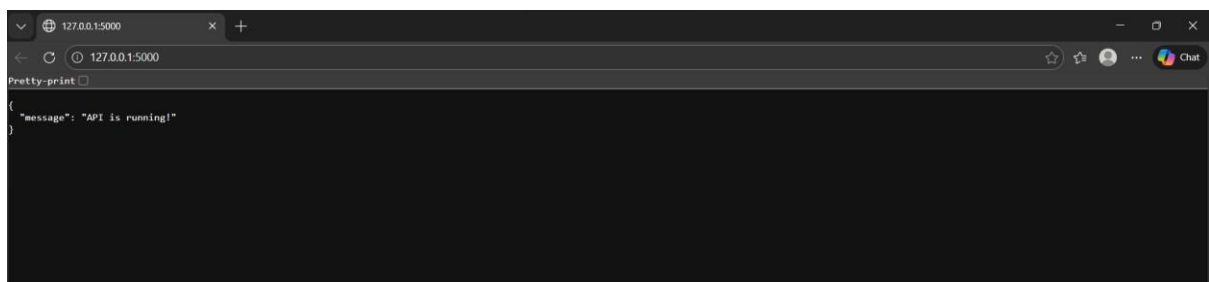
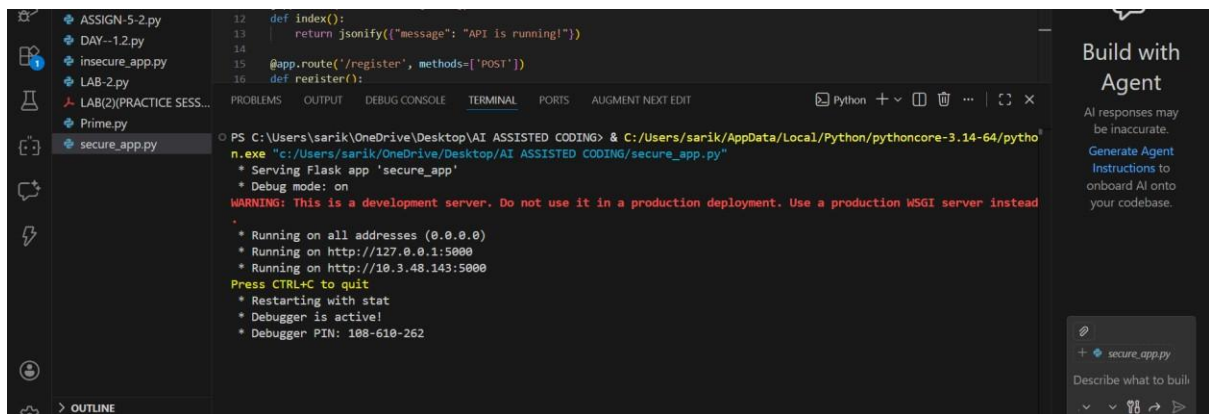
## Identified Security Flaws:

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 VS Code editor with a Python file named `secure_app.py`. The code is a Flask application with two routes: `/` (GET) and `/register` (POST). The `/register` route implements token-based authentication using JWT. The terminal output shows the application running on `http://0.0.0.0:5000`. The status bar at the bottom indicates the application is running on Python 3.14.2.

**OUTPUT:**



**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:

```

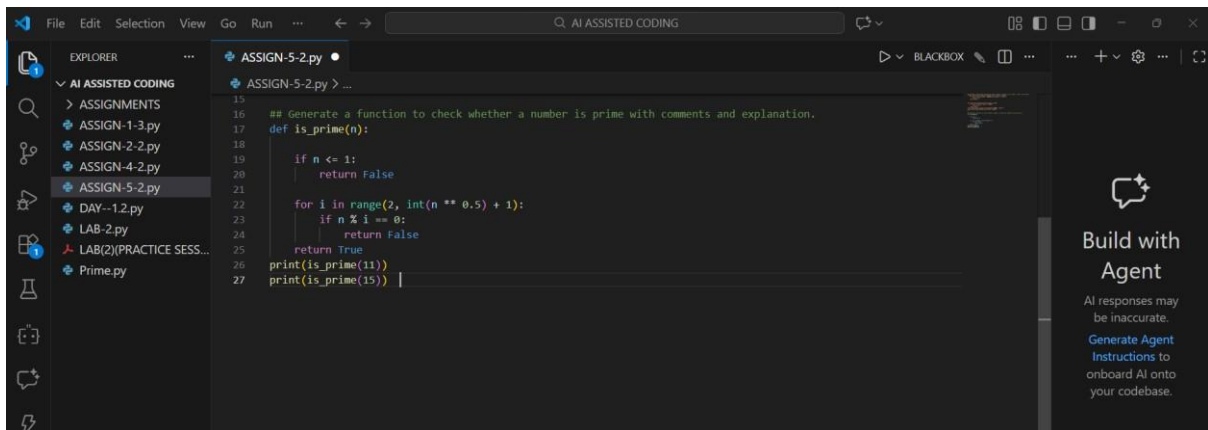
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-5-2.py"
True
True

```

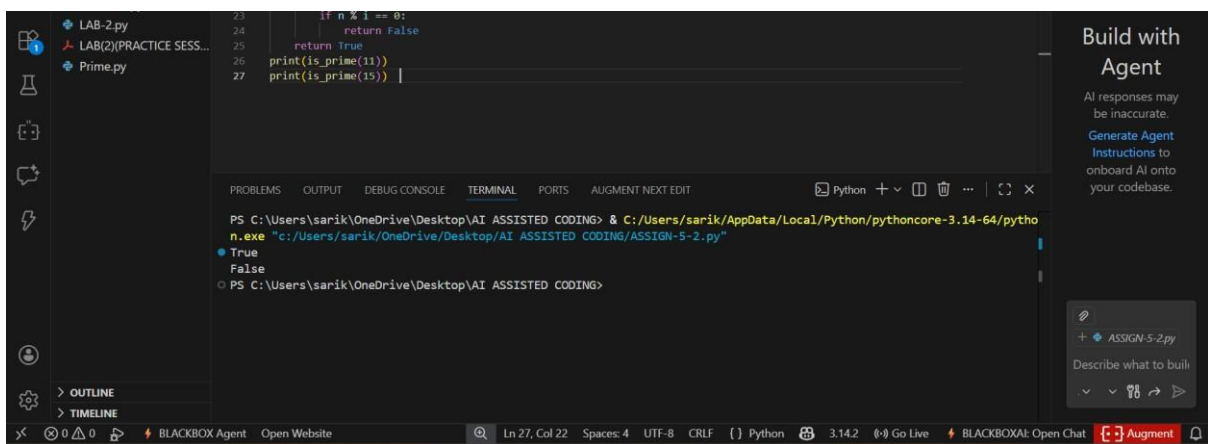
**Explanation:** The original logic introduces geographic bias by favoring urban students. Location should not be a deciding factor unless justified by policy. A fair system focuses on merit and economic need. Weighted or threshold-based criteria help ensure equitable access.

## Task Description – 3: Explainability

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



## OUTPUT:

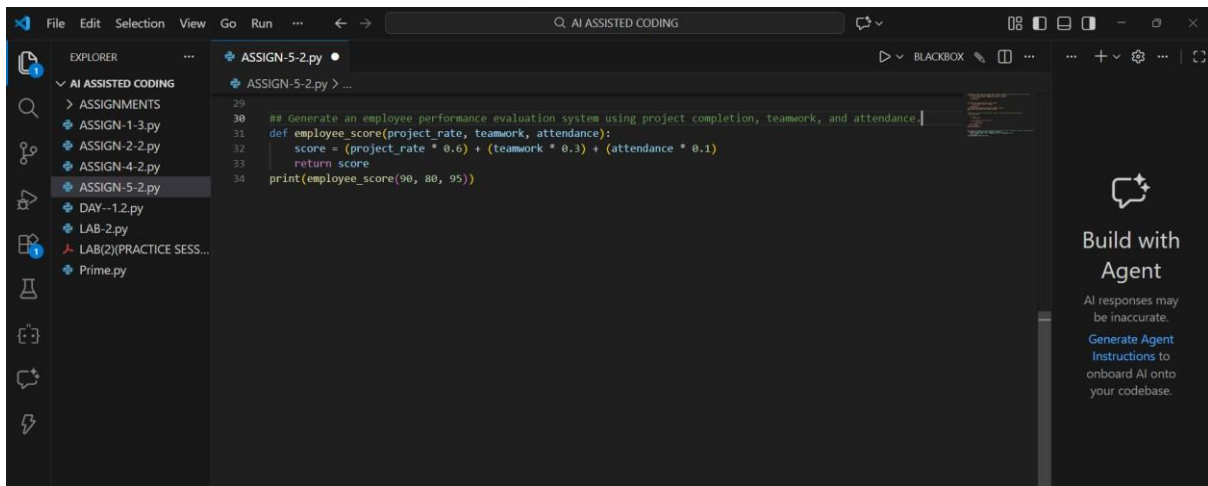


**Explanation:** The function first checks if the number is greater than 1. It then tests divisibility from 2 up to the square root of the number to reduce computation. If any divisor is found, the number is not prime; otherwise, it is prime.

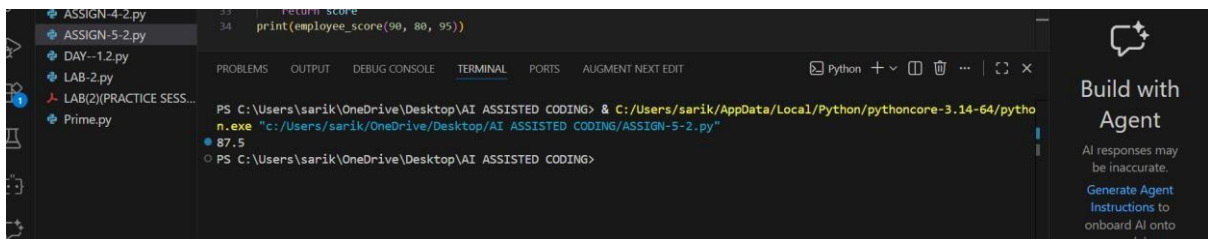
The explanation is clear, correct, and efficient. Inline comments improve readability and help beginners understand the logic easily.

## Task Description – 4: Ethical Scoring System

**Prompt:** Generate an employee performance evaluation system using project completion, teamwork, and attendance.



## OUTPUT:



## 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 x +
File Edit View

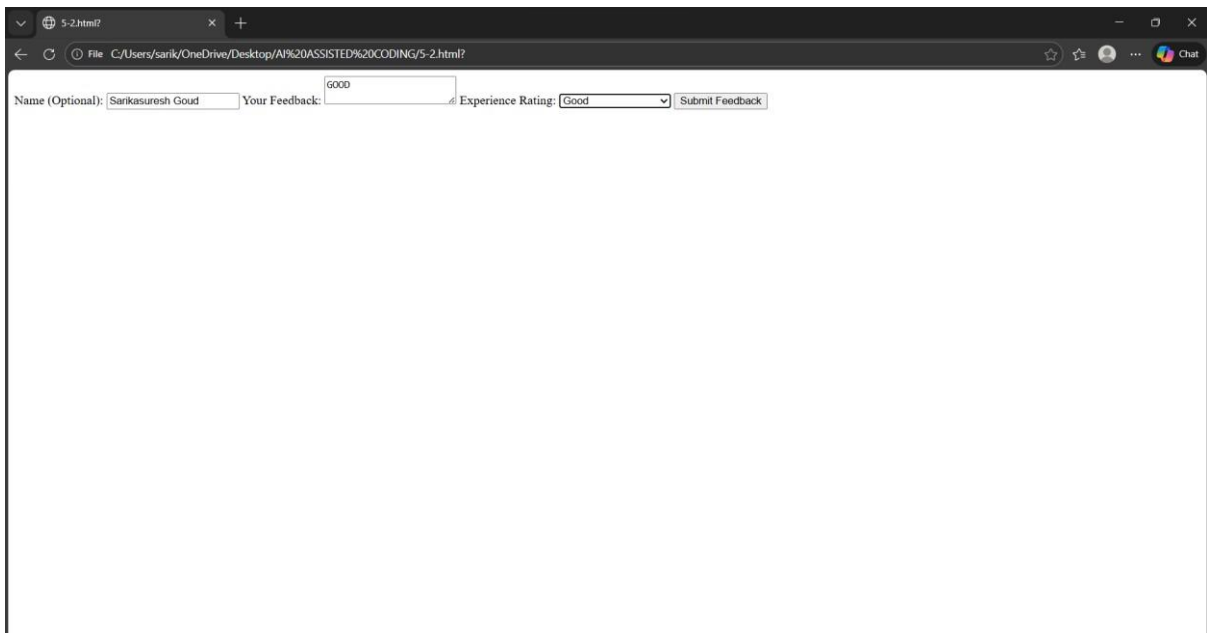
<form aria-label="User Feedback Form">
  <label for="name">Name (Optional):</label>
  <input type="text" id="name" aria-required="false">

  <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 titled 'User Feedback Form' and contains the following elements:

- Name (Optional):** A text input field containing the value 'Sarikasuresh Goud'.
- Your Feedback:** A text area containing the value 'Good'.
- Experience Rating:** A dropdown menu with the following options: 'Very Good', 'Good', 'Neutral', and 'Needs Improvement'. The current selection is 'Good'.
- Submit Feedback:** A button to submit the form.

**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.