

## School of Computer Science and Artificial Intelligence

### Lab Assignment 5.2

<b>Program</b>	: B. Tech (CSE)
<b>Course Title</b>	: AI Assisted Coding
<b>Course Code</b>	: 23CS002PC304
<b>Semester</b>	: III
<b>Academic Session</b>	: 2025-2026
<b>Name of Student</b>	: Sruthi
<b>Enrollment No.</b>	: 2403A51L10
<b>Batch No.</b>	: 51
<b>Date</b>	: 20-01-2026

### Task 1 : Secure API Usage

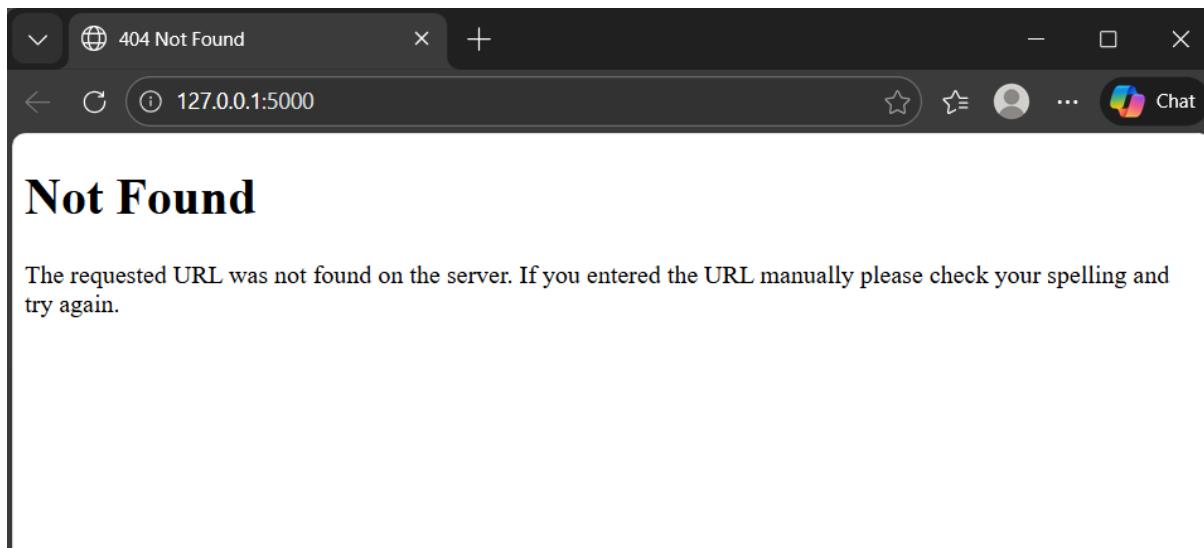
**Prompt:** Generate a simple REST API for user registration using Flask.

**Code:**

```
from flask import Flask, request, jsonify
app = Flask(__name__)
users = []
@app.route('/register', methods=['POST'])
def register_user():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')
    if not username or not password:
        return jsonify({'message': 'Username and password are required!'}), 400
    for user in users:
        if user['username'] == username:
            return jsonify({'message': 'Username already exists!'}), 400
    users.append({'username': username, 'password': password})
    return jsonify({'message': 'User registered successfully!'}), 201
if __name__ == '__main__':
    app.run(debug=True)
```

**Output:**

```
PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code>
 & "C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\venv\Scripts\python.exe" "c:/Users/BURRA SRUTHI/OneDrive/Desktop/3_2/AI/AI_code/Assign_5-2.py"
 * Serving Flask app 'Assign_5-2'
 * 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: 432-133-412
```



### **Explanation:**

The code does not show output because it is a Flask API that returns responses only when a POST request is sent to the /register endpoint, not through normal program execution or print statements.

### **Issues:**

1. Improper Handling of API Keys
  - The application does not require or use any API keys
  - There is no exposure or hardcoding of API keys
  - This is acceptable for a simple registration API, but real systems often require API keys or tokens
2. Missing Authentication
  - Any user can access the /register endpoint
  - There is no login system, token, or session handling
  - No authorization checks exist
3. Lack of Input Validation

## Corrected Code:

```

from flask import Flask, request, jsonify
app = Flask(__name__)
users = []
@app.route('/')
def home():
    return "Flask server is running"
@app.route('/register', methods=['GET', 'POST'])
def register_user():
    if request.method == 'GET':
        return jsonify({
            "message": "Use POST method with JSON data (username, password)"
        })
    if not request.is_json:
        return jsonify({'message': 'Request must be JSON'}), 400
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')
    if not username or not password:
        return jsonify({'message': 'Username and password are required'}), 400
    for user in users:
        if user['username'] == username:
            return jsonify({'message': 'Username already exists'}), 400
    users.append({'username': username, 'password': password})
    return jsonify({'message': 'User registered successfully'}), 201
if __name__ == '__main__':
    app.run(debug=True)

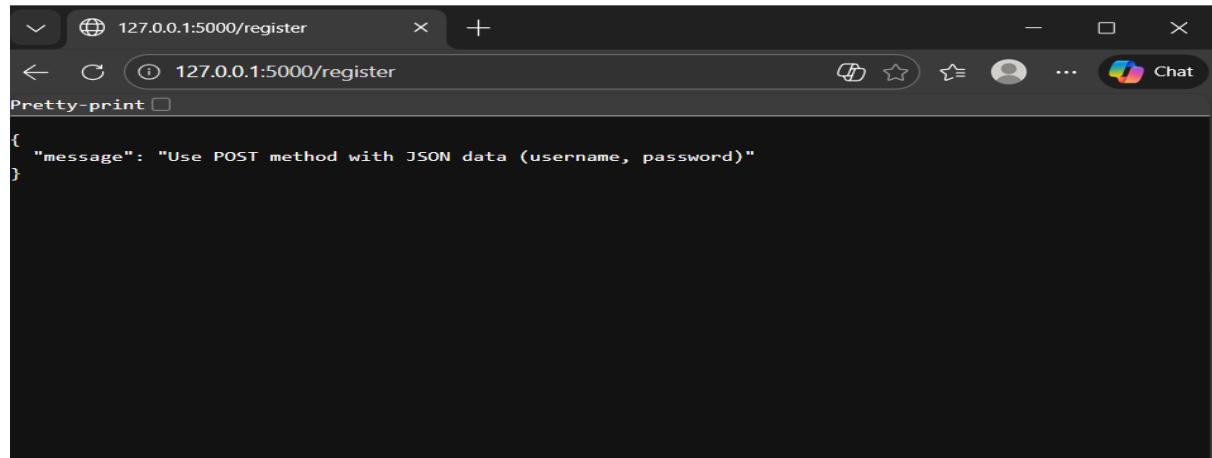
```

## Output:

```

PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code> & "C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\venv\Scripts\python.exe" "c:/Users/BURRA SRUTHI/OneDrive/Desktop/3_2/AI/AI_code/Assign_5-2.py"
* Serving Flask app 'Assign_5-2'
* 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: 432-133-412
127.0.0.1 - - [21/Jan/2026 13:52:29] "GET / HTTP/1.1" 200 -

```



## Task2: Fair Decision Logic

### Prompt:

Write a python program to generate a scholarship eligibility checker based on academic score, family income, and location.

### Code:

```
def check_scholarship_eligibility(academic_score, family_income, location):
    eligibility_criteria = {
        'academic_score': 85,
        'family_income': 50000,
        'location': ['rural', 'urban']
    }
    if (academic_score >= eligibility_criteria['academic_score'] and
        family_income <= eligibility_criteria['family_income'] and
        location in eligibility_criteria['location']):
        return "Eligible for scholarship"
    else:
        return "Not eligible for scholarship"
# Example usage
academic_score = 90
family_income = 40000
location = 'rural'
result = check_scholarship_eligibility(academic_score, family_income, location)
print(result) # Output: Eligible for scholarship
```

### Output:

```
PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code> & "C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\venv\Scripts\python.exe" "c:/Users/BURRA SRUTHI/OneDrive/Desktop/3_2/AI/AI_code/Assign_5-2.py"
Eligible for scholarship
```

### Observations on fairness and improvements:

The fixed academic score cutoff may disadvantage students from weaker educational backgrounds. The strict income limit excludes middle-income families who may also face financial hardship. Treating all locations the same ignores differences in access to resources between rural and urban areas. Fairness can be improved by using flexible score ranges, income brackets, and giving additional consideration to disadvantaged regions or backgrounds.

### Task3: Explainability

#### Prompt:

Write a python program to generate a function that checks whether a number is prime and include inline comments explaining each step.

#### Code:

```
def is_prime(number):
    # Check if the number is less than 2, which are not prime numbers
    if number < 2:
        return False
    # Loop through all numbers from 2 to the square root of the number
    for i in range(2, int(number**0.5) + 1):
        # If the number is divisible by any of these, it is not prime
        if number % i == 0:
            return False
    # If no divisors were found, the number is prime
    return True

# Example usage
num = 29
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.)
```

#### Output:

```
PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code> & "C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\.venv\Scripts\python.exe" "c:/Users/BURRA SRUTHI/OneDrive/Desktop/3_2/AI/AI_code/Assign_5-2.py"
29 is a prime number.
```

#### Explanation:

The program first checks if the number is less than 2, since numbers below 2 are not prime. Then, it tests divisibility only up to the square root of the number, which makes the algorithm efficient and logically correct. If any divisor is found, the function returns False; otherwise, it returns True, confirming the number is prime.

## Task4: Ethical Scoring System

### Prompt:

Write a python program to generate an employee performance evaluation system using inputs such as project completion rate, teamwork score, and attendance.

### Code:

```
def evaluate_employee_performance(project_completion_rate, teamwork_score, attendance):
    # Define thresholds for evaluation
    thresholds = {
        'project_completion_rate': 80,      # percentage
        'teamwork_score': 7,                # out of 10
        'attendance': 90                  # percentage
    }
    # Initialize performance score
    performance_score = 0
    # Check project completion rate
    if project_completion_rate >= thresholds['project_completion_rate']:
        performance_score += 1
    # Check teamwork score
    if teamwork_score >= thresholds['teamwork_score']:
        performance_score += 1
    # Check attendance
    if attendance >= thresholds['attendance']:
        performance_score += 1
    # Determine performance rating based on score
    if performance_score == 3:
        return "Excellent"
    elif performance_score == 2:
        return "Good"
    elif performance_score == 1:
        return "Satisfactory"
    else:
        return "Needs Improvement"
# Example usage
project_completion_rate = 85
teamwork_score = 8
attendance = 95
performance_rating = evaluate_employee_performance(project_completion_rate, teamwork_score, attendance)
print(f"Employee Performance Rating: {performance_rating}")
```

### Output:

```
PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code> & "C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\.venv\Scripts\python.exe" "c:/Users/BURRA SRUTHI/OneDrive/Desktop/3_2/AI/AI_code/Assign_5-2.py"
Employee Performance Rating: Excellent
```

### Analysis of Scoring Logic: Ethical Weighting and Bias

The scoring logic assigns equal weight to project completion rate, teamwork score, and attendance, which may introduce unintended bias. Treating attendance the same as project delivery and teamwork can disadvantage employees with valid health issues or flexible work arrangements. The fixed thresholds do not account for role differences, such as positions where teamwork or attendance may be less critical than output quality. To ensure ethical evaluation, the model should use role-specific weights, flexible attendance considerations, and normalized scoring rather than simple pass/fail thresholds.

## Task 5: Accessibility and Inclusiveness

**Prompt:** Write a python program to generate a user feedback form application.

### Code:

```

from flask import Flask, request, jsonify
app = Flask(__name__)
@app.route('/')
def home():
    return "Flask Feedback Form Server is running. Use /feedback to submit feedback."
@app.route('/feedback', methods=['GET', 'POST'])
def submit_feedback():
    # Handle GET request (for browser testing)
    if request.method == 'GET':
        return """
            <h2>User Feedback Form</h2>
            <p>Send a POST request with JSON data to submit feedback.</p>
            <p>Fields: name (optional), feedback (required), rating (required), accessibility_needs (optional)</p>
        """

    # Handle POST request (actual submission)
    if not request.is_json:
        return jsonify({'message': 'Request must be JSON format'}), 400
    data = request.get_json()
    # Extract inputs safely
    name = data.get('name', 'Anonymous')
    feedback = data.get('feedback')
    rating = data.get('rating')
    accessibility_needs = data.get('accessibility_needs', 'None')
    # Validate required fields
    if not feedback or rating is None:
        return jsonify({"message": "Feedback and rating are required"}), 400
    return jsonify([
        "message": "Thank you for your feedback!",
        "name": name,
        "rating": rating,
        "accessibility_needs": accessibility_needs
    ]), 200
if __name__ == '__main__':
    app.run(debug=True)

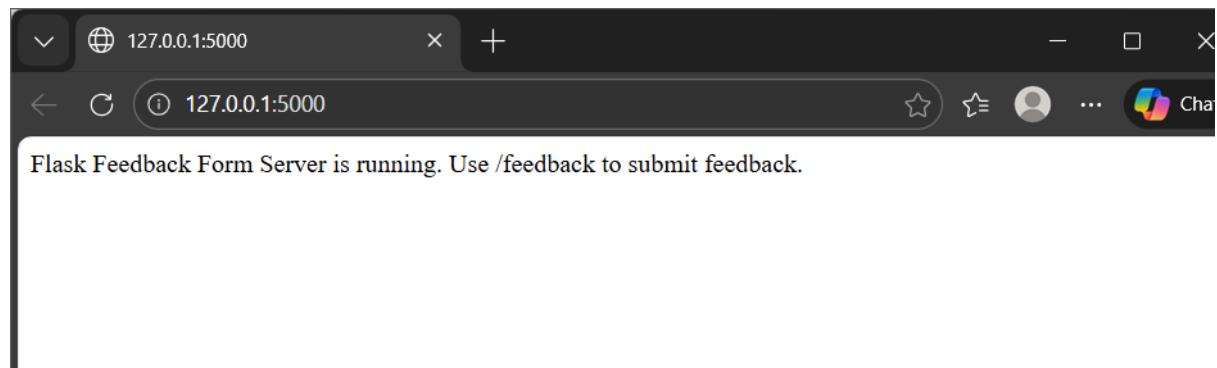
```

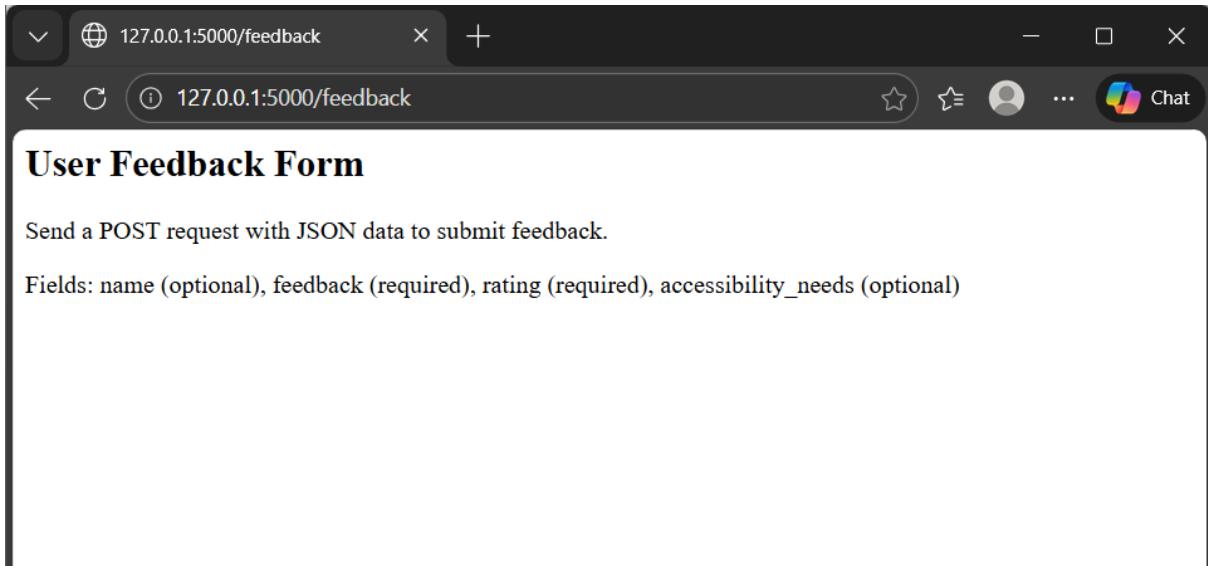
### Output:

```

PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code> & "C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\.venv\Scripts\python.exe" "c:/Users/BURRA SRUTHI/OneDrive/Desktop/3_2/AI/AI_code/Assign_5-2.py"
* Serving Flask app 'Assign_5-2'
* 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: 432-133-412

```





## Corrected code:

```

from flask import Flask, request, jsonify
app = Flask(__name__)
# Home route for server check
@app.route('/')
def home():
    return """
<h2>User Feedback Form Server is running</h2>
<p>To submit feedback, send a POST request to /feedback with JSON data.</p>
<p>Fields: name (optional), feedback (required), rating (1-5, required), accessibility_needs (optional).</p>
"""

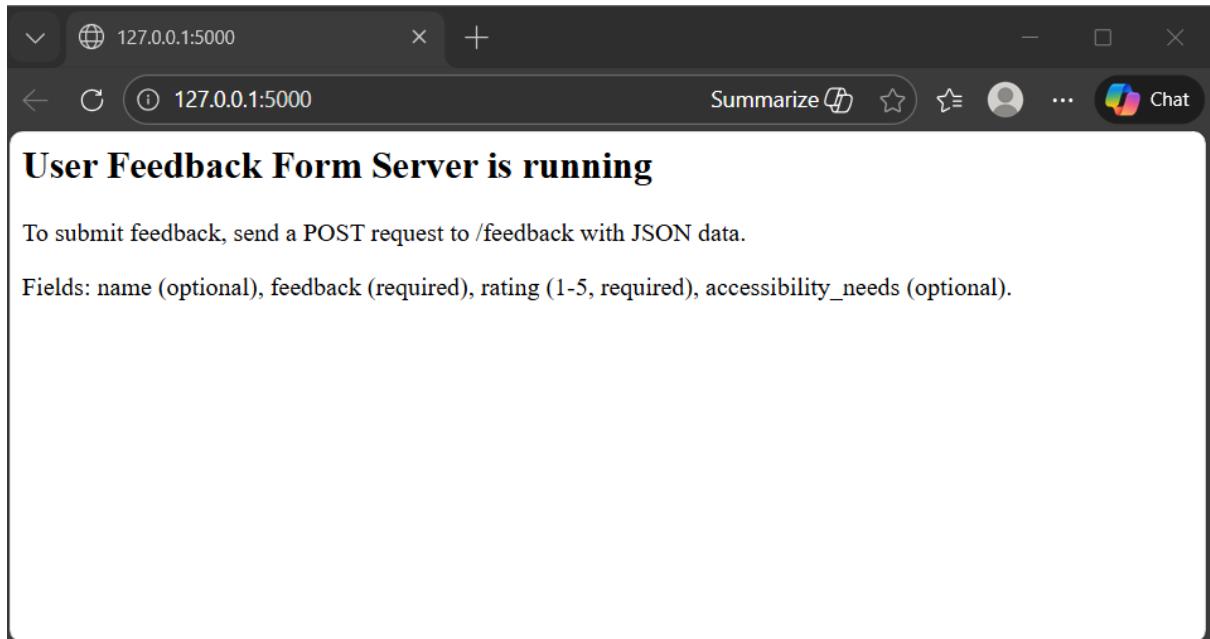
# Feedback route
@app.route('/feedback', methods=['GET', 'POST'])
def submit_feedback():
    # Handle GET request for browser instructions
    if request.method == 'GET':
        return """
<h3>Welcome to the User Feedback API</h3>
<p>Please submit feedback using a POST request with JSON data.</p>
<ul>
    <li><strong>name</strong> (optional): Your preferred name or 'Anonymous'.</li>
    <li><strong>feedback</strong> (required): Your feedback in your own words.</li>
    <li><strong>rating</strong> (required): Rate your experience from 1 (lowest) to 5 (highest).</li>
    <li><strong>accessibility_needs</strong> (optional): Any accessibility accommodations you may require.</li>
</ul>
<p>All users are welcome, and responses are treated respectfully and inclusively.</p>
"""

    # Handle POST request (actual feedback submission)
    if not request.is_json:
        return jsonify({'message': 'Request must be in JSON format'}), 400
    data = request.get_json()
    # Safely extract inputs
    name = data.get('name', 'Anonymous').strip()
    feedback = data.get('feedback', '').strip()
    rating = data.get('rating')
    accessibility_needs = data.get('accessibility_needs', 'None').strip()
    # Validate required fields
    if not feedback:
        return jsonify({'message': "Feedback text is required"}), 400
    if rating is None or not (1 <= rating <= 5):
        return jsonify({'message': "Rating is required and must be between 1 and 5"}), 400
    # Return confirmation
    return jsonify({
        "message": f"Thank you {name} for your feedback!",
        "rating": rating,
        "accessibility_needs": accessibility_needs
    }), 200
if __name__ == '__main__':
    app.run(debug=True)

```

## Output:

```
PS C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code> c: cd 'c:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code' & 'C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\venv\Scripts\python.exe' 'c:\Users\BURRA SRUTHI\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65167' '--' 'C:\Users\BURRA SRUTHI\OneDrive\Desktop\3_2\AI\AI_code\Assign_5-2.py'
* Serving Flask app 'Assign_5-2'
* 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: 432-133-412
□
```



## Explanation:

### Key Improvements for Accessibility & Inclusiveness

1. **Inclusive Language**
  - Uses neutral, welcoming language (“All users are welcome,” “Anonymous” option).
  - Encourages users to share feedback in their own words.
2. **Accessibility Features**
  - Field for accessibility\_needs to capture accommodations.
  - Clear instructions for GET requests to help users using screen readers.
3. **Avoids Exclusionary Assumptions**
  - Name is optional → no pressure to disclose identity.
  - Rating validated within 1–5 range → prevents errors or bias.
  - Feedback text is required, but wording is open-ended → supports diverse responses.
4. **Browser-Friendly Instructions**
  - GET request returns HTML instructions for users who open the endpoint in a browser.