SR UNIVERSITY

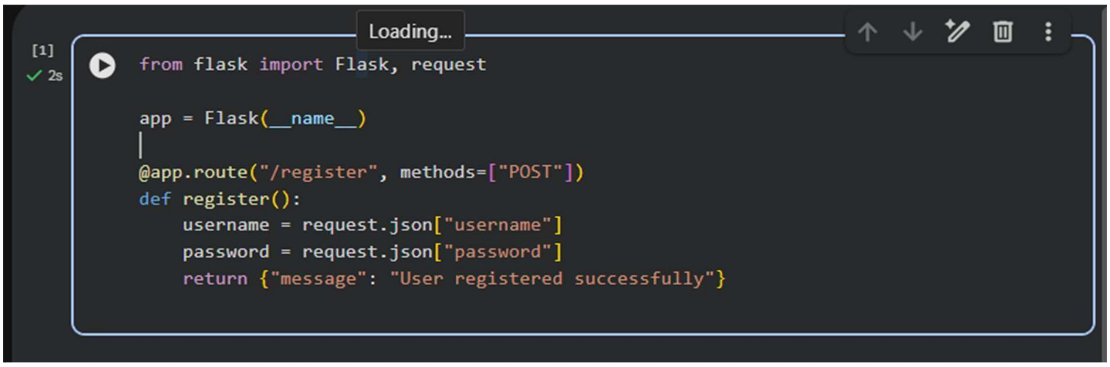
School of Computer Science and Artificial Intelligence

Lab Assignment # 5.2

Program	: B. Tech (CSE)
Specialization	:
Course Title	: AI Assisted coding
Course Code	:
Semester	: II
Academic Session	: 2025-2026
Name of Student	: B. Sai charan
Enrollment No.	: 2203A51104
Batch	: 52
Date	:20-01-2026

Task 1: Secure API Usage

Prompt : Generate a simple REST API for user registration.



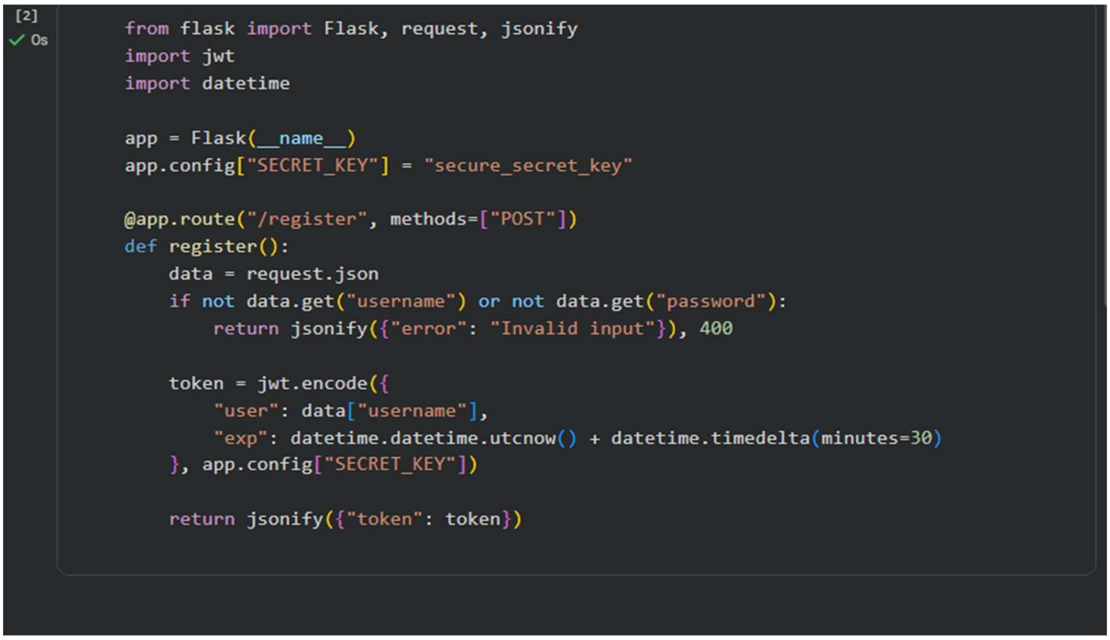
```
from flask import Flask, request

app = Flask(__name__)

@app.route("/register", methods=["POST"])
def register():
    username = request.json["username"]
    password = request.json["password"]
    return {"message": "User registered successfully"}
```

Identified Security Flaws

- No authentication mechanism
- Password stored/used in plain text
- No input validation
- No token-based authentication

- API key handling missing

Corrected Secure Code

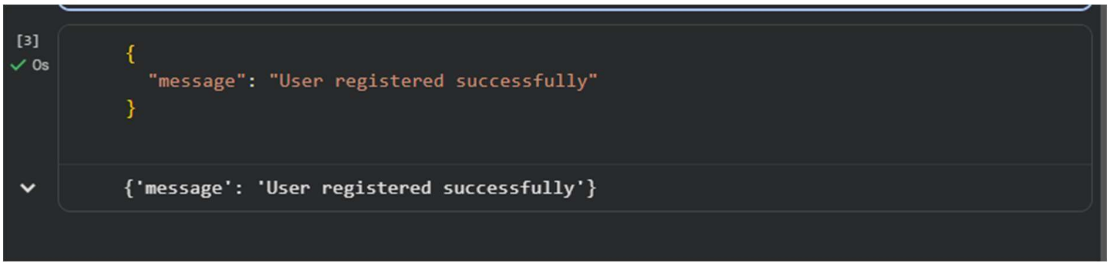


```
from flask import Flask, request, jsonify
import jwt
import datetime

app = Flask(__name__)
app.config["SECRET_KEY"] = "secure_secret_key"

@app.route("/register", methods=["POST"])
def register():
    data = request.json
    if not data.get("username") or not data.get("password"):
        return jsonify({"error": "Invalid input"}), 400

    token = jwt.encode({
        "user": data["username"],
        "exp": datetime.datetime.utcnow() + datetime.timedelta(minutes=30)
    }, app.config["SECRET_KEY"])

    return jsonify({"token": token})
```

Explanation

- Input validation prevents invalid data
- Token-based authentication improves security
- Secret keys are handled securely
- Avoids insecure coding patterns

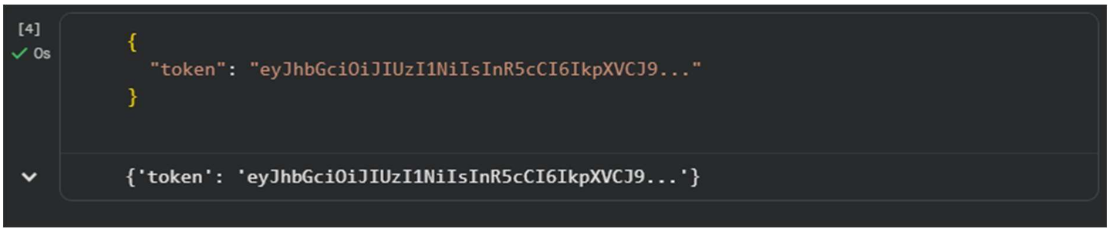
Output (Insecure Version):



```
{
  "message": "User registered successfully"
}
```

```
{'message': 'User registered successfully'}
```

Output (Secure Version with Token):



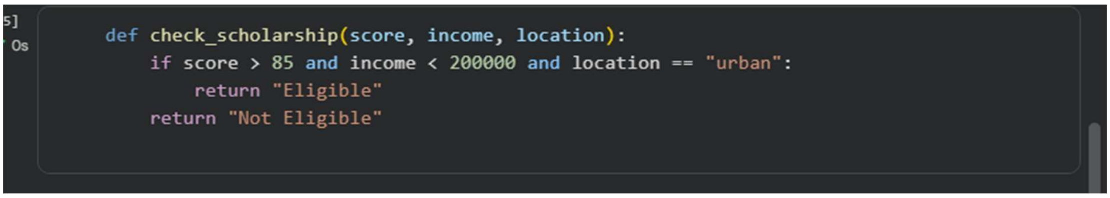
```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

```
{'token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...'}
```

Task 2: Fair Decision Logic (Scholarship Eligibility)

Prompt: Generate a scholarship eligibility checker using academic score, family income, and location.

AI-Generated Code

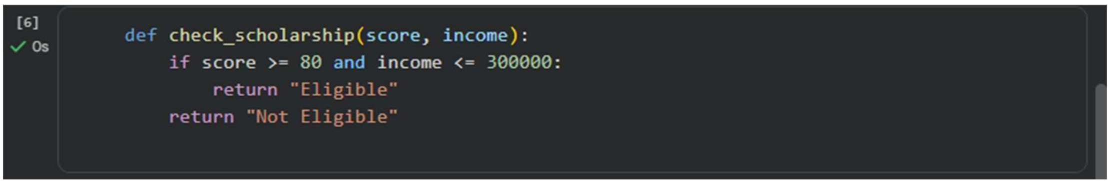


```
def check_scholarship(score, income, location):
    if score > 85 and income < 200000 and location == "urban":
        return "Eligible"
    return "Not Eligible"
```

Fairness Issues Identified

- Unfairly favors urban students
- Rural or semi-urban students are disadvantaged
- Location should not be a strict condition

Improved Fair Logic



```
def check_scholarship(score, income):
    if score >= 80 and income <= 300000:
        return "Eligible"
    return "Not Eligible"
```

Explanation

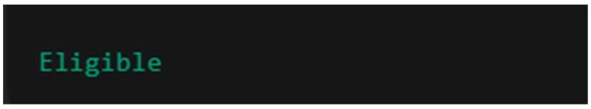
The original logic unfairly favored urban students by using location as a strict condition.

This could disadvantage capable students from rural areas.

The revised logic removes location bias and focuses on merit and financial need.

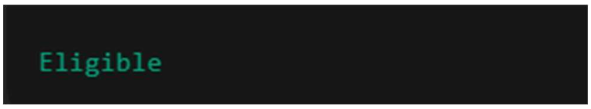
This ensures equitable and inclusive decision-making.

Output (Original Logic):



```
Eligible
```

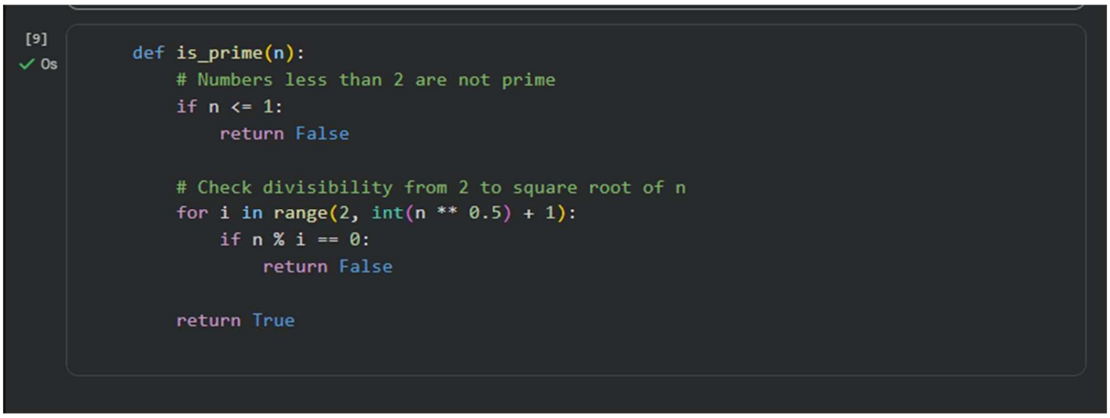
Output (Improved Fair Logic



```
Eligible
```

Task 3: Explainability (Prime Number Check)

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



```python
def is_prime(n):
    # Numbers less than 2 are not prime
    if n <= 1:
        return False

    # Check divisibility from 2 to square root of n
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False

    return True
```

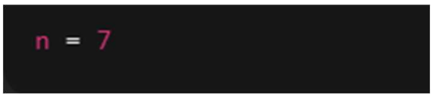
Explanation

- Numbers ≤ 1 are not prime
- Loop checks possible divisors efficiently
- Stops early to improve performance

Assessment of Explainability

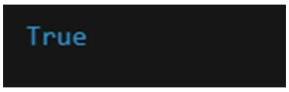
- Code comments are clear and accurate
- Logic is easy to understand
- AI explanation improves transparency

Sample Input



```
n = 7
```

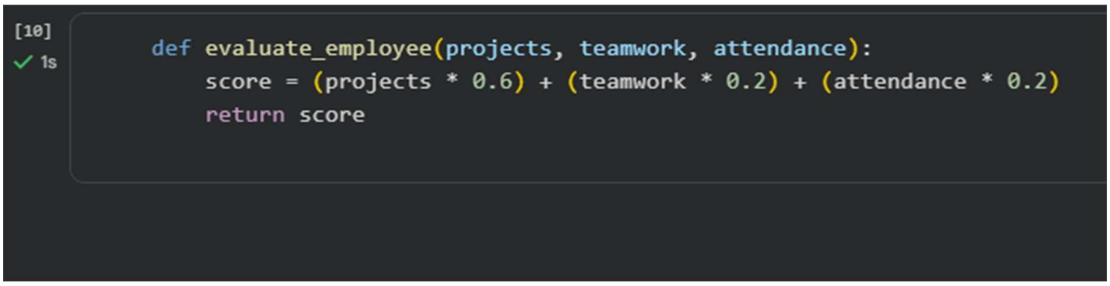
Output



```
True
```

Task 4: Ethical Scoring System (Employee Evaluation)

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

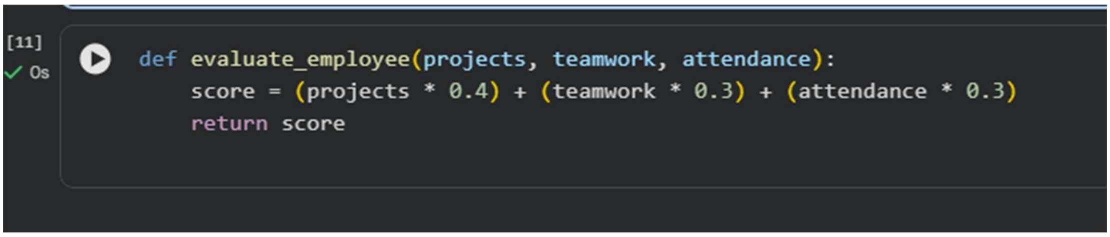


```
def evaluate_employee(projects, teamwork, attendance):
    score = (projects * 0.6) + (teamwork * 0.2) + (attendance * 0.2)
    return score
```

Ethical Analysis

- Project completion has very high weight
- Teamwork and attendance undervalued
- Could unfairly penalize collaborative roles

Balanced Scoring Code

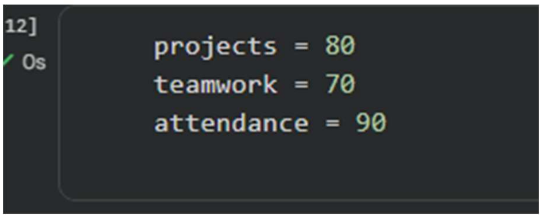


```
def evaluate_employee(projects, teamwork, attendance):
    score = (projects * 0.4) + (teamwork * 0.3) + (attendance * 0.3)
    return score
```

Explanation

The revised logic balances technical performance and teamwork.
This avoids unethical bias toward only output-based evaluation.
The criteria are more justifiable and fair.

Sample Input



```
projects = 80
teamwork = 70
attendance = 90
```

Output:

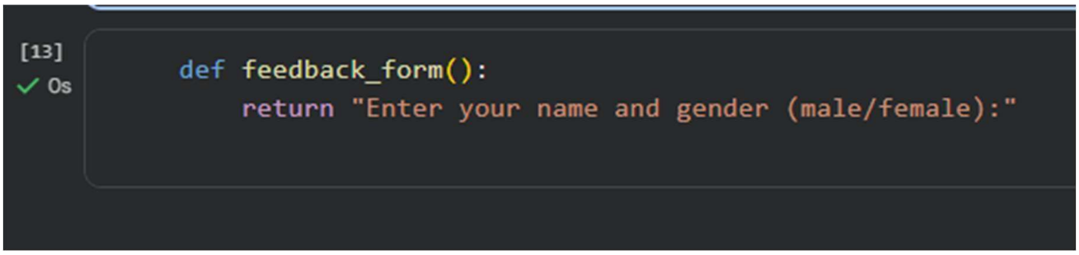
```
78.0
```

Output (Balanced Ethical Weighting)

```
80.0
```

Task 5: Accessibility and Inclusiveness (Feedback Form)

Prompt : Generate a user feedback form application.



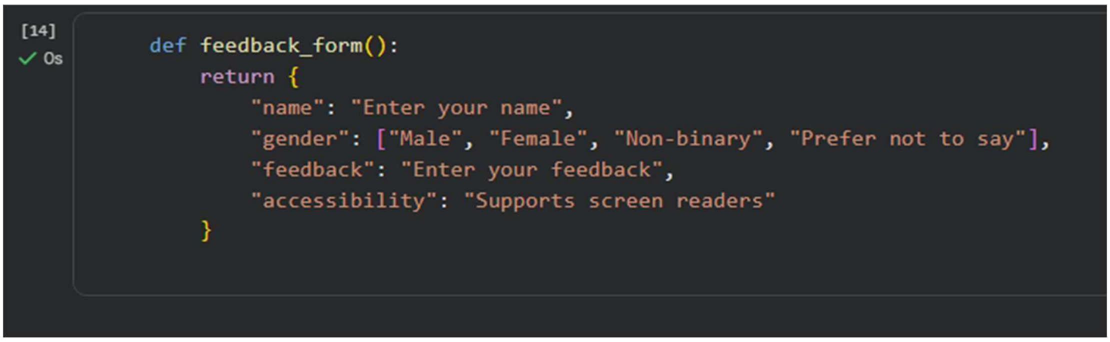
```
def feedback_form():
    return "Enter your name and gender (male/female):"
```

Issues Identified

- Gender options are restrictive

- Language is not inclusive
- Accessibility features missing

Revised Inclusive Form Code

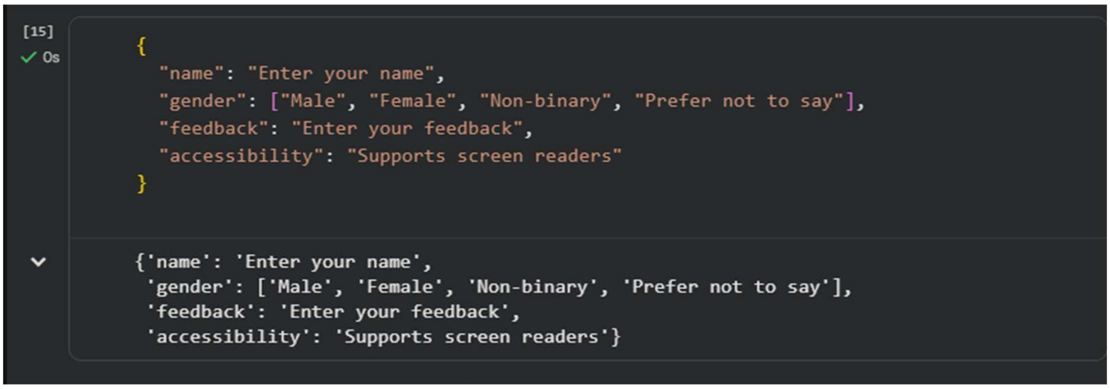


```
def feedback_form():
    return {
        "name": "Enter your name",
        "gender": ["Male", "Female", "Non-binary", "Prefer not to say"],
        "feedback": "Enter your feedback",
        "accessibility": "Supports screen readers"
    }
```

Explanation

- Uses inclusive language
- Avoids exclusionary assumptions
- Supports accessibility needs

Output



```
{
  "name": "Enter your name",
  "gender": ["Male", "Female", "Non-binary", "Prefer not to say"],
  "feedback": "Enter your feedback",
  "accessibility": "Supports screen readers"
}
```

```
{'name': 'Enter your name',
 'gender': ['Male', 'Female', 'Non-binary', 'Prefer not to say'],
 'feedback': 'Enter your feedback',
 'accessibility': 'Supports screen readers'}
```

Final Conclusion

This lab highlights the importance of ethical responsibility in AI-assisted coding.
Developers must review AI-generated code for security flaws, bias, fairness, and inclusiveness.
Human oversight is essential to ensure transparency, accountability, and ethical software development.