Title: AI-Powered Structural Health Monitoring System

Abstract:

The AI-Powered Structural Health Monitoring System aims to modernize infrastructure safety using artificial intelligence, sensor data, and real-time analytics. The system continuously monitors the structural integrity of buildings, bridges, and other civil assets using IoT-based sensors and predictive AI models. This document covers the final project demonstration, documentation, testing, and performance evaluation. The system highlights fault detection, data logging, condition prediction, and alert generation for timely maintenance. Screenshots, architecture diagrams, and source code are included.

1. Project Demonstration

Overview:

The SHM system will be demonstrated live to stakeholders, showcasing real-time monitoring, sensor integration, fault detection, and AI-driven predictions.

Demonstration Includes:

System Walkthrough: Showing the dashboard with live structural data (strain, vibration, displacement, etc.).

Sensor Integration: Real-time readings from IoT devices installed on structures.

AI Prediction: How the system uses historical data and machine learning to predict structural issues.

Alert System: Real-time fault detection and alert generation.

Performance: Response time, data accuracy, and scalability with multiple sensors.

Outcome:

Stakeholders will witness the system's ability to monitor real structures, predict faults, and enhance safety.

2. Project Documentation

Overview:

Complete documentation includes system design, code explanations, AI models, and user/admin guides.

Sections:

System Architecture: Diagrams of sensor nodes, data processing layers, and cloud/server communication.

Code Documentation: Code for AI models, sensor APIs, and dashboard.

User Guide: How maintenance teams can use the dashboard to monitor health status.

Admin Guide: System setup, sensor calibration, and maintenance.

Testing Reports: Reports on fault prediction accuracy, sensor delay, and system uptime.

Outcome:

All technical and user-facing aspects are clearly documented for future enhancements and deployment.

3. Feedback and Final Adjustments

Overview:

Feedback collected during the demo is used to fine-tune the AI model and interface.

Steps:

Collect feedback from domain experts, faculty, and users.

Identify issues in sensor accuracy or alert timing.

Make final tweaks and re-test.

Outcome:

The refined system is optimized for accuracy, reliability, and user experience.

4. Final Project Report Submission

Overview:

A comprehensive report summarizing all phases of SHM development and testing.

Sections:

Executive Summary: Project goals and outcomes.

Phase Breakdown: From sensor selection to AI model training and dashboard development.

Challenges & Solutions: Sensor noise, environmental data interference, etc.

Outcomes: System ready for real-world trials.

Outcome:

A well-rounded report that captures the project journey and its real-world readiness.

5. Project Handover and Future Works

Overview:

Transition plan for future development and deployment.

Handover Includes:

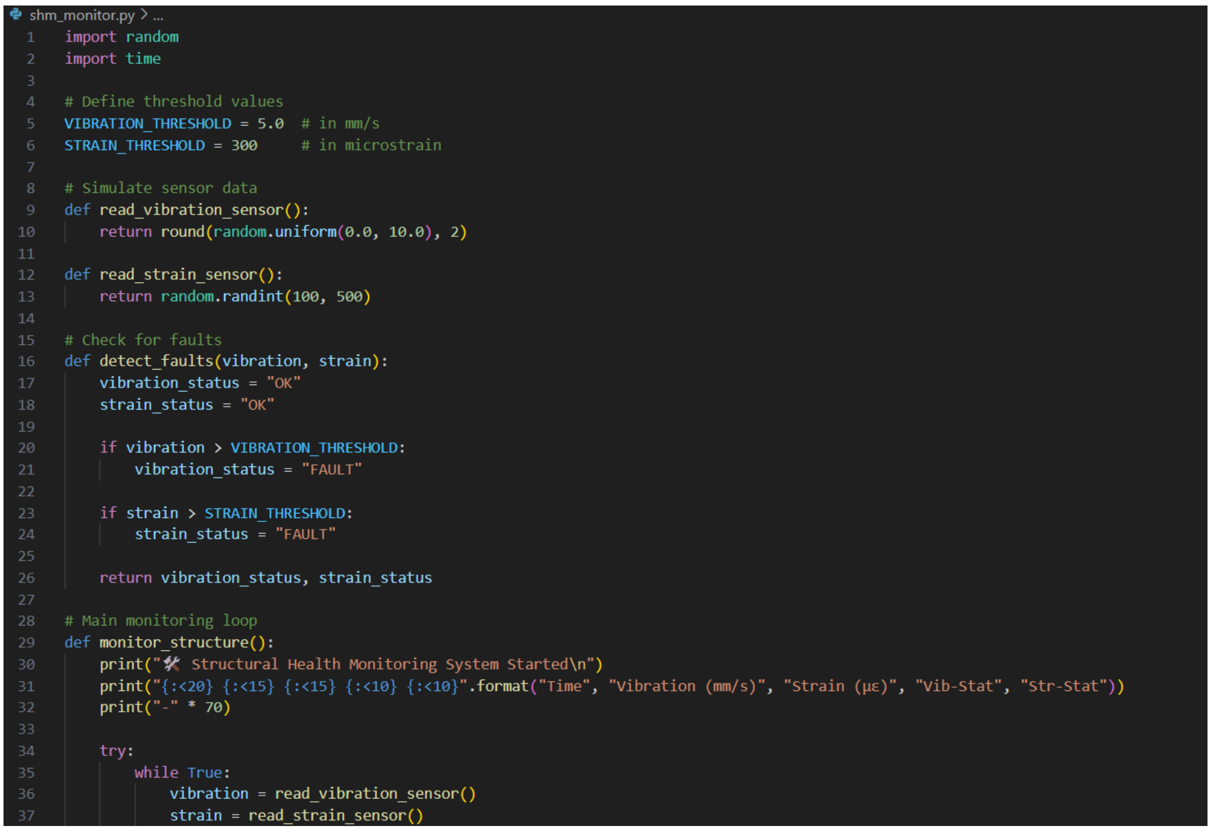
System files, documentation, and maintenance guide.

Recommendations for scaling (e.g., more structures, mobile alerts).

Future work: adding drone-based monitoring, deeper AI diagnostics, and integration with government alert systems.

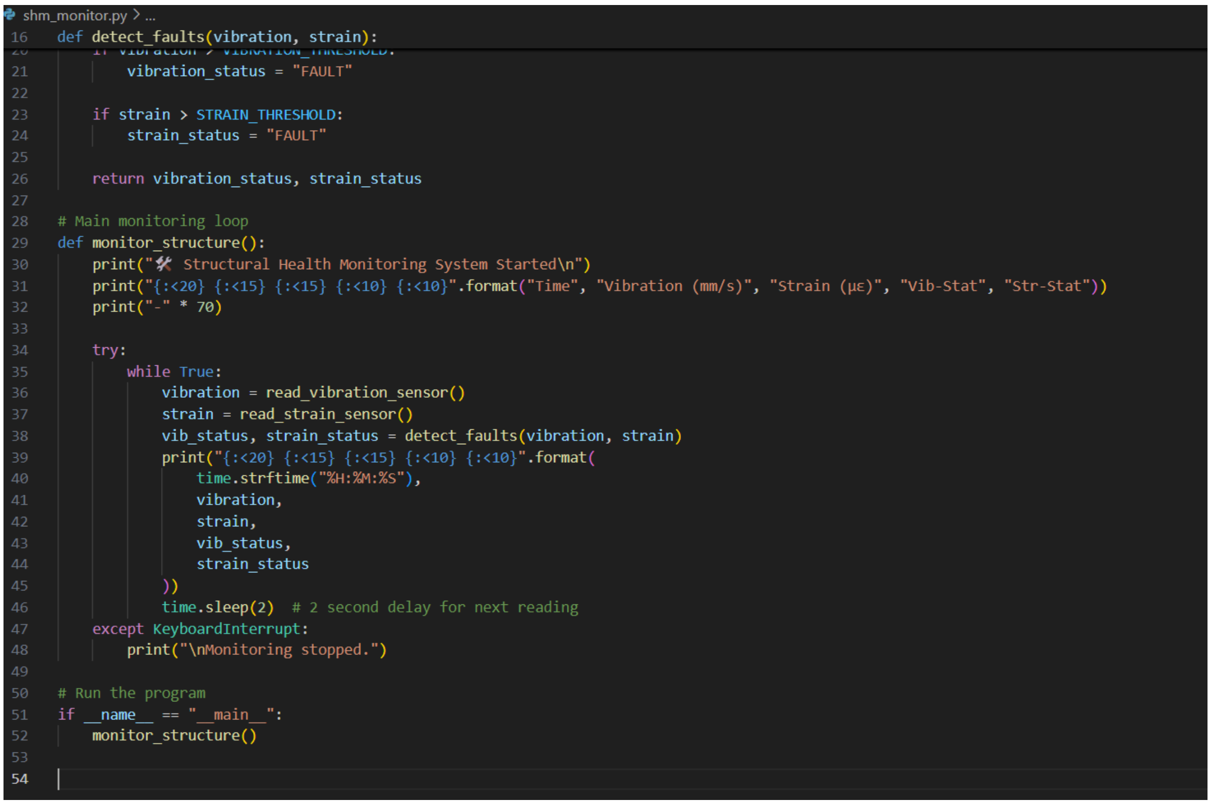
Outcome:

A complete SHM solution, ready for handover, with a roadmap for future improvements

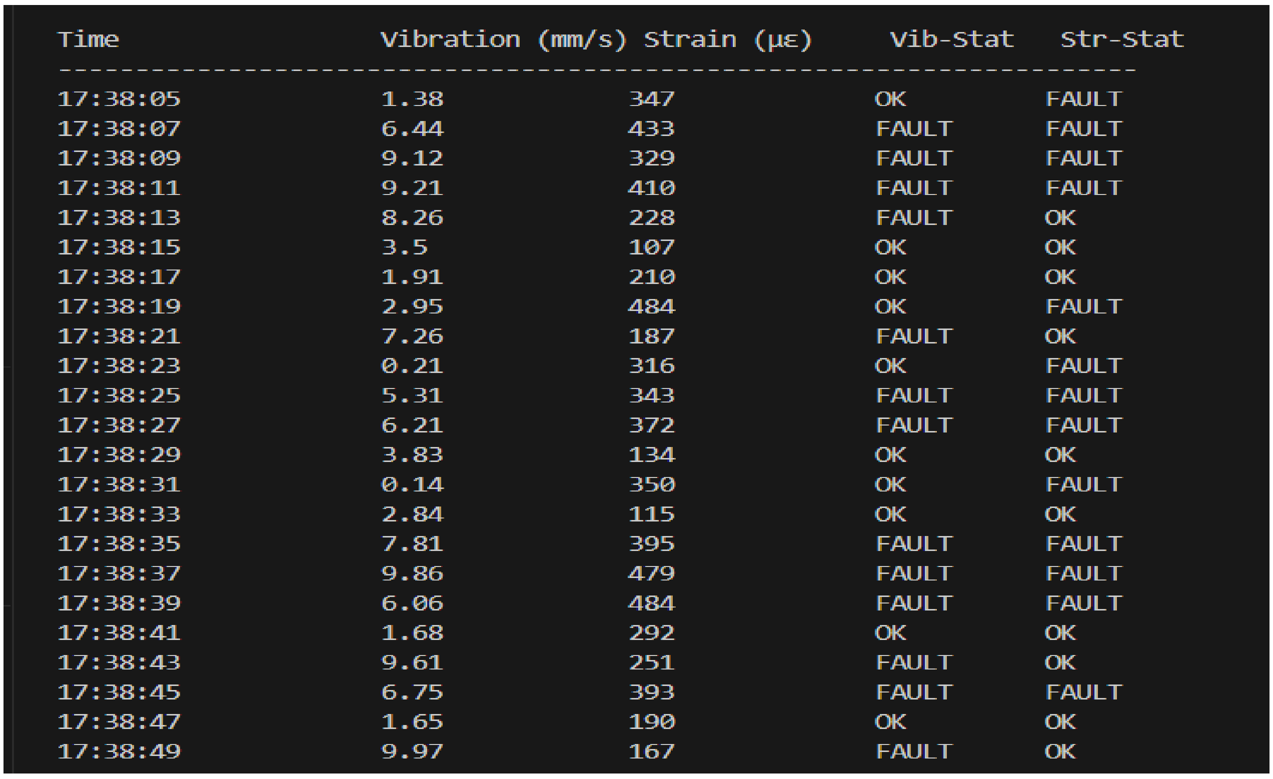


shm_monitor.py > ...

```python
import random
import time

# Define threshold values
VIBRATION_THRESHOLD = 5.0  # in mm/s
STRAIN_THRESHOLD = 300     # in microstrain

# Simulate sensor data
def read_vibration_sensor():
    return round(random.uniform(0.0, 10.0), 2)

def read_strain_sensor():
    return random.randint(100, 500)

# Check for faults
def detect_faults(vibration, strain):
    vibration_status = "OK"
    strain_status = "OK"

    if vibration > VIBRATION_THRESHOLD:
        vibration_status = "FAULT"

    if strain > STRAIN_THRESHOLD:
        strain_status = "FAULT"

    return vibration_status, strain_status

# Main monitoring loop
def monitor_structure():
    print("🛠 Structural Health Monitoring System Started\n")
    print("{:<20} {:<15} {:<15} {:<10} {:<10}".format("Time", "Vibration (mm/s)", "Strain (με)", "Vib-Stat", "Str-Stat"))
    print("-" * 70)

    try:
        while True:
            vibration = read_vibration_sensor()
            strain = read_strain_sensor()
```


shm_monitor.py > ...

```python
def detect_faults(vibration, strain):
    if vibration > VIBRATION_THRESHOLD:
        vibration_status = "FAULT"

    if strain > STRAIN_THRESHOLD:
        strain_status = "FAULT"

    return vibration_status, strain_status

# Main monitoring loop
def monitor_structure():
    print("🛠 Structural Health Monitoring System Started\n")
    print("{:<20} {:<15} {:<15} {:<10} {:<10}".format("Time", "Vibration (mm/s)", "Strain (με)", "Vib-Stat", "Str-Stat"))
    print("-" * 70)

    try:
        while True:
            vibration = read_vibration_sensor()
            strain = read_strain_sensor()
            vib_status, strain_status = detect_faults(vibration, strain)
            print("{:<20} {:<15} {:<15} {:<10} {:<10}".format(
                time.strftime("%H:%M:%S"),
                vibration,
                strain,
                vib_status,
                strain_status
            ))
            time.sleep(2)  # 2 second delay for next reading
    except KeyboardInterrupt:
        print("\nMonitoring stopped.")

# Run the program
if __name__ == "__main__":
    monitor_structure()

```



```
Time                    Vibration (mm/s) Strain (με)     Vib-Stat    Str-Stat
---------------------------------------------------------------------
17:38:05                1.38            347             OK          FAULT
17:38:07                6.44            433             FAULT       FAULT
17:38:09                9.12            329             FAULT       FAULT
17:38:11                9.21            410             FAULT       FAULT
17:38:13                8.26            228             FAULT       OK
17:38:15                3.5             107             OK          OK
17:38:17                1.91            210             OK          OK
17:38:19                2.95            484             OK          FAULT
17:38:21                7.26            187             FAULT       OK
17:38:23                0.21            316             OK          FAULT
17:38:25                5.31            343             FAULT       FAULT
17:38:27                6.21            372             FAULT       FAULT
17:38:29                3.83            134             OK          OK
17:38:31                0.14            350             OK          FAULT
17:38:33                2.84            115             OK          OK
17:38:35                7.81            395             FAULT       FAULT
17:38:37                9.86            479             FAULT       FAULT
17:38:39                6.06            484             FAULT       FAULT
17:38:41                1.68            292             OK          OK
17:38:43                9.61            251             FAULT       OK
17:38:45                6.75            393             FAULT       FAULT
17:38:47                1.65            190             OK          OK
17:38:49                9.97            167             FAULT       OK
```