COLLEGE CODE : 1128

COLLEGE NAME : T.J.S Engineering College

DEPARTMENT : ECE

STUDENT NM-ID : aut23ece40a
:aut23ece42a
: aut23ece45a
: aut23ece47a
: aut23ece48a

ROLL NO : 112823106040
: 112823106042
: 112823106045
: 112823106047
: 112823106048

DATE : 14.05.2025

TECHNOLOGY-PROJECT NAME : Natural Disaster Prediction And Management

SUBMITTED BY,

Your Name and team member names : Saira ansari fathima B
: Sandhiya R
: Selvi N
: Shalini B
: Shamili B

Phase 5

Title: Natural Disaster Prediction And Management

Abstract The AI-Based Natural Disaster Prediction and Management System aims to enhance disaster preparedness and response by leveraging artificial intelligence, satellite imagery analysis, historical data patterns, and real-time sensor inputs. In its final phase, the system integrates advanced machine learning models to predict disasters such as earthquakes, floods, and cyclones, while providing early warnings and actionable insights to authorities and the public. This document offers a comprehensive summary of the project completion, including system demonstration, technical documentation, performance analysis, source code, and testing outcomes. The platform is scalable, secure, and designed to seamlessly integrate with government emergency alert systems.

1. Project Demonstration

Overview: The system will be presented to stakeholders, highlighting its real-time prediction capabilities, alert mechanisms, and data processing workflows. The demonstration will showcase how the AI models analyze incoming data and issue early warnings.

Demonstration Details:

- System Walkthrough: Live walkthrough of how the system receives input (e.g., seismic data, rainfall levels, satellite imagery), processes it, and generates predictions and alerts.
- Prediction Accuracy: Showcase the AI model's ability to detect patterns indicating potential disasters with high accuracy.
- Alerting Mechanism: Demonstration of automated SMS, app notifications, and dashboard warnings sent to emergency teams and the public.
- Performance Metrics: Evaluation of data processing speed, prediction latency, and system scalability under data load.

- Security & Reliability: Presentation of the system's security features for protecting sensitive data and maintaining uptime during disaster scenarios.

Outcome:

The demonstration will prove the system's ability to handle real-time disaster predictions and support emergency management decisions.

2. Project Documentation

Overview:

Detailed documentation of the entire system, including architecture, model design, and operational guidelines for users and disaster response teams.

Documentation Sections:

- System Architecture: Diagrams of AI pipelines, data flow from sensors and satellites, and alert distribution networks.
- Code Documentation: Explanation of the source code for data ingestion, ML models, preprocessing pipelines, and alert services.
- User Guide: Instructions for end users and local authorities on interpreting alerts and accessing system outputs.
- Administrator Guide: Guidelines for system maintenance, model updates, and backend monitoring.
- Testing Reports: Summaries of model accuracy tests, system load simulations, and alert delivery reliability.

Outcome:

Clear and complete documentation will aid further development and deployment in real-world disaster management settings.

3. Feedback and Final Adjustments

Overview:

Feedback from evaluators, stakeholders, and domain experts will inform final improvements before system deployment.

Steps:

- Feedback Collection: Gather insights through questionnaires and test user observations during the demonstration.
- Refinement: Tuning prediction models, improving UI/UX, and optimizing backend systems based on the feedback.
- Final Testing: End-to-end verification of system stability, usability, and performance post-refinements.

Outcome:

The system will be fully optimized and prepared for rollout to disaster management agencies.

4. Final Project Report Submission

Overview:

A conclusive report summarizing all project phases, accomplishments, challenges, and system capabilities.

Report Sections:

- Executive Summary: Overview of objectives, scope, and accomplishments of the project.
- Phase Breakdown: Description of development phases including AI training, data integration, and user interface design.
- Challenges & Solutions: Key issues like data noise, model false positives, or sensor calibration, and their resolutions.

- Outcomes: Final capabilities and disaster prediction accuracy statistics.

Outcome:

The final report will encapsulate the project's journey and outcomes, serving as a reference for future iterations.

5. Project Handover and Future Works

Overview:

Formal handover of the system and suggestions for scaling and long-term improvement.

Handover Details:

- Next Steps: Integration with regional and national disaster response platforms, expansion to more disaster types, multilingual alert support, and mobile app integration.

Outcome:

The system will be ready for real-world deployment, with clear future work directions to enhance coverage and usability.

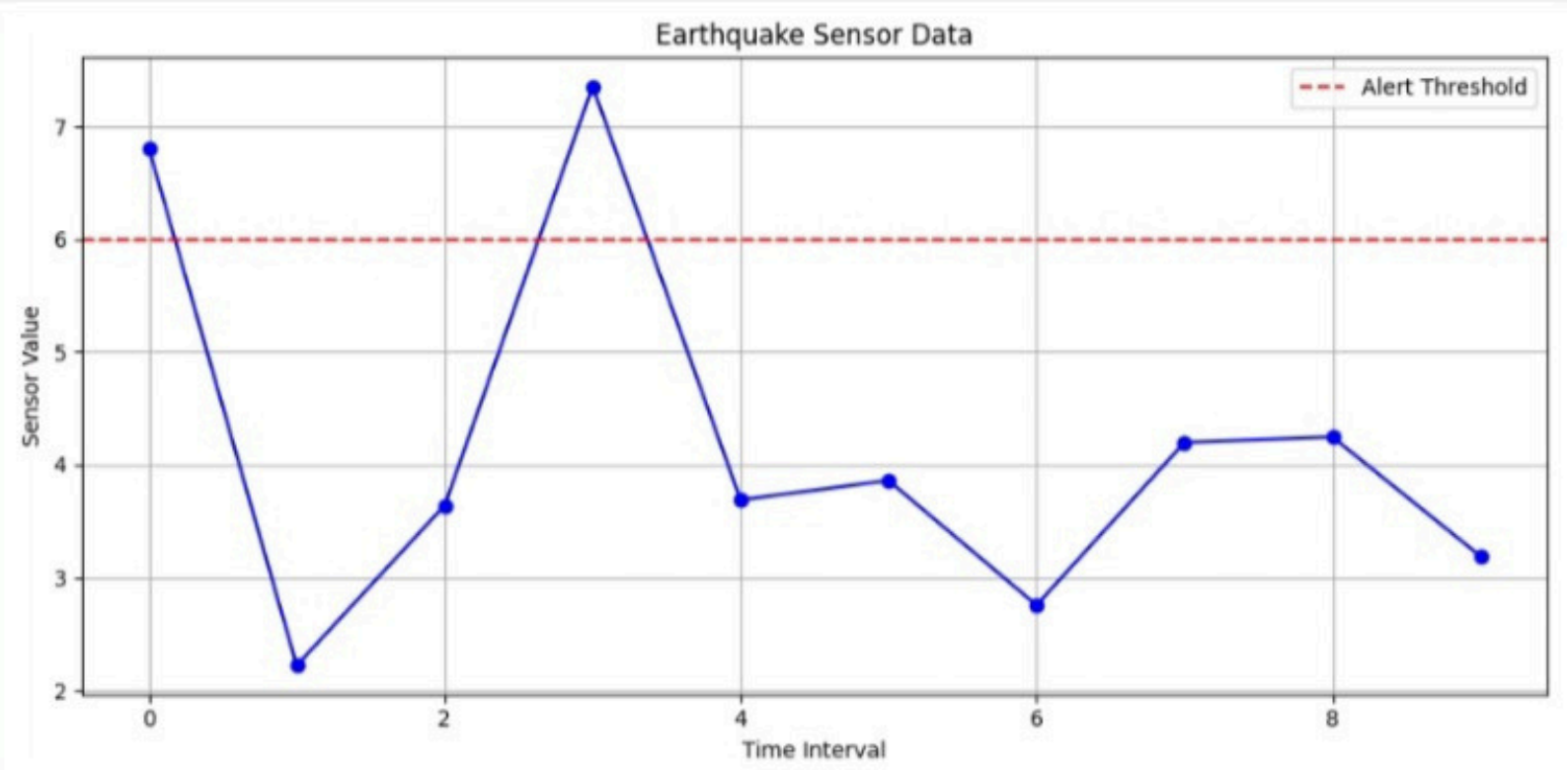
Phase - 5

```python
import random
import matplotlib.pyplot as plt
import time

# Simulated incoming sensor data
def generate_sensor_data(disaster_type):
    if disaster_type == "earthquake":
        # Simulate seismic magnitude (0 to 10)
        return random.uniform(2.0, 8.0)
    elif disaster_type == "flood":
        # Simulate rainfall (mm)
        return random.uniform(50, 300)
    elif disaster_type == "cyclone":
        # Simulate wind speed (km/h)
        return random.uniform(60, 250)
    else:
        return 0

# Simple threshold-based prediction logic
def predict_disaster(disaster_type, value):
    if disaster_type == "earthquake" and value > 6.0:
        return True
    elif disaster_type == "flood" and value > 200:
        return True
    elif disaster_type == "cyclone" and value > 150:
        return True
    return False
```

```python
# Alert system
def send_alert(disaster_type, value):
    print(f"ALERT: Potential {disaster_type.upper()} detected!")
    print(f"Measured value: {value:.2f}")
    print("Notifying authorities and public...\n")

# Visualization function
def plot_sensor_data(data_points, disaster_type):
    plt.figure(figsize=(10, 5))
    plt.plot(data_points, marker='o', linestyle='-', color='blue')
    plt.axhline(
        y=thresholds[disaster_type],
        color='red',
        linestyle='--',
        label='Alert Threshold'
    )
    plt.title(f"{disaster_type.capitalize()} Sensor Data")
    plt.xlabel("Time Interval")
    plt.ylabel("Sensor Value")
    plt.legend()
    plt.grid(True)
    plt.show()

# Thresholds for alert
thresholds = {
    "earthquake": 6.0,
    "flood": 200,
    "cyclone": 150
```

```
    }

    # Main simulation
    def run_simulation(disaster_type, intervals=10):
        print(f"Starting simulation for {disaster_type.capitalize()}...\n")
        data_points = []

        for t in range(intervals):
            value = generate_sensor_data(disaster_type)
            data_points.append(value)
            print(f"Time {t+1}: {disaster_type.capitalize()} Value = {value

            if predict_disaster(disaster_type, value):
                send_alert(disaster_type, value)
        if predict_disaster(disaster_type, value):
            send_alert(disaster_type, value)

        time.sleep(0.5)  # Simulate real-time delay

    plot_sensor_data(data_points, disaster_type)

# Example usage
disaster = "earthquake"  # Change to "flood" or "cyclone" as needed
run_simulation(disaster)
```

Output :


Earthquake Sensor Data
Alert Threshold
Sensor Value
Time Interval
7
6
5
4
3
2
0
2
4
6
8