Elder Health Monitoring & SOS Alert System

1. Introduction:

- Caring for the elderly presents an increasing challenge globally. Due, to the emergence of nuclear family setups and longer lifespans numerous older adults reside by themselves. Face dangers during urgent situations.
- This project offers an automated health surveillance system that mimics signs and notifies emergency responders when abnormalities are identified.
- The system is developed using Python and demonstrates how fundamental programming concepts can solve real-world problems effectively.

2. The Real-World Problem:

Many elders face difficulties such as:

- Heart-related medical emergencies
- High fever or infections
- Low oxygen levels
- Sudden falls or unconsciousness
- No immediate caretaker response

In most cases, the major issue is delay in medical attention.

A simple automated monitoring system helps reduce risk and can potentially saves the lifes

3. Objectives:

This project aims to:

- Simulate health data such as heart rate and temperature
- Check for abnormal ranges
- Generate SOS alerts instantly
- Provide a continuous, real-time monitoring loop
- Offer a beginner-friendly demonstration of a healthcare-based Python model

4. Expected Outcomes:

- After completing this project, users can:
- Understand sensor simulation using Python
- Learn the use of loops and conditional logic
- Recognize how alerts can be programmed
- Build a foundation for advanced health IoT systems

5. Concepts Used From Coursework :

- The project uses:
- Loops for continuous monitoring
- Functions for modular code
- Conditional Statements to detect unsafe vitals
- Random Module for simulated sensor values
- Time Module for delay and real-time behavior
- These concepts are core components of beginner-level programming courses.

6. Tools and Technologies:

- Python 3.x
- Libraries: `random`, `time`
- Text-based output (console UI)

7. Problem Definition:

- Elders require continuous monitoring, but manual observation is not practical.
- A system that automatically checks vitals and alerts during emergencies is essential.

8. Requirements Analysis:

Functional Requirements

- Generate heart rate
- Generate body temperature
- Compare values with thresholds
- Print output and warnings

- Trigger SOS alerts

Non-Functional Requirements

- Easy to operate
- Readable output
- Reliable detection
- Low resource usage

9. Top-Down Design:

The program uses five modules:

1. generate_vitals() – creates random health values.
2. check_thresholds() – evaluates readings.
3. display_output() – prints results.
4. trigger_sos() – shows alert message.
5. main_loop() – keeps simulation running.

10. Step-Wise Algorithm:

1.Start.

2.Generate vitals.

3.Print values.

4.Check thresholds.

5.If abnormal → SOS alert.

6.Else → continue.

7.Wait 1 second.

8.Repeat.

11. Flowchart Explanation:

Start → Generate Vitals → Check Thresholds →

Abnormal?

- Yes → Trigger SOS → Continue

- No → Continue

12. Implementation Summary:

Python-based logic that simulates a health monitoring IoT system without actual sensors.

13. Testing & Refinement

Multiple random ranges

- Threshold verification
- Long-duration continuous loop tests
- Ensuring no crashes

Improvements included better warnings and stable loops.

14. Features of the Project:

- Real-time simulation
- Automated health checks
- Simple operation
- Beginner-friendly coding technique
- Clean logic and modular functions

15. Detailed Workflow:

- Simulation: Uses random values to mimic sensors.
- Display: Shows vital signs clearly.
- Evaluation: Checks safe values.
- Alert: SOS displayed when needed.
- Loop: Repeats indefinitely.

16. Folder Structure :

project structure:

README.md

/src

main.py

/screenshots

/recordings

The main Python file that runs the program is located in the src folder and is named main.py

17. How to Operate Program :

Follow these steps to run Elder Health Monitoring Program:

Step 1 – Download or Clone Your GitHub Repository

Step 2 – Open the Project Folder

Step 3 – Navigate to the src Folder

Step 4 – Run the main.py File

Step 5 – Observe Real-Time Monitoring

- The console will display:
- Heart rate
- Temperature
- Health status (Normal / Warning / SOS)

Step 6 – SOS Alerts

Whenever vitals exceed thresholds, you will see a message like:

“SOS ALERT! Abnormal vitals detected!”

Step 7 – Stop the Program

(to stop the continuous loop)

18. Future Enhancements:

- Add oxygen monitoring.
- Integrate real sensors.
- Use SMS or call API.
- Create a mobile app.
- Store data on cloud.

19. Real-World Applications Useful for:

- Elders living alone
- Remote patient monitoring
- Home health IoT systems
- Hospitals
- Smart wearable devices

20. Importance for Learning Students learn:

- Modular design
- Real-world application of programming
- Understanding IoT fundamentals
- Project structuring and documentation

21. Conclusion:

The Elder Health Monitoring & SOS Alert System showcases how straightforward yet organized programming principles can tackle an important real-life issue—guaranteeing prompt assistance for senior citizens. Through the simulation of indicators and the automatic identification of irregularities the system offers a dependable ongoing and automated approach to tracking health status. The integration of SOS alerts emphasizes the value of quick action in emergencies minimizing the dangers linked to delays, in receiving medical care.

This project also showcases a strong application of core Python concepts such as functions, loops, conditionals, and modular coding, making it both educational and impactful. While the current model is simulated, it lays a strong foundation for future integration with real sensors, cloud databases, and mobile applications. Overall, this system reflects how basic programming skills can evolve into meaningful solutions that enhance safety, independence, and quality of life for elderly individuals.