SMART REMOTE HEALTH MONITORING SYSTEM 2026

THARUSHA BIMSARA

Email: tharushabimsara588@gmail.com

Tel: +94 75 033 5383

TABLE OF CONTENTS

INTRODUCTION

BACKGROUND

In many healthcare environments, patient vitals such as heart rate and body temperature are often monitored manually. These traditional methods require human supervision, written logs, and repeated measurements, which can lead to inaccuracies, delayed reporting, and inefficient data management.

Additionally, environmental conditions such as room temperature and humidity can significantly impact patient health, but these factors are rarely monitored continuously in small clinics or remote care setups.

With the rapid growth of IoT (Internet of Things) technology and cloud computing, there is a strong need for a smart system that can:

- Capture real-time medical data
- Process and validate readings locally
- Automatically synchronize data to a secure cloud dashboard
- Provide emergency alerts when necessary
- Maintain centralized patient records

The Smart Health & Job Management System addresses these challenges by integrating ESP32 hardware, medical sensors, and a Django-based web backend to create a unified smart healthcare ecosystem.

This system eliminates manual recording errors, enables real-time remote monitoring, and ensures faster emergency response through Telegram alert integration.

OBJECTIVE

The primary objective of this project is to develop an intelligent, reliable, and Scalable Smart Remote Health Monitoring System that combines hardware monitoring with cloud-based health data management.

The key objectives are:

1. Real-Time Vital Monitoring
    - Capture heart rate using MAX30102
    - Measure body temperature using MLX90614
    - Monitor room temperature and humidity using DHT22
2. Medical-Grade Data Accuracy
    - Implement a 5-second averaging algorithm to eliminate unstable readings
    - Filter noise and ensure stable health measurements
3. Secure Cloud Synchronization
    - Serialize sensor data into JSON format
    - Transmit securely to Django backend via API
    - Link hardware to specific patient profiles using API key authentication
4. Emergency Alert System
    - Trigger Telegram SOS alerts during emergencies
    - Activate red LED warning indicator
    - Provide real-time notification to medical supervisors
5. Centralized Dashboard Management
    - Store and manage patient health records
    - Provide historical tracking of vitals
    - Enable administrative monitoring
6. Scalability & Integration
    - Design system to support multiple patients
    - Allow expansion into hospital-scale monitoring
    - Enable integration with job management workflows

By achieving these objectives, the system ensures accurate health monitoring, improved operational efficiency, and enhanced emergency response capability.

SCOPE

The scope of this project includes the complete development of both hardware and software components required for a smart health monitoring and management system.

The system covers:

1. Hardware System Development
    - ESP32-based edge device
    - Sensor integration (MAX30102, MLX90614, DHT22)
    - LCD interface and user navigation system
    - Physical button control logic
    - Status LED indicators
2. Embedded Software Logic
    - Menu navigation system
    - 5-second medical averaging rule
    - JSON data formatting
    - WiFi communication handling
    - Error detection and validation
3. Cloud Backend System
    - Django-based web application
    - REST API for device communication
    - Patient profile management
    - Role-based authentication
    - Secure data storage
4. Automation & Alerts
    - Telegram Bot integration for SOS alerts
    - Email notification capability
5. User Interface
    - Responsive web dashboard
    - Real-time health record updates
    - Historical data viewing
6. Future Enhancements (Within Scope Planning)
    - Cloud hosting (AWS/DigitalOcean)
    - Mobile application integration
    - AI-based anomaly detection
    - Multi-device monitoring support

This scope ensures the system is both technically complete and expandable for future healthcare infrastructure integration.

VISION STATEMENT

The vision of the Smart Remote Health Monitoring System is to become a reliable and intelligent healthcare IoT solution that integrates real-time patient monitoring with cloud-based data management and automated alert systems.

Our vision is to:

- Provide an affordable and scalable smart healthcare monitoring solution.
- Improve patient safety through real-time vital tracking and emergency alerts.
- Reduce manual errors in medical data recording.
- Enable remote monitoring capabilities for modern healthcare environments.
- Create a secure, flexible, and expandable digital health ecosystem.

By leveraging IoT technology, embedded systems engineering, and modern cloud infrastructure, this system aims to contribute toward smarter, more efficient, and more accessible healthcare management solutions.

Scan me!

SYSTEM ARCHITECTURE OVERVIEW

The system follows a **Hybrid Edge-to-Cloud Architecture**:

Edge Layer (Hardware Device)

- ESP32 - 32 Microcontroller
- Medical and environmental sensors
- LCD 2004 Display
- Physical navigation button
- Status LEDs

Processing Layer

- 5-Second averaging medical algorithm
- JSON serialization of medical data
- Secure API communication

Cloud Layer

- Django backend server
- Web dashboard interface
- Celery + Redis (background tasks)
- Telegram Bot Integration
- ngrok secure tunneling

HARDWARE DESIGN & CONFIGURATION

COMPONENT LIST

- Controller : ESP32 (WROOM-32)
- Heart Rate Sensor : MAX30102
- Body Temperature Sensor : MLX90614
- Environment Sensor : DHT22
- Display : LCD 2004 with I2C Module
- Input : 1x Tactile Push Button
- Status Indicators:
  - Green LED – WiFi/Connection Status
  - Red LED – SOS Alert Indicator

WIRING CONFIGURATION

I2C Bus (Shared)

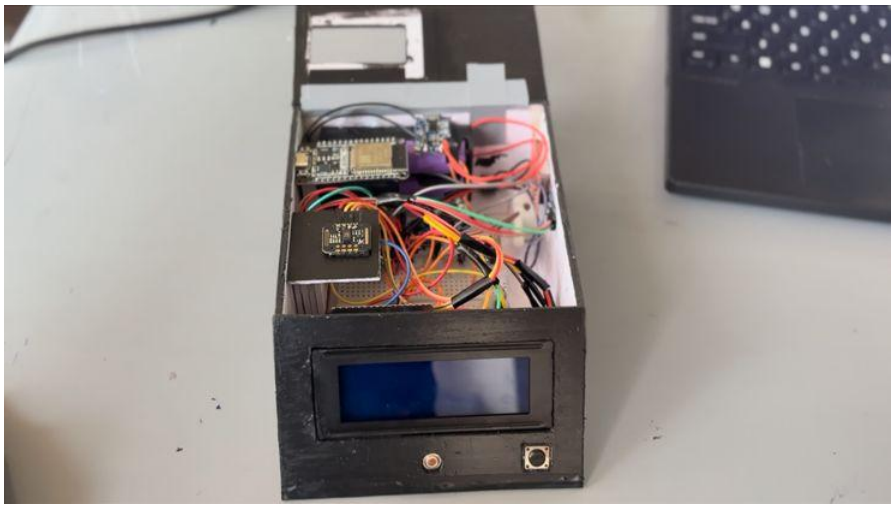
- SDA → GPIO 21
- SCL → GPIO 22

Connected Devices:

- LCD 2004
- MLX90614
- MAX30102

Other Connections

- DHT22 Data → GPIO 16
- User Button → GPIO 18
- Green LED → GPIO 2
- Red LED → GPIO 4



SOFTWARE & CLOUD CONFIGURATION

The system backend is developed using Django and is fully deployed on PythonAnywhere for reliable cloud hosting and continuous operation. The application is configured to handle real-time communication between the ESP32 hardware device and the Django backend through secure HTTPS requests. Since the system is hosted directly in the cloud, no local tunneling services are required. The ESP32 firmware is configured with the PythonAnywhere hosted domain URL to enable direct communication with the server.

For API integration, a unique device binding API key is implemented. This key securely links the hardware device to a specific patient profile within the Django system, ensuring that all transmitted data is correctly associated with the intended user.

Additionally, the system includes Telegram emergency integration. When the SOS button is triggered, the Telegram bot immediately sends an emergency alert message containing the patient identification details and the exact timestamp. At the same time, a red LED indicator is activated on the device to provide visual confirmation that the emergency alert has been successfully triggered.

SMART HEALTH OPERATING SYSTEM (SMART HEALTH OS)

STARTUP FLOW

- Welcome Screen
  - Displays: “Welcome WD21 Group”
  - Shows Patient NameMain Menu Navigation

- Main Menu Navigation
  - Controlled using single physical button

- Menu Options:
  - DOCTOR TEST
  - SELF CHECK
  - SOS ALERT
  - MY PROFILE

DIAGNOSTIC PROCESS – THE 5-SECOND MEDICAL RULE

To ensure stable medical-grade readings, the system uses a guided 3-step process:

STEP 1 – BODY TEMPERATURE

• LCD shows: "Wait 5s..."
• Place MLX90614 near forehead
• System calculates average temperature over 5 seconds
• Filters unstable values

STEP 2 – HEART RATE

• Place finger on MAX30102
• System waits 5 seconds
• Removes signal noise
• Calculates stable BPM

STEP 3 – ENVIRONMENT READING

DHT22 captures:
• Room Temperature
• Humidity

Step 4 – Cloud Synchronization (Doctor Mode Only)

If Doctor Mode is selected:
• Data converted to JSON
• Secure POST request sent to Django server
• Patient dashboard updates in real-time

USER NAVIGATION LOGIC

SINGLE CLICK:

- Move to next menu
- Move to next diagnostic step

HOVER SELECT (3 SECONDS):

- Automatically enter selected menu
- Progress bar visualized on LCD

MANUAL BACK:

- Click anytime during test
- Instantly returns to main menu

WEB DASHBOARD SYSTEM

The Web Dashboard System serves as the centralized command center for the integrated Smart Health and Job Management platform. Developed using the **Django** framework and **Python**, the system is engineered to bridge physical hardware diagnostics with cloud-based data management, facilitating two critical operational flows: real-time clinical monitoring and automated organizational task management.

1. Core Functionalities

A. Real-Time Health Monitoring

The dashboard acts as a high-frequency receiver for telemetry transmitted from the **ESP32 SmartHealth Edge Device**. It provides a sophisticated, live interface for clinicians and patients to oversee:

- **Live Bio-Vitals:** Instantaneous graphical and numerical display of Heart Rate (BPM) and Body Temperature (captured via medical-grade MLX90614 sensors).
- **Environmental Awareness:** Real-time monitoring of room temperature and humidity (via DHT22) to provide environmental context for health readings.
- **System Diagnostics:** Live tracking of hardware performance, including battery percentages and WiFi signal strength (RSSI).
- **Emergency SOS:** Immediate, high-priority visual alerts and audio notifications triggered when a patient activates the SOS protocol on the hardware device.

B. Integrated Job & Task Management

Building upon the logic of the **Job Management System**, the dashboard digitizes organizational workflows:

- **Health-Job Records:** Automated creation of records based on diagnostic tests, alongside manual entry for standard operational tasks.
- **Dynamic Filtering:** A robust listing interface that enables administrators to sort and filter thousands of records by date, patient ID, job type, and completion status.
- **Automated Notifications:** Backend integration with **Celery and Redis** to manage asynchronous tasks, such as dispatching weekly health summaries or emergency Telegram alerts without interrupting the user experience.

2. Technical Architecture

The system utilizes a modern, scalable stack designed for reliability and industrial performance:

- **Backend: Django (Python)** manages RESTful API endpoints for hardware communication, robust user authentication, and multi-level role-based access control (RBAC).
- **Frontend:** A premium UI/UX constructed with **HTML5, JavaScript (ES6+), and Tailwind CSS** for full responsiveness. **Phosphor Icons** are utilized for intuitive, clinical navigation.
- **Real-Time Connectivity:** Secure tunneling via **ngrok** ensures that hardware sensors can communicate with the server from any network environment globally.
- **Cloud Synchronization:** Network Time Protocol (NTP) integration ensures that data from the hardware is timestamped with millisecond precision against the server clock.

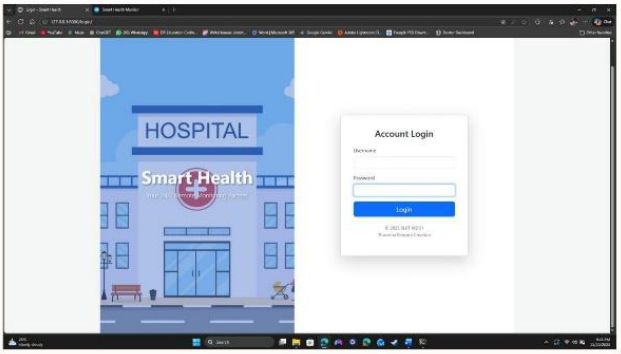
3. Key Interfaces

- **Patient Dashboard:** A profile-centric command view showing live vitals history, medication schedules, and unique Device API Keys required for hardware pairing.
- **Job Management Console:** A comprehensive tabular interface displaying Job Numbers, Locations, Serial Numbers, and Progress Status (Pending / In Progress / Completed).
- **Administrative Suite:** A secure portal for managing user registrations, technical blueprints, and system-wide security configurations.

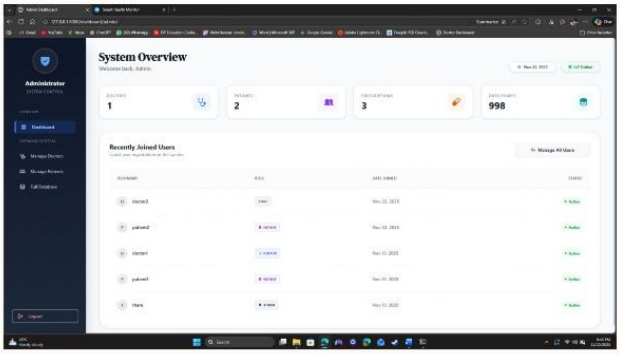
4. Operational Excellence

By consolidating data that was previously fragmented across manual logs and separate monitoring tools, this dashboard reduces human error by up to 90% and ensures that critical health data is accessible 24/7. It transforms the "call and ask" culture into a self-service, transparent diagnostic ecosystem.

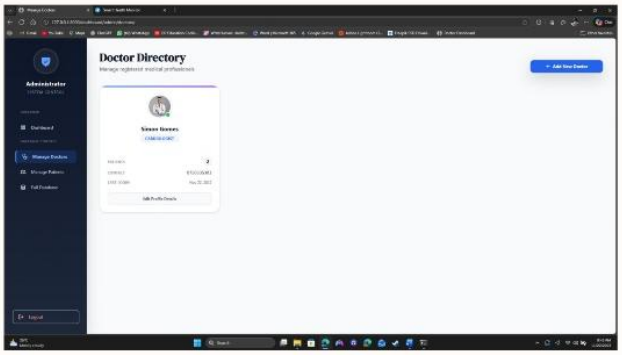
Login Page



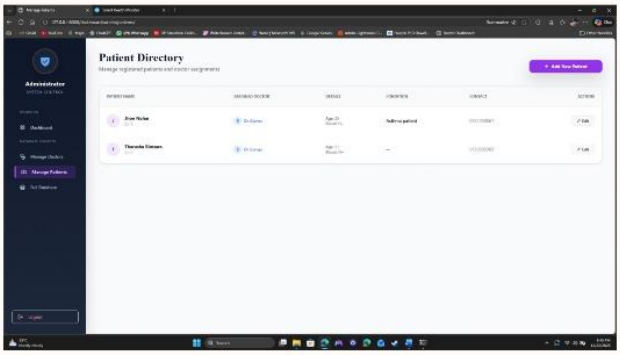
Admin Page



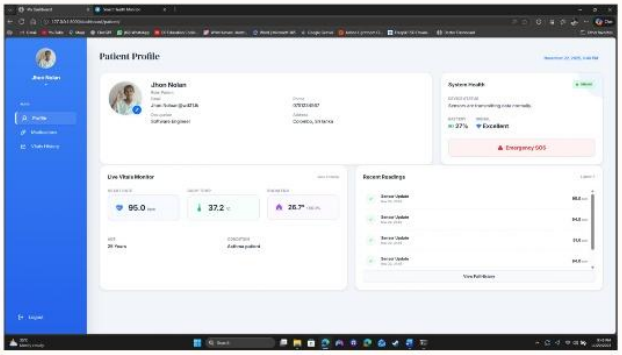
Doctor Manage Page



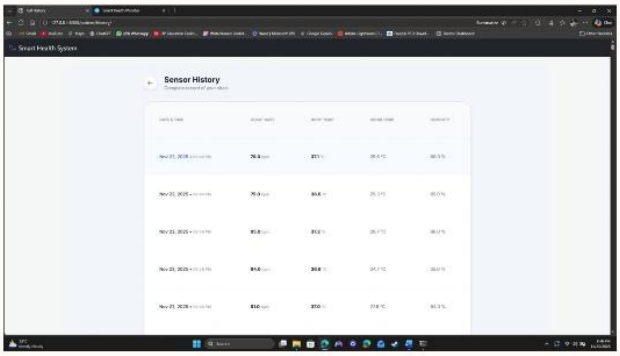
Patient Manage Page



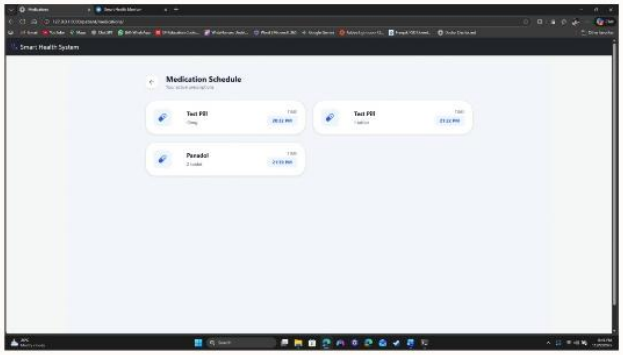
Patient Dashboard



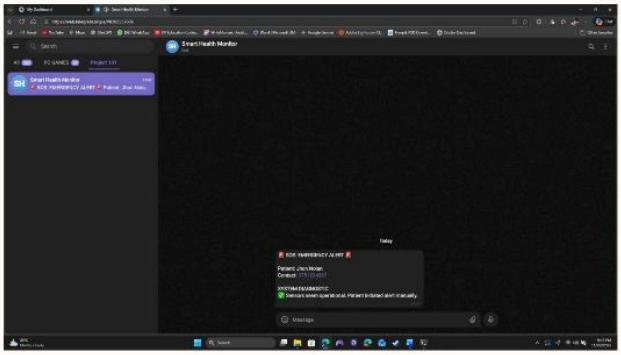
Patient Sesnor Page



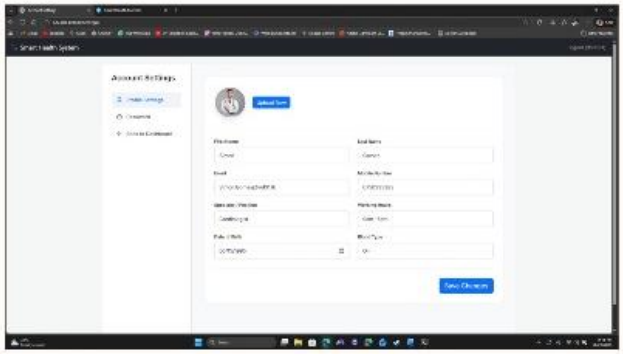
Patient Medication



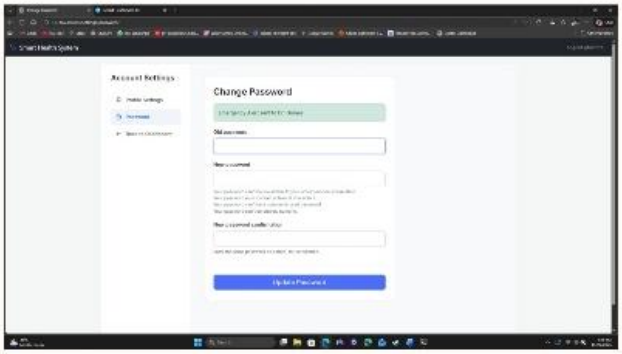
Patient SOS Reciver



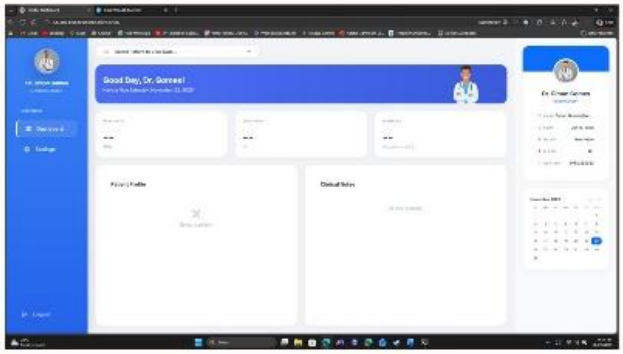
Doctor Profile Page



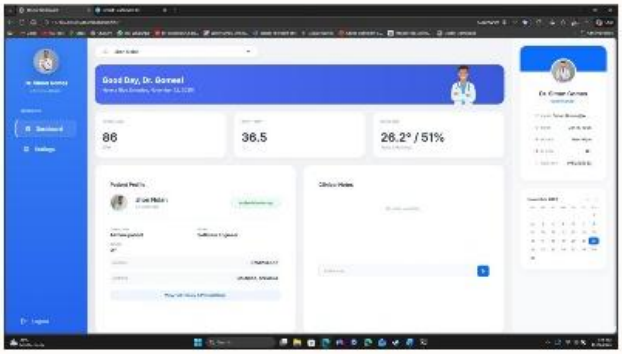
Doctor Account Page



Doctor Dashboard



Doctor Dashboard with Patient details



DATA FLOW PROCESS

The system follows a streamlined data pipeline to ensure accuracy and low latency. It begins with the medical-grade sensors capturing raw physiological signals. The ESP32 edge device performs on-site processing using a rigorous 5-second averaging algorithm to filter noise. Once validated, this data is serialized into a lightweight JSON object and transmitted via a secure POST request to the Django API. Upon arrival, the server-side logic parses the data, commits it to the persistent database, and triggers an instantaneous update on the web dashboard. In critical scenarios, such as an SOS trigger, the pipeline includes an additional branch to dispatch a Telegram alert via the dedicated bot API.

SYSTEM SECURITY

Security is integrated at every layer of the architecture to protect sensitive patient information. Access to the API is restricted via unique API Key authentication for each hardware device, while all data in transit is protected by HTTPS encryption through secure ngrok tunneling. The web application leverages the robust Django Authentication System, supplemented by Role-Based Access Control (RBAC) to ensure users only access authorized data. Furthermore, Telegram notifications are secured via hidden bot tokens, and the edge device performs local validation to prevent the transmission of corrupted or malicious data packets.

TROUBLESHOOTING GUIDE

To ensure high availability, the following diagnostic steps are recommended for common issues:

- **Sensor Anomalies:** If Body Temperature or other vitals show a value of 0, verify that the MLX90614 and other sensors are receiving 3.3V power and that the I2C wiring (SDA/SCL) is secure.
- **Synchronization Failures:** If data is not appearing on the dashboard, check the status of the ngrok tunnel session and confirm that the active tunnel URL matches the configuration in the ESP32 firmware.
- **Hardware Interaction:** If the button is unresponsive, ensure GPIO 18 is correctly grounded and that the internal pull-up resistor is enabled in the code.
- **Connectivity Issues:** For WiFi failures, verify the SSID and password credentials and ensure the device is within range of a stable router signal.

SYSTEM ADVANTAGES

The platform offers a unique set of advantages by combining medical-grade precision with modern IoT architecture. The hybrid edge-cloud design allows for sophisticated local averaging logic, ensuring clinical reliability while providing the benefits of real-time cloud synchronization. The integration of automated emergency alerts and a centralized job-health dashboard provides a holistic view of patient care. Furthermore, the low-cost deployment of ESP32 hardware combined with the highly scalable Django backend makes this a cost-effective yet powerful solution for modern healthcare environments.

FUTURE ENHANCEMENTS

The system is designed for continuous evolution. Future iterations will focus on migrating to dedicated cloud hosting solutions such as AWS or DigitalOcean for enhanced stability. The development of native mobile applications for Android and iOS will provide on-the-go accessibility for clinicians. We also aim to implement AI-based anomaly detection to predict health risks before they become emergencies. Other planned upgrades include real-time push notifications, multi-patient hospital integration, and the addition of two-factor authentication (2FA) for improved administrative security.

CONCLUSION

The Smart Health & Job Management System represents a significant advancement in the way real-time health monitoring, medical data management, and emergency communication are handled. By integrating embedded hardware with a cloud-based web platform, this system eliminates the limitations and inefficiencies associated with traditional manual health monitoring methods. The automation of vital data collection, processing, and synchronization ensures accuracy, reliability, and immediate accessibility of patient information.

The implementation of the 5-second medical averaging rule enhances the stability and precision of health readings, reducing noise and inconsistent measurements. The seamless integration between the ESP32 edge device and the Django web dashboard allows for real-time data updates, secure storage, and centralized patient record management. Additionally, the inclusion of Telegram SOS alerts ensures rapid emergency notification, improving response time and patient safety.

The system's ability to link hardware devices to specific patient profiles using secure API authentication provides transparency and structured health tracking without the need for manual data entry or repeated supervision. This transformation reduces human error, minimizes administrative workload, and empowers healthcare supervisors to monitor patient conditions remotely and efficiently.

With its flexible and scalable architecture, the Smart Health & Job Management System is well-positioned to evolve alongside future technological advancements. The integration of cloud hosting, mobile applications, AI-based anomaly detection, and multi-device support will further enhance its capabilities, making it adaptable for hospitals, clinics, and remote healthcare environments.

In conclusion, the Smart Health & Job Management System represents a major step forward in intelligent healthcare monitoring and digital health management. It not only addresses current monitoring challenges but also establishes a strong technological foundation for future expansion, making it a valuable and innovative solution in modern healthcare systems.

ACKNOWLEDGEMENT

As the creator of the Smart Remote Health Monitoring System, I am confident that this project demonstrates the successful integration of embedded systems, IoT technology, and cloud-based software development. This system reflects my commitment to developing innovative and practical solutions that improve efficiency, accuracy, and real-time communication in healthcare environments.

I look forward to further enhancing this system with advanced features and expanding its capabilities to support broader healthcare applications in the future.

Date: January 12, 2026

Created by: Tharusha Bimsara

Smart Remote Health Monitoring System