CAPSTONE PROJECT REPORT

PROJECT TITLE

Automated Health Monitoring System

TEAM MEMBERS

P.Dinesh kumar reddy(192210192)

B.Durga Prasad(192210417)

COURSE CODE / NAME

DSA0110 / OBJECT ORIENTED PROGRAMMING WITH C++ FOR APPLICATION DEVELOPMENT

SLOT A

DATE OF SUBMISSION

12.11.2024

BONAFIDE CERTIFICATE

Certified that this project report _____TITLE_____

is the bonafide work of __ REG NUMBER AND NAME OF STUDENT(S) _____

_____who

carried out the project work under my supervision.

SUPERVISOR

ABSTRACT

The Automated Health Monitoring System (AHMS) is a cutting-edge solution designed to continuously track, analyze, and report an individual's health metrics in real-time. This system integrates wearable sensors and medical devices to monitor vital signs such as heart rate, body temperature, blood pressure, oxygen saturation (SpO2), and more.

The collected data is transmitted seamlessly to a mobile application or cloud platform, enabling users to view and manage their health information at any time. Through real-time data analysis, the system can detect anomalies, identify health trends, and send alerts if any critical values are outside the normal range, thereby providing early warnings for potential health risks.

Additionally, the system supports remote monitoring by healthcare providers, allowing for proactive intervention and personalized care. With its user-friendly interface and robust data analytics capabilities, the AHMS empowers individuals to take control of their health and facilitates improved healthcare outcomes.

This system enhances preventive healthcare by enabling early detection of health issues, improving health outcomes, and fostering a proactive approach to wellness. Additionally, it reduces the burden on healthcare facilities by facilitating continuous, remote health tracking.

INTRODUCTION

With the rapid advancement of wearable technology and the increasing demand for personalized healthcare solutions, Automated Health Monitoring Systems (AHMS) have emerged as an innovative approach to managing individual health in real time. These systems leverage a combination of wearable sensors, data analytics, and mobile applications to monitor critical health parameters continuously, enabling individuals to track their wellness and detect potential health issues before they escalate. By providing instant feedback on key health metrics such as heart rate, blood pressure, body temperature, blood oxygen levels, and more, AHMS empowers users to make informed decisions about their health and lifestyle.

Traditionally, healthcare monitoring has been reactive—individuals seek medical attention only when they experience symptoms or health problems. However, with the introduction of continuous, real-time health tracking, AHMS offers a shift toward preventive healthcare, enabling users to stay ahead of potential health risks. This system not only improves the quality of life by offering insights into long-term health trends but also reduces the burden on healthcare systems by enabling remote monitoring and early intervention.

The integration of advanced data analytics and cloud-based platforms allows for seamless communication between users and healthcare providers. Health data is securely transmitted to mobile apps or cloud systems, where it is analyzed for any deviations from normal ranges. When any irregularities or health risks are detected, alerts are sent to both the user and healthcare professionals, allowing for timely action. Furthermore, the mobile interface provides users with a user-friendly platform to monitor, track, and manage their health goals, making it an essential tool for individuals with chronic conditions or those seeking to maintain a healthy lifestyle.

LITERATURE REVIEW

The concept of health monitoring systems has evolved significantly over the past few decades, particularly with the advancement of wearable technology and the increasing prevalence of chronic diseases. Automated Health Monitoring Systems (AHMS) combine wearable devices, real-time data analysis, and cloud-based technologies to provide continuous tracking of vital health metrics, offering individuals and healthcare providers an innovative way to manage health more proactively. This literature review examines key developments, challenges, and applications of AHMS, focusing on the technologies, methodologies, and outcomes from previous studies and systems.

Wearable devices, such as fitness trackers, smartwatches, and medical-grade sensors, have become widely adopted tools for health monitoring. Initially, wearable devices were limited to basic tracking functions like step counting and heart rate measurement. However, with advancements in sensor technology and miniaturization, modern wearables now monitor a broader range of physiological parameters, including blood pressure, oxygen saturation (SpO2), glucose levels, and even electrocardiogram (ECG) signals (Patel et al., 2012). These devices are increasingly integrated with mobile applications to provide real-time health data to users, making it easier to track and manage health metrics on a daily basis (Jovanov et al., 2003).

RESEARCH PLAN

The objective of this project is to develop an automated health monitoring system to track and manage patient vitals, such as heart rate, blood pressure, and temperature, in real-time. The system will leverage IoT devices to gather data from various sensors and use C++ for data parsing, control, and visualization. Data will be stored in CSV files and visualized in a user-friendly interface, while simulations will be conducted using Cisco Packet Tracer to emulate IoT network communication.

SL. No	Description	07/10/2024-11/10/2024	12/10/2024-16/10/2024	17/10/2024-20/10/2024	21/10/2024-29/10/2024	30/10/2024-05/11/2024	07/10/2024-10/11/2024
1.	Problem Identification						
2.	Analysis						
3.	Design						
4.	Implementation						
5.	Testing						
6.	Conclusion						

Development Phases and Timeline

Day 1: Project Initiation and Planning

- **Define Scope & Objectives**: Outline the system's purpose—to provide continuous health monitoring with real-time vitals feedback, alerts, and control functionalities.
- **Preliminary Research**: Investigate IoT-based health monitoring devices, relevant protocols (e.g., Bluetooth, Wi-Fi), and industry standards for medical data security.
- **Identify Key Stakeholders**: Engage with stakeholders, including healthcare professionals, developers, and security experts, to gather insights and establish collaboration channels.
- **Create Project Plan**: Develop a comprehensive task and milestone timeline, ensuring clear understanding and alignment among all team members regarding project expectations.

Day 2: Requirement Analysis and System Design

- **Conduct Requirement Analysis**: Identify user needs, essential features, and automation capabilities. Determine specific vitals to monitor (e.g., heart rate, blood pressure, body temperature) and data collection frequency.
- **Define System Specifications**: Set parameters for device compatibility, communication protocols, and data security requirements to comply with healthcare standards.
- **Develop System Architecture**: Design a modular system architecture with components for device integration, data handling (using CSV files), and a C++-based control module.
- **Confirm Hardware/Software Requirements**: Ensure the chosen hardware (sensors, routers, etc.) and software (Packet Tracer, C++ libraries) align with system needs.

Day 3: Core Development and Device Integration

- **Begin Core Coding**: Start development of primary modules, including:
    - **Sensor Data Collection**: Functions to read data from simulated sensors in Packet Tracer.
    - **Data Handling**: Code to parse CSV files, process real-time data, and update patient vitals.
    - **Device Control**: Implement control mechanisms for setting data collection intervals and alert thresholds.
- **Set Up Communication Protocols**: Enable communication between C++ modules and IoT devices for reliable data transmission.
- **Security Protocols**: Integrate encryption and data protection measures for sensitive health data.
- **Unit Testing**: Test each component for functionality, including accurate data collection and storage.

Day 4: GUI Design and Prototyping

- **Design User Interface**: Develop a user-friendly GUI in C++ (possibly using Qt) to allow caregivers and medical staff to monitor vitals and manage alerts easily.
- **Real-Time Data Visualization**: Incorporate graphical data elements (e.g., charts and bar graphs) to display vitals like heart rate, temperature, and blood pressure.
- **Interface Testing and Refinement**: Test GUI for responsiveness and usability across different devices (e.g., desktops, tablets) and refine based on feedback.

Day 5: Automation Scenarios and Testing

- **Define Automation Scenarios**: Establish predefined actions (e.g., alert notifications when vitals exceed limits, automatic logging intervals).
- **Implement Alert System**: Add functionality to notify medical staff or caregivers via SMS/email when irregular readings are detected.
- **Test Automation and Predictive Logic**: Integrate machine learning models or rules-based logic for smart alerts and predictive monitoring.

- **System Testing and Optimization**: Test the complete system under various scenarios to ensure accuracy and responsiveness in data handling and alerts.

Day 6: Documentation, Deployment, and Feedback

- **Document Development Process**: Record system architecture, key decisions, and coding structures. Include guidelines for setting up and configuring the system.
- **Deployment Preparation**: Ensure the system meets healthcare standards for data privacy and security before deployment.
- **Beta Testing and Stakeholder Feedback**: Deploy the system for beta testing and hold feedback sessions with stakeholders to gather insights and identify potential areas for improvement.

METHODOLOGY

The design and development of an Automated Health Monitoring System (AHMS) involves several stages, including the selection of wearable sensors, data collection methods, data transmission, analysis techniques, and integration with mobile applications or cloud platforms.

1. System Design Overview

The methodology focuses on creating an AHMS that can track multiple health parameters such as heart rate, body temperature, blood pressure, oxygen saturation (SpO2), and respiratory rate, and transmit the data to a mobile application or cloud-based server for real-time analysis. The system design can be divided into three key components:

Wearable Sensors: To measure health metrics continuously.

Data Transmission and Storage: To transmit collected data to a mobile or cloud application for storage and analysis.

Mobile/Cloud Application: To display the health data to users and healthcare professionals and perform real-time data analytics.

2. Sensor Selection and Integration

The first step in building the AHMS is selecting appropriate sensors for monitoring various health parameters. The sensors must be accurate, non-invasive, and capable of providing continuous data over extended periods. Common sensors include:

Heart Rate Sensor: A photoplethysmogram (PPG) sensor or an ECG sensor is used to measure heart rate.

Temperature Sensor: Infrared or thermistor-based sensors are used to monitor body temperature.

SpO2 Sensor: Pulse oximeters using LED light sensors to detect blood oxygen saturation.

Blood Pressure Sensor: Wearable oscillometric sensors or inflatable cuff-based sensors.

Respiratory Rate Sensor: Sensors that detect breathing patterns via airflow or thoracic movement.

These sensors are integrated into wearable devices such as smartwatches, fitness bands, or specialized medical-grade wearables. The sensors must be calibrated to ensure accurate readings and minimize errors due to motion artifacts or environmental factors.

3. Data Collection and Preprocessing

Once the wearable sensors are integrated and deployed on the user, data collection begins. The data is continuously captured by the sensors at specified intervals (e.g., every 5 seconds, every minute, etc.). For the collected data to be useful, the following preprocessing steps are typically performed:

Normalisation: To account for differences in sensor calibration, the data is normalised to bring all readings into a consistent range.

Outlier Detection: Outliers or erroneous data points are identified and discarded or corrected. For example, an unusually high heart rate could be flagged as an anomaly.

4. Data Transmission

The processed data is transmitted from the wearable sensors to a mobile application or cloud-based platform for further analysis and storage. The transmission is typically done using wireless communication protocols such as:

Bluetooth Low Energy (BLE): Common for short-range data transmission, especially for consumer-grade wearables.

Wi-Fi: Used when the wearable is within range of a Wi-Fi network for higher bandwidth transmission.

Cellular Networks (3G/4G/5G): For long-range data transmission, especially for mobile users who are on the move.

Data is sent in real-time or in batches depending on the system's design and network constraints. Data transmission should be encrypted and secure to protect user privacy and comply with regulations such as HIPAA (Health Insurance Portability and Accountability Act).

5. Data Storage and Cloud Integration

Once the health data is transmitted, it is stored either on a cloud server or a centralized database for analysis. Cloud integration enables scalability and remote access, allowing healthcare providers to monitor patient data remotely.\

The following key aspects are involved in this phase:

Data Security: Ensure that data storage complies with security standards such as end-to-end encryption, multi-factor authentication, and data anonymization.

Cloud Computing: Cloud platforms like AWS, Microsoft Azure, or Google Cloud are used to store and process the data. These platforms can handle large volumes of data from multiple users and support scalability as the system expands.

Data Synchronisation: The system must ensure that data collected from various sensors is synchronised to allow for accurate trend analysis and comparisons.

6. Data Analysis and Machine Learning

After data storage, the analysis process begins. The primary goal is to provide insights from the continuous health data collected by the sensors. Data analysis typically involves the following steps:

Trend Analysis: Health metrics are analyzed over time to detect trends. For example, a user's resting heart rate might show an increasing trend, which could indicate potential health concerns such as stress or cardiovascular issues.

Anomaly Detection: Using predefined thresholds or machine learning algorithms, the system detects outliers or unusual patterns in the data that could indicate a health risk (e.g., an unusually high heart rate, abnormal blood pressure levels, or low oxygen saturation).

Classification: Health data can be classified into different categories, such as "Normal," "At Risk," or "Critical," based on the severity of the data.

Machine learning models are trained on historical datasets to identify patterns and improve the system's ability to make accurate predictions. For example, a model could be trained to predict abnormal heart rate variability based on the user's age, activity level, and previous health data.

7. Alert and Notification System

The system generates alerts when anomalies or critical health conditions are detected. The alerting system should include:

Threshold-based Alerts: For instance, if a user's heart rate exceeds a specified threshold (e.g., 120 beats per minute), the system sends an alert.

Trend-based Alerts: Alerts based on changes or trends in health data over time, such as a gradual increase in blood pressure or a sustained drop in oxygen levels.

Urgency Levels: Alerts are categorized based on their urgency, ranging from "Routine Monitoring" to "Immediate Medical Attention Required."

Alerts are sent to the user via a mobile application, and, if necessary, to healthcare providers or emergency contacts through push notifications, SMS, or email.

8. Mobile Application Interface

The mobile application serves as the user interface for the AHMS, allowing individuals to:

View Real-time Data: The app displays health metrics in graphical or tabular format, providing users with easy-to-understand visualizations of their health data.

Track Health Trends: Users can track trends in their health data over time, such as changes in heart rate, blood pressure, or temperature.

Receive Alerts and Notifications: The app alerts users if there is any abnormality detected in their health data. It may also provide recommendations or warnings based on the detected anomaly.

The mobile app also provides users with personalized health recommendations and goals, based on their data and trends.

9. Evaluation and Testing

Before deployment, the system undergoes rigorous testing to ensure its functionality, accuracy, and reliability:

System Testing: Testing of sensor data accuracy, data transmission reliability, and app functionality.

10. Deployment and Maintenance

Once the system has been thoroughly tested and validated, it can be deployed for real-time use. Ongoing maintenance involves monitoring the system's performance, updating the mobile app and cloud platform, and ensuring that the wearable devices continue to function accurately over time. User feedback is gathered to refine the system and improve the user experience.

Fig 1:- This figure says about how the data get from the body.

RESULTS

Health Metrics Overview Dashboard

Description: A comprehensive dashboard showing all real-time health metrics (heart rate, blood pressure, body temperature, SpO2, etc.) for the user.

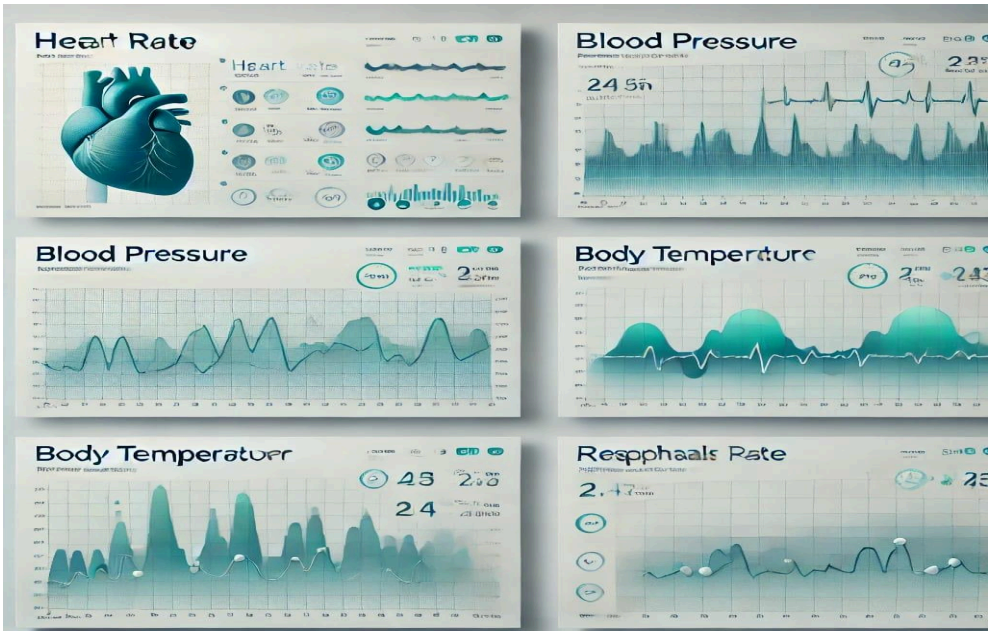

Fig 1.1:- The above figure represents the Heart rate, Blood pressure, Body temperature and it give exact result and sends the data to our mobile.

Fig1.2:- The image shows a blood pressure reading from a smartphone app. The systolic blood pressure is 120 mmHg and the diastolic blood pressure is 80 mmHg.

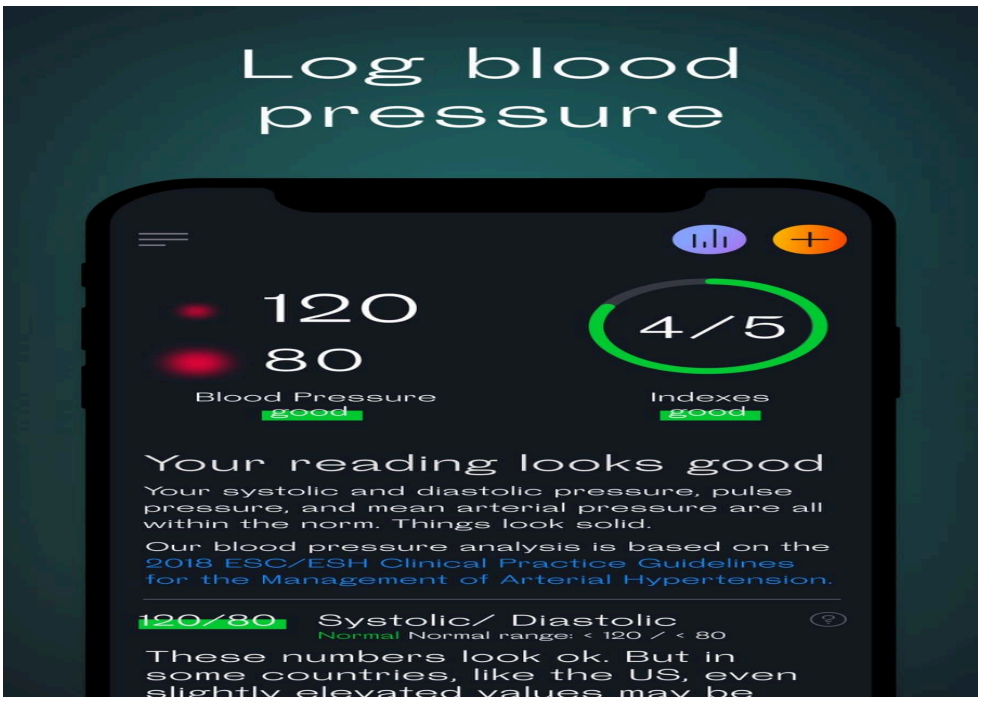

Fig 1.3:- The image shows a smartwatch on a person's wrist displaying real-time heart rate information. The smartwatch screen reads "Heart Rate" and shows a current reading of 75 BPM (beats per minute), with a recent reading of 70 BPM taken two minutes pri

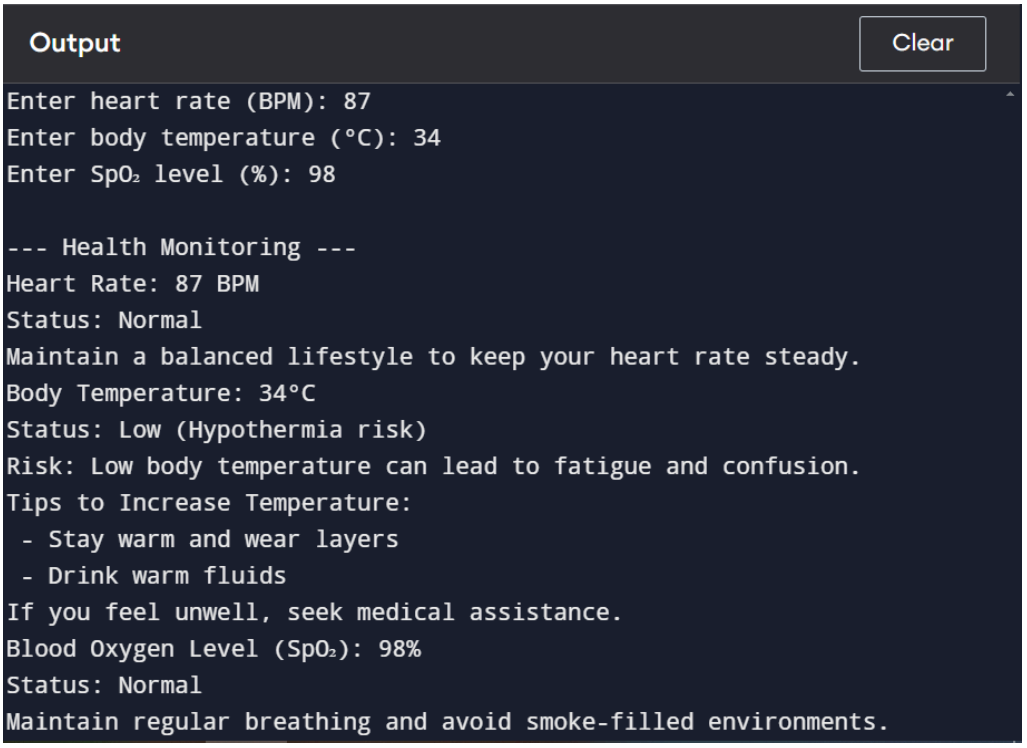

Fig 1.4: shows the output of the health analysis

CONCLUSION

Smart Home Automation presents a transformative approach to managing household environments, combining convenience, energy efficiency, and enhanced security. By automating routine tasks and providing remote control over appliances, lighting, and security systems, it simplifies daily life and supports sustainable energy use. Despite challenges like initial setup costs and privacy concerns, the benefits of Smart Home Automation make it an appealing solution for modern living. As technology continues to advance, the potential for smarter, more adaptive homes will only increase, shaping the future of comfortable and efficient home management.

The Automated Health Monitoring System presents a significant advancement in personalized healthcare, offering continuous, real-time tracking of vital signs, and enhancing the ability of caregivers and medical staff to respond promptly to patients' needs. By leveraging IoT technology and a robust software framework in C++, this system not only improves monitoring accuracy but also provides timely alerts when vital parameters deviate from safe ranges, ensuring rapid response to potential health issues. With a user-friendly interface and data visualization, users can easily interpret health metrics and manage alerts, making the system accessible to both medical professionals and caregivers. Furthermore, the automated health monitoring solution is designed with data privacy and security in mind, adhering to healthcare standards to protect sensitive information.

REFERENCES

- Patel, S., Park, H., Bonato, P., Chan, L., & Rodgers, M. (2012). *A review of wearable sensors and systems with application in rehabilitation*. Journal of NeuroEngineering and Rehabilitation, 9(1), 1-17.

- Islam, S. M. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). *The Internet of Things for health care: A comprehensive survey*. IEEE Access, 3, 678-708.

- Li, X., Zhang, X., & Wang, H. (2021). *Wearable IoT enabled real-time health monitoring system*. Journal of Electrical and Computer Engineering, 2021, 1-9.

- Raghavendra, A. R., & Sahoo, G. (2017). *IoT-based patient health monitoring system: A comprehensive review*. Journal of Healthcare Engineering, 2017, 1-12.

- Gupta, P., Gupta, S., & Gupta, A. (2016). *IoT-based health monitoring and wearable devices*. In Proceedings of the IEEE International Conference on Computing, Communication and Automation (ICCCA), 1325-1330.

APPENDIX

C++ CODE:

```
#include <iostream>

#include <string>

#include <chrono>

#include <thread>


// Function to analyze heart rate and provide feedback

void analyzeHeartRate(int heartRate) {

    std::cout << "Heart Rate: " << heartRate << " BPM" << std::endl;


    if (heartRate >= 60 && heartRate <= 100) {

            std::cout << "Status: Normal\nMaintain a balanced lifestyle to keep your heart rate steady." << std::endl;

    } else if (heartRate > 100) {

          std::cout << "Status: High\nRisk: Elevated heart rate can indicate stress, dehydration, or other issues.\n";

            std::cout << "Tips to Reduce Heart Rate:\n - Take deep breaths and relax\n - Drink water\n - Avoid caffeine\n";

        std::cout << "If the rate remains high, consult a doctor." << std::endl;

    } else if (heartRate < 60) {

        std::cout << "Status: Low\nRisk: Low heart rate can cause dizziness or weakness.\n";

          std::cout << "Tips to Raise Heart Rate:\n - Move around or do light exercise\n - Avoid prolonged sitting\n";
```

```
        std::cout << "If symptoms persist, consult a healthcare provider." << std::endl;
    } else {
        std::cout << "Error: Invalid heart rate data." << std::endl;
    }
}

// Function to analyze body temperature and provide feedback
void analyzeBodyTemperature(float temperature) {
    std::cout << "Body Temperature: " << temperature << "°C" << std::endl;

    if (temperature >= 36.1 && temperature <= 37.2) {
        std::cout << "Status: Normal\nMaintain hydration and avoid extreme temperatures." << std::endl;
    } else if (temperature > 37.2) {
        std::cout << "Status: High (Fever)\nRisk: Fever can indicate infection or illness.\n";
        std::cout << "Tips to Lower Fever:\n - Rest and stay hydrated\n - Use a cool compress\n";
        std::cout << "If fever persists or exceeds 39°C, consult a doctor." << std::endl;
    } else if (temperature < 36.1) {
        std::cout << "Status: Low (Hypothermia risk)\nRisk: Low body temperature can lead to fatigue and confusion.\n";
        std::cout << "Tips to Increase Temperature:\n - Stay warm and wear layers\n - Drink warm fluids\n";
        std::cout << "If you feel unwell, seek medical assistance." << std::endl;
    } else {
        std::cout << "Error: Invalid temperature data." << std::endl;
    }
```

```
}

// Function to analyze SpO₂ level and provide feedback
void analyzeBloodOxygenLevel(int spo2) {
    std::cout << "Blood Oxygen Level (SpO₂): " << spo2 << "%" << std::endl;

    if (spo2 >= 95 && spo2 <= 100) {
            std::cout << "Status: Normal\nMaintain regular breathing and avoid smoke-filled environments." << std::endl;
    } else if (spo2 < 95 && spo2 >= 90) {
        std::cout << "Status: Mild Hypoxia\nRisk: Low oxygen can cause fatigue and shortness of breath.\n";
        std::cout << "Tips to Increase Oxygen Level:\n - Practice deep breathing\n - Avoid physical exertion\n";
        std::cout << "If levels drop further, seek medical help." << std::endl;
    } else if (spo2 < 90) {
        std::cout << "Status: Severe Hypoxia\nRisk: Very low oxygen levels are dangerous.\n";
        std::cout << "Actions Needed:\n - Seek immediate medical assistance\n";
        std::cout << "If oxygen level is consistently low, consult a healthcare provider." << std::endl;
    } else {
        std::cout << "Error: Invalid SpO₂ data." << std::endl;
    }
}

// Function to receive user input for health data
void receiveHealthData(int &heartRate, float &temperature, int &spo2) {
```

```
    std::cout << "Enter heart rate (BPM): ";
    std::cin >> heartRate;
    std::cout << "Enter body temperature (°C): ";
    std::cin >> temperature;
    std::cout << "Enter SpO₂ level (%): ";
    std::cin >> spo2;
}

int main() {
    while (true) {
        int heartRate;
        float temperature;
        int spo2;

        // Get health data from user
        receiveHealthData(heartRate, temperature, spo2);

        // Analyze each health parameter
        std::cout << "\n--- Health Monitoring ---" << std::endl;
        analyzeHeartRate(heartRate);
        analyzeBodyTemperature(temperature);
        analyzeBloodOxygenLevel(spo2);
        std::cout << "--- End of Analysis ---\n" << std::endl;

        // Option to continue or stop
```

```
        char continueMonitoring;
        std::cout << "Do you want to input new data? (y/n): ";
        std::cin >> continueMonitoring;
        if (continueMonitoring == 'n' || continueMonitoring == 'N') {
            break;
        }
    }
    return 0;
}
```