**HACKVERESE 1.0**

# TEAM NAME: DIGITAL DEVELOPERS

HEALTH CARE

“CARE CONNECT – Automated Patient Alert System”

**PROBLEM STATEMENT:**

Here we create a real-time patient monitoring system that detects vital sign abnormalities and instantly alerts medical staff via calls or messages while securely storing and updating patient health data for continuous monitoring and analysis

We have got this idea while visiting the hospital randomly and we have discussed to create a app that can give alerts to respected/concerned doctors or nurses so they can immediately respond with in no time and can reduce the critical or serious emergencies of the particular patient.

We have followed a step-by-step process

**STEP 1:**

**->Objective:**

We have created a system to monitor vital signs like pulse rate, blood pressure, blood sugar etc.., post-operation and display them on mobile app.

We have to use both Hardware and Software in the prototype.

**STEP 2:** Selecting Hardware and Software Components

**->Hardware:**

Hardware components we use are different type of sensors to measure the vital sign changes the patient body.

Sensors:

Pulse Rate: Use a pulse oximeter sensor (e.g., MAX30102) to measure heart rate and SpO2 (oxygen saturation).

Blood Pressure: Non-invasive BP sensors (e.g., Omron or similar wearable BP modules). Alternatively, use a cuff-based sensor for periodic measurements.

Blood Sugar: Non-invasive glucose monitoring is challenging; consider integrating with existing devices like continuous glucose monitors (CGM) such as Dexcom or Freestyle Libre via APIs or Bluetooth.

Additional Sensors: Temperature sensor (e.g., DS18B20) or ECG sensor for heart activity (e.g., AD8232).

As there many sensors we have used a basic sensor to maintain a fast response.

**->Software:**

Software component we use is a mobile app to display real-time data, store historical data and alert users if vitals exceed safe thresholds

The main software tools use for Frontend are:

Html: To create a web page appearance.

CSS: To make the page more attractive,

TAILWIND: A popular utility-firs CSS framework called Tailwind CSS.

TSX: It is file extension in web development, that indicates a TypeScript file that contains JSX syntax (JSX-it is a java script library)

TS: It adds a static typing to java script , allowing developers to catch errors during development instead of at runtime.

**->Connectivity:**

Wireless data transmission from sensors to the mobile app through wi-fi or Bluetooth. And it gives a notification for abnormal readings.

**->Microcontroller:**

Use a microcontroller like Arduino, Raspberry Pi, or ESP32 to process sensor data and handle connectivity.

**STEP 3:** Design the Hardware System

**->Circuit Design:**

We designed the circuit by connecting sensors to the microcontroller’s analog /digital pins. Ensure power management voltage regulators for stable sensor

We should use a breadboard or PCB to assemble the sensors, microcontroller, and connectivity module.

**STEP 4:** Develop the Software Backend

**->Data Processing:**

Data Collection:

We used a Program for the microcontroller to read sensor data at regular intervals.

Implement algorithms to filter noise and ensure accurate measurements (e.g., averaging pulse readings over 10 seconds).

Noise Filtering :

Apply algorithms to ensure accuracy by removing noise from sensor readings like moving average, median filter, thresholding, frequency.

Ex : for pulse average 10 readings over 10 seconds to reduce noise from movement.

Data Formatting :

We formatted the data into a usable format by using the above mentioned software applications for transmission to the mobile app via Bluetooth of wi-fi.

Data Transmission :

After formatting the data we have to send the processed data rto the mobile app for real-time monitoring and storage.

**Step 5 :** Develop the Mobile Application

We developed the mobile application by using some platforms and features.

Platforms :

We used flutter, React Native , or native development for cross-platform compatibility.

Features:

The features we used are Dashboard, Alerts, History, UserInterface.

**STEP 6:** Integrated Hardware and Software for patient Monitoring System

Integration involves connecting the hardware (sensors and microcontroller) with the software (mobile app) to enable seamless data flow from patient vital signs monitoring to real-time display and alerts.

Here’s a brief explanation:

Pairing Devices:

Establish a connection between the microcontroller (e.g., Arduino, ESP32) and the mobile app using Bluetooth (e.g., HC-05 module) or Wi-Fi for wireless data transfer.

In the app, implement a pairing mechanism to discover and connect to the hardware (e.g., using Bluetooth APIs in Flutter or React Native).

Data Transfer:

The microcontroller processes sensor data (e.g., pulse, BP, temperature) and sends it to the app in a structured format (e.g., JSON: {"pulse": 75, "temp": 36.5}).

Use protocols like Serial communication (Bluetooth) or MQTT/HTTP (Wi-Fi) for reliable transmission.

Real-Time Sync:

The mobile app receives and parses incoming data to display real-time vital signs (e.g., on a dashboard with graphs).

Implement error handling to manage connection drops (e.g., buffer data locally on the microcontroller if the app is offline).

Testing Integration:

Verify that data flows correctly from sensors to the app (e.g., pulse readings update every 5 seconds).

Test for latency and ensure the app reflects accurate, filtered data.

**Step 7:** Testing and Validation

Unit Testing:

Test each sensor individually for accuracy.

Integration Testing:

Verify that data flows correctly from sensors to the app.

User Testing:

Conduct trials with mock patients to ensure usability and reliability.

Medical Validation:

Compare readings with medical-grade devices to ensure accuracy (consult with healthcare professionals if possible).

**Step 8:** Ensure Regulatory Compliance

Data Privacy:

Implement encryption and comply with regulations like HIPAA (US) or GDPR (EU).

Medical Certification:

If the system is intended for commercial use, seek FDA (US) or CE (EU) approval for medical devices. Certificates are mandatory as per the appointed patient.

User Safety:

Ensure sensors are non-invasive and safe for prolonged use.

**Step 9:** Deployment and Iteration

Pilot Testing:

Deploy the system in a controlled environment (e.g., a hospital ward) for feedback.

Iterate:

Refine the system based on user feedback, improving accuracy, battery life, or app features.

Documentation:

Provide user manuals for patients and healthcare providers.

Revenue:

we can generate revenue by selling our website , app to the big MNC companies or to the private ,govt hospitals. We expect this app may give more service and help to the doctors as well as the patients.

We can also gain a good income through this idea as far of now this is not existed .

As this app is not existed we may get a good revenue income and may helpful to lot of hospitals.

Presentation(PPT):

We build a PPT briefly regarding our problem statement .

Our ppt consists:

->Introduction, key features, objectives,

->Hardware and Software components or the tools we used to build our app, website.

->Designing of the software using html, css, js, tsx, ts, tailwind, etc related directories and files.

->Designing of the hardware by sensors and detecters.

->Hardware, Software integration.

-> Impact and novelty.

->Future steps and demo.

Prototype:

We designed a prototype regarding the code we implemented and the libraries we used. We build a short note prototype not deeply explaining about the each and every programming code.