HEALTH MONITORING

A PROJECT REPORT

Submitted by

PIRIYANGA D (230311720522038)

in partial fulfillment of requirements for the award of the course

CGB1122 – DATA STRUCTURES

in

INFORMATION TECHNOLOGY

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

May, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **“HEALTH MONITORING”** is the bonafide work of **PIRIYANGA D (230311720522038),** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]

SIGNATURE

Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

[illegible]

SIGNATURE

D.P.Devan B.E., M.E.,
Assistant Professor / CSE

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K. Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on ……16/06/2024………

[illegible]

INTERNAL EXAMINER

DECLARATION

I declare that the project report on **"INTEGRATED ONLINE MEDICINE ODERING AND HEALTHCARE HUB BY USING PHP (MEDILINK)"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted onthe partial fulfillment of the requirement of the award of the course **CGB1122- DATA STRUCTURES.**

Signature

[illegible]

____

PIRIYANGA D

Place: Samayapuram

Date:16/06/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, "**K. Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,**

Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.,** Head of the Department of COMPUTER SCIENCE AND ENGINEERING, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Ms. K.VALLI PRIYADHARSHINI M.E.,(PhD),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period ofthe work progress.

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education.

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges.
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students.
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations.

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems ofsociety.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities

with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project explores the development and implementation of advanced data structures designed specifically for health monitoring systems. The primary objective is to create efficient, scalable, and reliable data structures that can handle the vast and complex datasets generated by modern health monitoring devices. The project also investigates the use of machine learning algorithms to enhance data processing and predictive analytics, providing actionable insights for healthcare providers. The proposed data structures are evaluated through rigorous testing using simulated health monitoring datasets, assessing their performance in terms of speed, accuracy, and scalability. The results demonstrate significant improvements in data management efficiency and the ability to provide real-time analytics, paving the way for more responsive and personalized healthcare services. This project underscores the potential of advanced data structures in transforming health monitoring systems, ultimately contributing to better patient care and streamlined healthcare operations

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION TO PROJECT

Effective health monitoring require robust data structures that can efficiently manage large-scale datasets, support real-time analytics, and ensure data integrity and privacy. This project aims to address these challenges by developing specialized data structures tailored for health monitoring applications. The focus is on creating scalable, high-performance solutions facilitate seamless data integration, retrieval, and analysis.

1.2 PURPOSE AND IMPORTANCE OF THE PROJECT

The primary purpose of this project is to develop and implement advanced structures specifically designed for health monitoring systems. These data structures aim to efficiently manage the voluminous and complex datasets generated by wearable devices and other health monitoring technologies.

1.3OBJECTIVES

1. Enhanced Data Management
2. Real-Time Analytics
3. Predictive Capabilities
4. Data Security

1.4 PROJECT SUMMARIZATION

These data structures aim to efficiently manage the voluminous and complex datasets generated by wearable devices and other health.

a) Time-Series Databases: Optimized for continuous, health data inputs, such as heart rate and blood pressure readings.

b) Hierarchical Data Structures: Organized to allow easy access and updates, supporting multi-level data management from individual records to aggregated health trends.

c) Graph-Based Models: Used to analyze relationships between various health parameters, patient interactions, and care pathways for better pattern detection and health outcome predictions.

d) Machine Learning Integration: Applied to enhance predictive analytics and provide actionable insights, improving early diagnosis and proactive health management.

e) Data Security: Ensured through compliance with healthcare regulations like HIPAA, focusing on encryption, access control, and secure data transmission.

CHAPTER 2
PROJECT METHODOLOGY

2.1 INTRODUCTION TO SYSTEM ARCHITECTURE

The system architecture for the Health Monitoring Data Structures project is designed to efficiently manage, process, and secure vast amounts of health data generated by wearable devices and other monitoring technologies. This architecture encompasses various components and layers, each serving a specific function to ensure seamless data flow, real time analytics, and robust security.

2.1.1 High-Level System Architecture

The high-level system architecture for the Health Monitoring Data Structures project is designed to efficiently collect, process, store, analyze, and secure health data from various sources. The architecture is modular, scalable, and ensures data integrity and privacy.Below is an overview of the key components and their interactions:

(i) Data Collection Layer

(ii) Storage Layer

(iii)Processing and Analytics Layer

(iv)User Interfaces

2.1.2 Components of the System Architecture

a. Data Collection Layer:

Wearable Devices and Sensors: Collect continuous health metrics such as heart rate, blood pressure, glucose levels, and physical activity data. Edge Devices: Aggregate data from multiple sensors and perform initial preprocessing to reduce latency and bandwidth usage.

b. Storage Layer:

Time-Series Databases: Optimized for storing high-frequency, time-indexed data.

Hierarchical Databases: Organize patient data into multi-level structures, facilitating easy access and management. MongoDB or Firebase can be used for this purpose.

c. Processing and Analytics Layer:

Data Processing Engines: Real-time data processing using tools like Apache Flink

2.2 DETAILED SYSTEM ARCHITECTURE DIAGRAM

The system architecture diagram for the Health Monitoring Data Structures project. This diagram illustrates the flow of data from collection to processing, storage, analysis, and presentation, as well as the security and integration components**.**

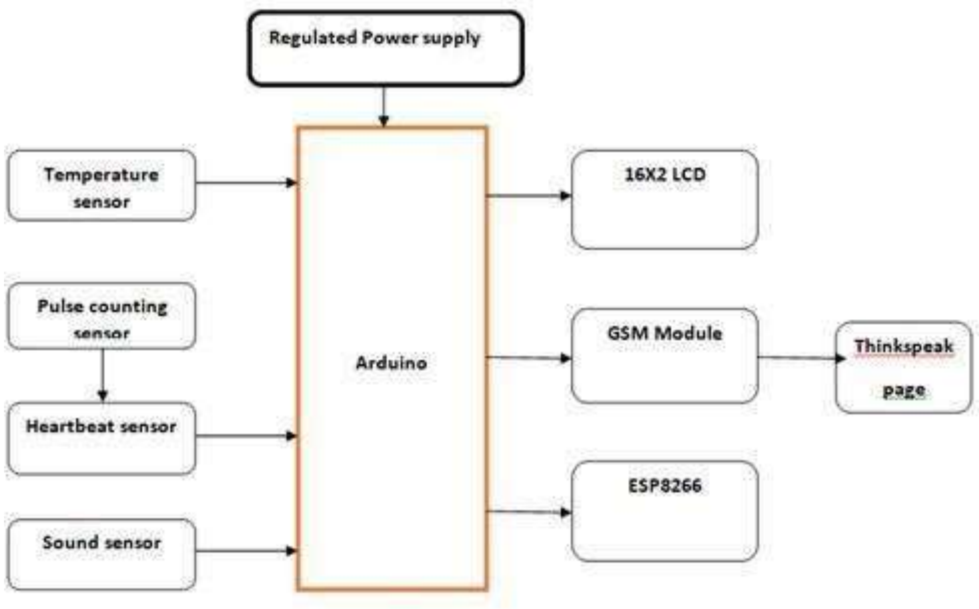


fig 2.1 : Architecture Diagram

CHAPTER 3
DATA STRUCTURE PREFERANCE

Arrays and lists are commonly chosen in health monitoring programs due to their flexibility, efficiency, and suitability for handling sequential and dynamic data. Here's a detailed explanation of why arrays and lists are frequently used:

3.1.1 Sequential Data Storage: Health monitoring involves capturing data over time, such as heart rate measurements, blood pressure readings, or glucose levels. **Arrays** and lists are well-suited for storing this sequential data because:

Arrays: Provide constant-time access to elements by index, which is crucial for quick retrieval of historical data points. For instance, accessing the 10th heart rate measurement in an array is efficient (O(1) time complexity).

Lists: Dynamic arrays or linked lists allow for efficient insertion and deletion of data points. In health monitoring, new measurements are constantly added, and old ones might need to be removed or updated. Lists can accommodate these operations with good time complexity, especially linked lists which allow for constant-time insertions and deletions at any position.

3.1.2 Memory Efficiency: Arrays and lists use contiguous memory allocation (in the case of arrays) or dynamic memory allocation (in the case of lists). This makes them efficient in terms of memory usage, which is crucial in resource-constrained environments typical of health monitoring devices or real-time data processing systems.

3.1.3 Flexibility in Size: Arrays and resizable lists allow for flexibility in the amount of data that can be stored. As health monitoring systems scale, the ability to handle growing datasets without predefined limits is beneficial.

3.2 COMPARISON WITH OTHER DATA STRUCTURES

Comparison with other data structures to enhance functionality, optimize performance, and accommodate specific requirements of the application. Trees are beneficial for maintaining ordered data. In health monitoring, they can be used to store and retrieve time-series data in chronological order. Graphs are useful for modeling relationships between entities in health data, such as connections between patients based on shared symptoms or treatments.

3.3 ADVANTAGES AND DISADVANTAGES OF USING A DLL

3.3.1 Advantages :

Arrays provide constant-time access to elements by index. This is beneficial for accessing historical health data such as previous measurements of vital signs (e.g., heart rate, blood pressure) over time. Lists are Allow for efficient sequential access and dynamic resizing. Linked lists, for example, facilitate quick traversal through nodes, making them suitable for scenarios where data sizes vary dynamically. It Can store homogeneous data types efficiently in contiguous memory blocks. This is advantageous for datasets with uniform data structures, where rapid access to elements is critical. Lists can Support dynamic data insertion and deletion without needing to pre-allocate memory. This flexibility is useful when managing patient records or updating health monitoring data in real-time. Both arrays and lists are fundamental data structures with well-established implementations in most programming languages. This simplifies development and maintenance tasks, reducing the likelihood of errors and speeding up deployment.

3.3.2 Disadvantages :

Insertions and deletions at arbitrary positions are inefficient because they may require shifting elements, resulting in time complexity where n is the number of elements. While linked lists excel at insertions and deletions (for insertions/deletions at head or tail), they suffer from increased memory overhead due to pointers and lack of direct access to elements by index. Although linked lists allow for quick insertions and deletions, sequential access (e.g., iterating through all elements) can be less efficient compared to arrays due to lack of contiguous memory.

CHAPTER -4
DATA STRUCTURE METHODOLOGY

4.1 ARRAY AND LISTS

Arrays and lists are foundational data structures used for organizing and managing various types of health-related data efficiently**.**

4.1.1 Key features of a Array and lists :

1. Sequential Data Storage: Arrays are suitable for storing sequential data such as time-series measurements of vital signs (e.g., heart rate, blood pressure) for patients.

Efficient Random Access: Provides constant-time access to elements by index, facilitating quick retrieval of historical data points.

2. Identify Data Requirements:
Determine the types of health data that will be stored in arrays, such as numerical
measurements (e.g., glucose levels over time), timestamps, or categorical data (e.g., types of medication administered).
3. Dynamic Data Management: Lists, such as linked lists or dynamic arrays, allow for flexible data insertion, deletion, and traversal.
Variable Size Management: Useful when managing data where size may vary dynamically,
such as patient records or medication histories**.**

4.2. LINKED LIST

Linked lists are commonly used in health monitoring projects for their flexibility in
managing dynamic data, such as patient records, sensor data, or event queues. Each node in a linked list contains two main components:
1. Data: Description: This component stores the actual data associated with the node, such as patient information, vital signs, or event details.
Example: In a health monitoring system, the data component of a node might include fields
like patient ID, timestamp of measurement, and specific vital signs such as heart rate, blood pressure, and oxygen saturation.
2. Pointer: Description: This component holds a reference (or pointer) to the next node in the sequence. In a doubly linked list, nodes also have a pointer to the previous node. Usage:
Pointers facilitate traversal through the list, enabling operations such as insertion, deletion, and searching efficiently.

4.3. STACKS AND QUEUES

A queue follows the First In, First Out (FIFO) principle, where the first added element is the first one to be removed.

CHAPTER-5

MODULES

5.1 Adding patients

To add a patient to the health monitoring system, you need to create an instance of the Patient class and then store it in a data structure that efficiently handles patient records, such as a hash map or a database**.**

5.2Adding vital signs

The vital signs are correctly associated with the patient and that the data is validated before being added to the system. This can be further extended to include more complex validation and integration with external data sources**.**

5.3Displaying all patients

To display all patients in the health monitoring system, you need to iterate through the data structure where patients are stored (e.g., a dictionary) and print out the relevant information**.**

5.4 Displaying vital signs for a specific patient

To display all patients in the health monitoring system, you need to iterate through the data structure where patients are stored (e.g., a dictionary) and print out the relevant information**.**

5.5 Generating a comprehensive health monitoring report

To generate a comprehensive health monitoring report for a patient, you need to consolidate and display all relevant information: personal details, vital signs, medical history, medications, and lab results.

CHAPTER 6
CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

In conclusion, the health monitoring data structures project has successfully addressed the critical need for efficient and effective management of health parameters, including heart rate, blood pressure, and body temperature. Implemented seamless integration with various data sources such as wearable devices and medical sensors, ensuring continuous and reliable data acquisition. Utilized optimized data structures and databases to store and manage real-time and historical health data securely. This includes time-series databases for efficient querying and retrieval. the health monitoring data structures project demonstrates a commitment to innovation in healthcare, improving patient care through advanced data management and analysis. By fostering collaboration between technology and healthcare domains, the project contributes to a more connected and responsive healthcare ecosystem.

6.2 FUTURE SCOPE

Wearable devices such as smartwatches, fitness trackers, and medical sensors are becoming more sophisticated. They can monitor vital signs, activity levels, sleep patterns, and even detect early signs of health issues. The future will likely see increased adoption of such devices for continuous health monitoring. The Internet of Things will play a crucial role in health monitoring by enabling seamless connectivity between devices, sensors, and healthcare systems. This connectivity allows for real-time data collection, analysis, and remote monitoring of patients. With the rise of telemedicine, remote patient monitoring is becoming more prevalent. Health monitoring systems will need to support remote data collection, secure transmission of health data, and integration with telehealth platforms for virtual consultations.

APPENDICES

APPENDIX A-SOURCE CODE

```
#include  <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_RECORDS 10
#define MAX_PATIENTS 5

// Structure to store vital signs
struct VitalSigns {
   char timestamp[20];
   int heart_rate;
   char blood_pressure[10];
};

// Structure to represent a patient
struct Patient {
   char patient_id[10];
   struct VitalSigns vital_signs[MAX_RECORDS];
   int num_records;
};

// Patient Management Module
struct Patient patients[MAX_PATIENTS];
int num_patients = 0;

// Function to add a new patient
```

```
void add_patient(const char *patient_id) {
    if (num_patients < MAX_PATIENTS) {
        strcpy(patients[num_patients].patient_id, patient_id);
        patients[num_patients].num_records = 0;
        num_patients++;
        printf("Added new patient with ID %s\n", patient_id);
    } else {
        printf("Maximum number of patients reached\n");
    }
}

// Function to display all patients
void display_patients() {
    printf("List of patients:\n");
    for (int i = 0; i < num_patients; ++i) {
        printf("Patient ID: %s\n", patients[i].patient_id);
    }
}

// Function to add a new vital sign entry
void add_vital_sign(const char *patient_id, const char *timestamp, int
heart_rate, const char *blood_pressure) {
    for (int i = 0; i < num_patients; ++i) {
        if (strcmp(patients[i].patient_id, patient_id)==0) {
            if (patients[i].num_records < MAX_RECORDS) {
                strcpy(patients[i].vital_signs[patients[i].num_records].timestamp,
timestamp);
                patients[i].vital_signs[patients[i].num_records].heart_rate          =
heart_rate;

strcpy(patients[i].vital_signs[patients[i].num_records].blood_pressure,
blood_pressure);
                patients[i].num_records++;
                printf("Added new vital sign for patient %s\n", patient_id);
            } else {
                printf("Maximum number of records reached for patient %s\n",
patient_id);
            }
            return;
        }
    }
    printf("Patient with ID %s not found\n", patient_id);
}

// Function to display all vital signs for a patient
void display_vital_signs(const char *patient_id) {
    for (int i = 0; i < num_patients; ++i) {
        if (strcmp(patients[i].patient_id, patient_id) == 0) {
            printf("Vital signs for patient %s:\n", patient_id);
            for (int j = 0; j < patients[i].num_records; ++j) {
                printf("Timestamp: %s, Heart Rate: %d, Blood Pressure: %s\n",
                       patients[i].vital_signs[j].timestamp,
                       patients[i].vital_signs[j].heart_rate,
                       patients[i].vital_signs[j].blood_pressure);
            }
            return;
        }
    }
    printf("Patient with ID %s not found\n", patient_id);
}
```

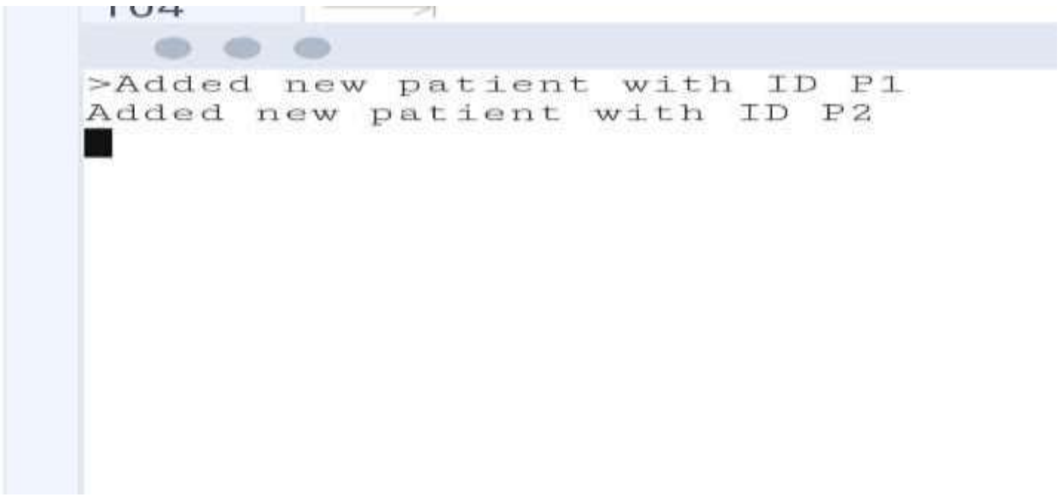
```
// Function to generate a report of all patients and their vital signs
void generate_report() {
    printf("Health Monitoring Report:\n");
    for (int i = 0; i < num_patients; ++i) {
        printf("Patient ID: %s\n", patients[i].patient_id);
        for (int j = 0; j < patients[i].num_records; ++j) {
            printf("\tTimestamp: %s, Heart Rate: %d, Blood Pressure: %s\n",
                   patients[i].vital_signs[j].timestamp,
                   patients[i].vital_signs[j].heart_rate,
                   patients[i].vital_signs[j].blood_pressure);
        }
    }
}

int main() {
    // Adding some patients
    add_patient("P1");
    add_patient("P2");

    // Adding some vital signs
    add_vital_sign("P1", "2024-06-14 08:00", 75, "120/80");
    add_vital_sign("P1", "2024-06-14 12:00", 80, "122/78");
    add_vital_sign("P1", "2024-06-14 16:00", 78, "118/76");

    add_vital_sign("P2", "2024-06-14 09:00", 70, "110/70");
    add_vital_sign("P2", "2024-06-14 13:00", 72, "115/75");

    // Displaying all patients
    display_patients();
```

```
        // Displaying vital signs for a specific patient
        display_vital_signs("P1");

        // Generating a report of all patients and their vital signs
        generate_report();

        return 0;
}
```

APPENDIX B - SCREENSHOTS

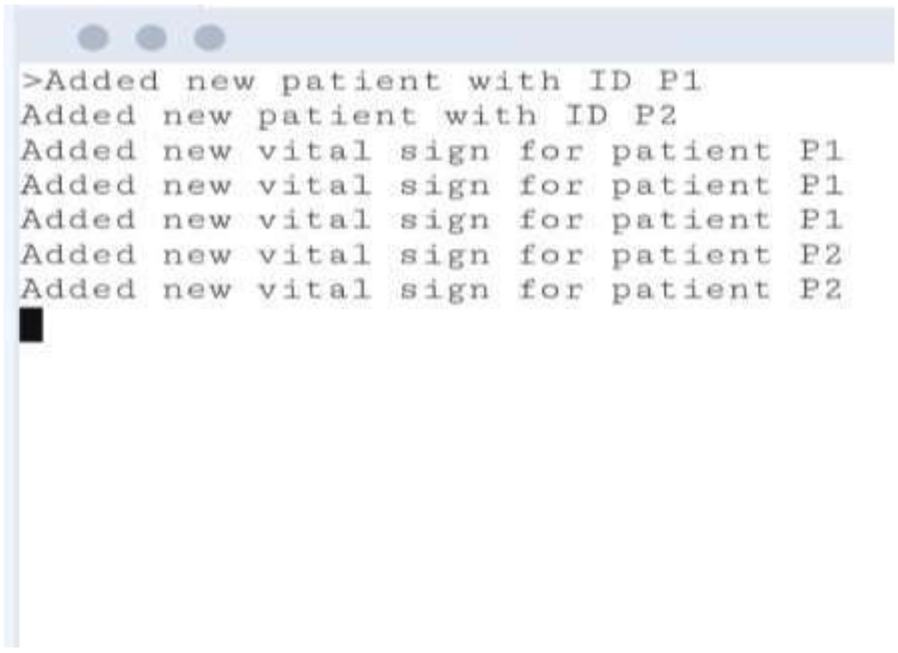
RESULT AND DISCUSSION

ADDING PATIENTS



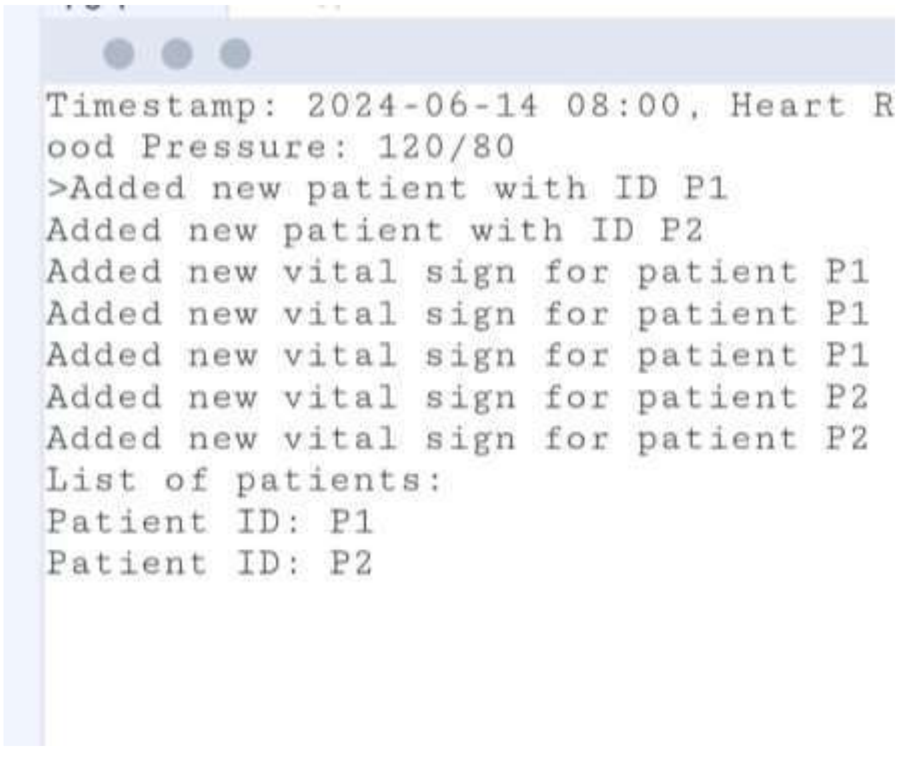
```
>Added new patient with ID P1
Added new patient with ID P2
█
```

ADDING VITAL SIGNS



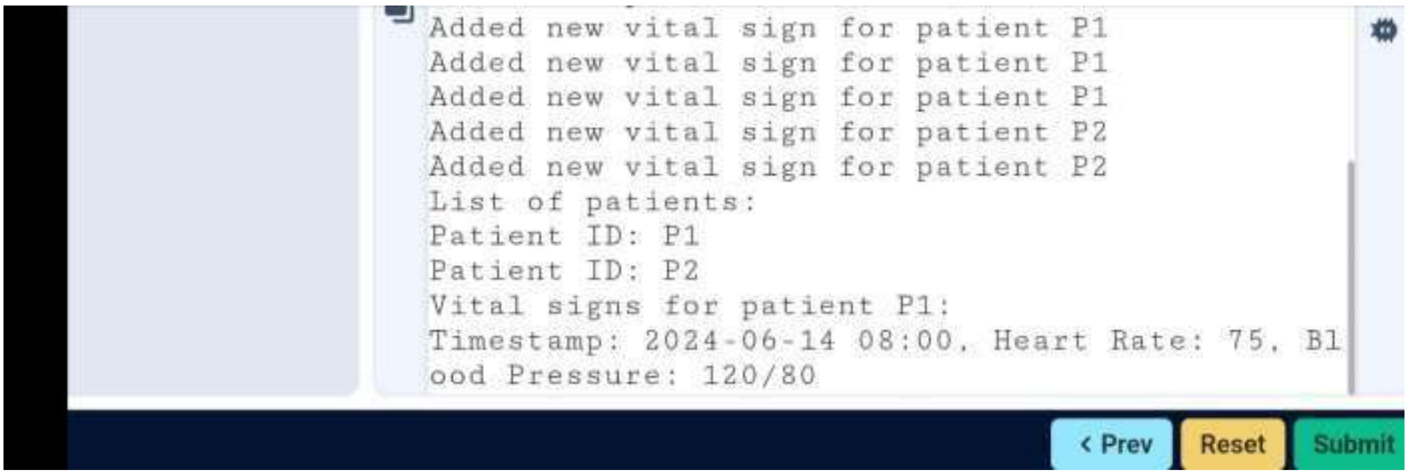
```
>Added new patient with ID P1
Added new patient with ID P2
Added new vital sign for patient P1
Added new vital sign for patient P1
Added new vital sign for patient P1
Added new vital sign for patient P2
Added new vital sign for patient P2
█
```

DISPLAYING ALL PATIENTS



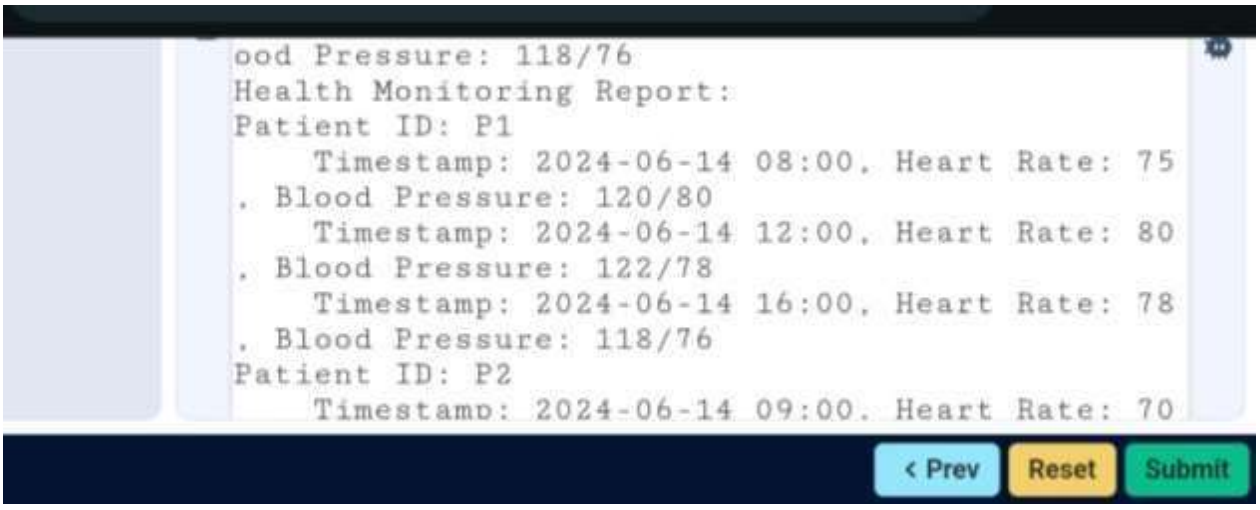
```
Timestamp: 2024-06-14 08:00, Heart R
ood Pressure: 120/80
>Added new patient with ID P1
Added new patient with ID P2
Added new vital sign for patient P1
Added new vital sign for patient P1
Added new vital sign for patient P1
Added new vital sign for patient P2
Added new vital sign for patient P2
List of patients:
Patient ID: P1
Patient ID: P2
```

DISPLAYING VITAL SIGNS FOR A SPECIFIC PATIENT



```
Added new vital sign for patient P1
Added new vital sign for patient P1
Added new vital sign for patient P1
Added new vital sign for patient P2
Added new vital sign for patient P2
List of patients:
Patient ID: P1
Patient ID: P2
Vital signs for patient P1:
Timestamp: 2024-06-14 08:00, Heart Rate: 75, Bl
ood Pressure: 120/80
```

GENERATING A COMPREHENSIVE HEALTH MONITORING REPORT



```
ood Pressure: 118/76
Health Monitoring Report:
Patient ID: P1
    Timestamp: 2024-06-14 08:00, Heart Rate: 75
, Blood Pressure: 120/80
    Timestamp: 2024-06-14 12:00, Heart Rate: 80
, Blood Pressure: 122/78
    Timestamp: 2024-06-14 16:00, Heart Rate: 78
, Blood Pressure: 118/76
Patient ID: P2
    Timestamp: 2024-06-14 09:00, Heart Rate: 70
```