# *Software Design Specifications*

## *Patient Vitals Management System*

**Version: 1**

| Project Team: | Submission Date: |
| --- | --- |
| Aiman Rizwan – ar08513<br>Hiba Shahid – hs08036<br>Fatima Hasan – fh08026 | 28th April ‘25 |

# Document Sign-Off

| Version | Sign-off Authority | Project Role | Signature | Sign-off Date |
|---|---|---|---|---|
| 1.04 | Ateeb Ahmed | Supervisor | | 23rd Apr'18 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Document Information

| Category | Information |
|---|---|
| Customer | Hospitals |
| Project | Patient Vitals Management System |
| Document | Software Design Specification |
| Document Version | 1 |
| Identifier | PVMS-2025 |
| Status | Draft |
| Author(s) | Aiman Rizwan, Hiba Shahid, Fatima Hasan |
| Approver(s) | Ateeb Ahmed |
| Issue Date | 28th April ‘25 |
| Document Location | Karachi |
| Distribution | Advisor |

# Table of Contents

# 1 Introduction

## *1.1 Purpose of Document*

This document provides the design specifications of the Patient Vitals Monitoring System. It outlines both the high-level and low-level structure of the system, including the interaction between end-users (doctors, nurses, and admin staff) and the system, as well as communication between various components like the data generators, anomaly detection engine, dashboards, and the database.

## *1.2 Intended Audience*

This document is intended to be reviewed by the project sponsor, academic supervisors, and particularly, the development or IT teams responsible for deployment, maintenance, or future enhancements. It introduces them to both the process and technical structure of the system, should they intend to make modifications based on changing healthcare or technological requirements.

## *1.3 Document Convention*

Font: Arial
Size: 10
1.5 Space Gapping

## *1.4 Project Overview*

The Patient Vitals Monitoring System is a healthcare-focused application developed to address the challenges of manually tracking patient vital signs, such as heart rate, blood pressure, body temperature, and oxygen saturation. Previously, these metrics were recorded manually by nursing staff, making them prone to human error and delays. The new system provides real-time data monitoring using synthetic and real datasets, with automatic alert generation for anomalies based on predefined thresholds. Medical professionals access this data via customized dashboards that allow both current status monitoring and historical trend analysis, significantly improving clinical decision-making and response times.

## *1.5 Scope*

- To automatically record and store patient vitals from simulated data sources in real time.

- To display vitals dashboards tailored for different user roles (nurses see real-time only; doctors see historical + real-time).
- To generate alerts when vital signs cross safe thresholds and escalate those alerts based on severity.
- To allow healthcare staff to view historical trends of vitals for clinical assessment.
- To implement role-based access control to restrict or allow specific user capabilities within the system.

# 2 Design Considerations

## *2.1 Assumptions and Dependencies*

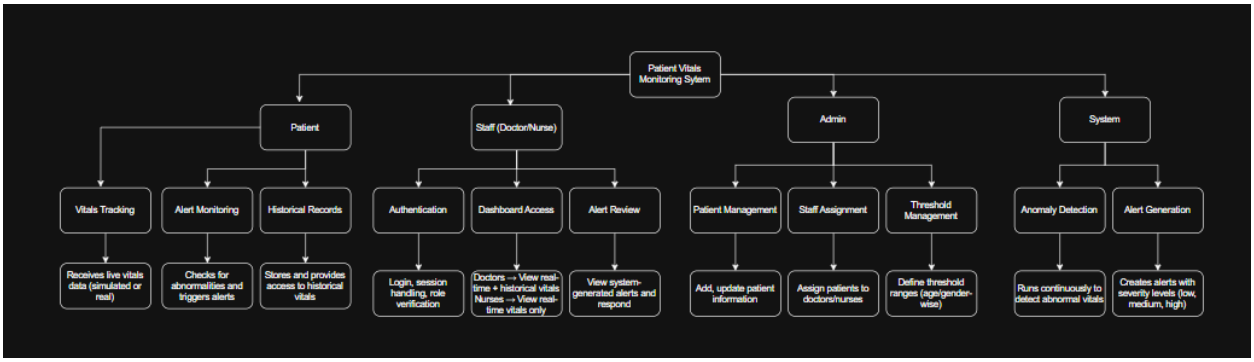
The proper functioning of the Patient Vitals Monitoring System is based on several software and hardware assumptions. It is assumed that the system will be deployed on a computer or server equipped with a compatible Python environment, including Flask for the backend, SQLAlchemy for ORM-based database management, and access to a SQLite database. In terms of performance, the system requires sufficient RAM and CPU resources to handle the continuous generation and analysis of real-time patient data. A stable TCP/IP network connection is necessary to ensure seamless communication between backend services and the frontend interface. For visualization and user interaction, it is assumed that healthcare staff will access the dashboards using modern web browsers such as Google Chrome or Mozilla Firefox. Furthermore, the system relies on the assumption that both synthetic and real patient data will be correctly formatted and reliably inserted into the database using pre-configured automated scripts. Any deviation from these assumptions may impact the accuracy and usability of the system.

## *2.2 Risks and Volatile Areas*

The most sensitive and volatile components of the system include the vitals data, threshold ranges, alert records, and access control mappings. Since vitals data is updated in real time, any corruption or interruption in this data stream—especially involving timestamps—can directly affect the system's ability to detect anomalies accurately or generate proper visualizations. Threshold values are critical to alert generation; any improper updates to these gender- and age-based thresholds may result in false alarms or missed critical warnings. Alerts themselves are highly sensitive since they trigger clinical responses—if alerts are not generated or recorded accurately, healthcare staff may overlook urgent patient conditions. Finally, the patient-to-staff access mappings govern which doctor or nurse can view or manage a specific patient's information. Errors in these assignments could either prevent critical care or result in the unintended disclosure of confidential medical data. For these reasons, any updates to these components must be handled with caution, preferably by system administrators with proper validation mechanisms in place.

# System Architecture

## *2.3 System Level Architecture*




The decomposition level diagram has been provided to break down the Patient Vitals Monitoring System at a high level in a way that sets the foundation for deeper technical design. It outlines what the system does functionally without delving into implementation specifics. The system is centered around three primary actors: patients, medical staff (doctors and nurses), and administrative users, with a separate layer for automated internal processes.

At the patient level, the system continuously receives and records vital signs, stores historical records, and monitors for abnormalities using pre-defined thresholds. Staff, based on their role, log in to access dashboards that reflect the current and historical health status of patients assigned to them, and they can view alerts when abnormalities are detected.

Admins handle data setup and access control, including adding patients, updating thresholds, and assigning staff to patients. The automated backend process is responsible for anomaly detection and the creation of alerts, ensuring that real-time patient data is evaluated continuously. A reporting module is planned as a future enhancement to generate downloadable patient summaries and trends for clinical use or documentation.

## *2.4 Software Architecture*

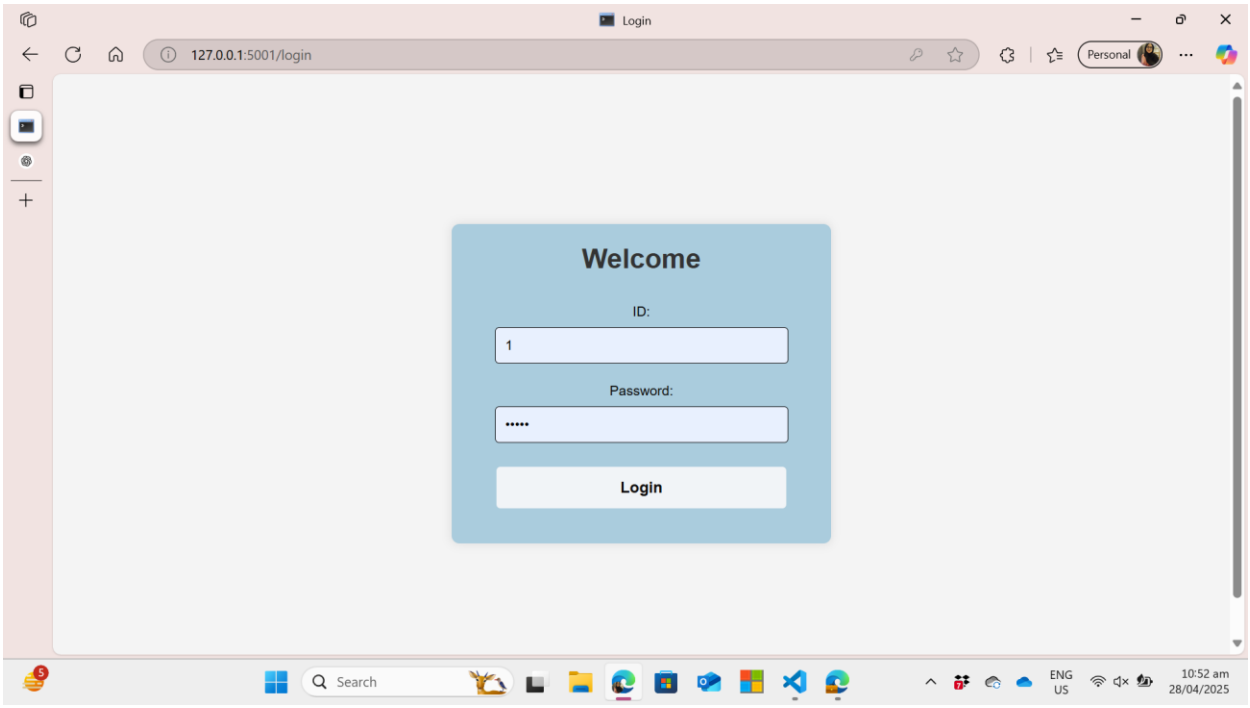



**Figure 1: Login screen**

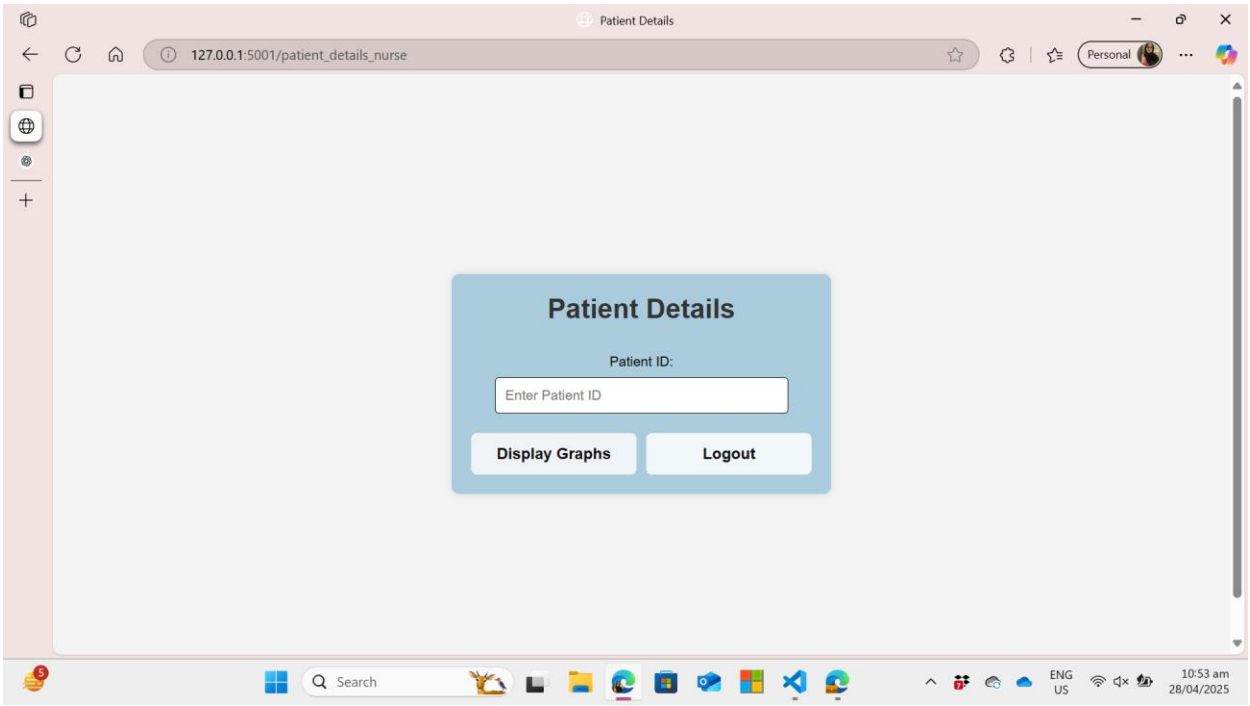



**Figure 2: Patient Details access page for non-doctors (nurse, head-nurse)**

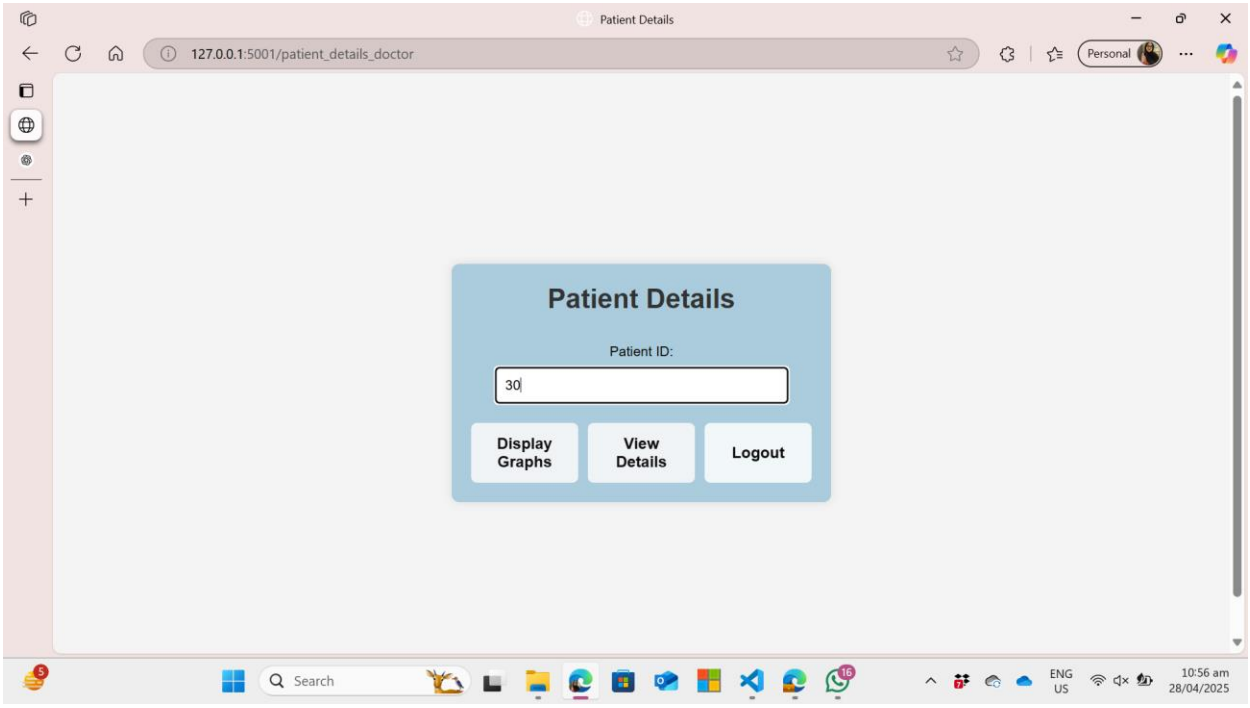



**Figure 3: Patient Details access screen for doctors**

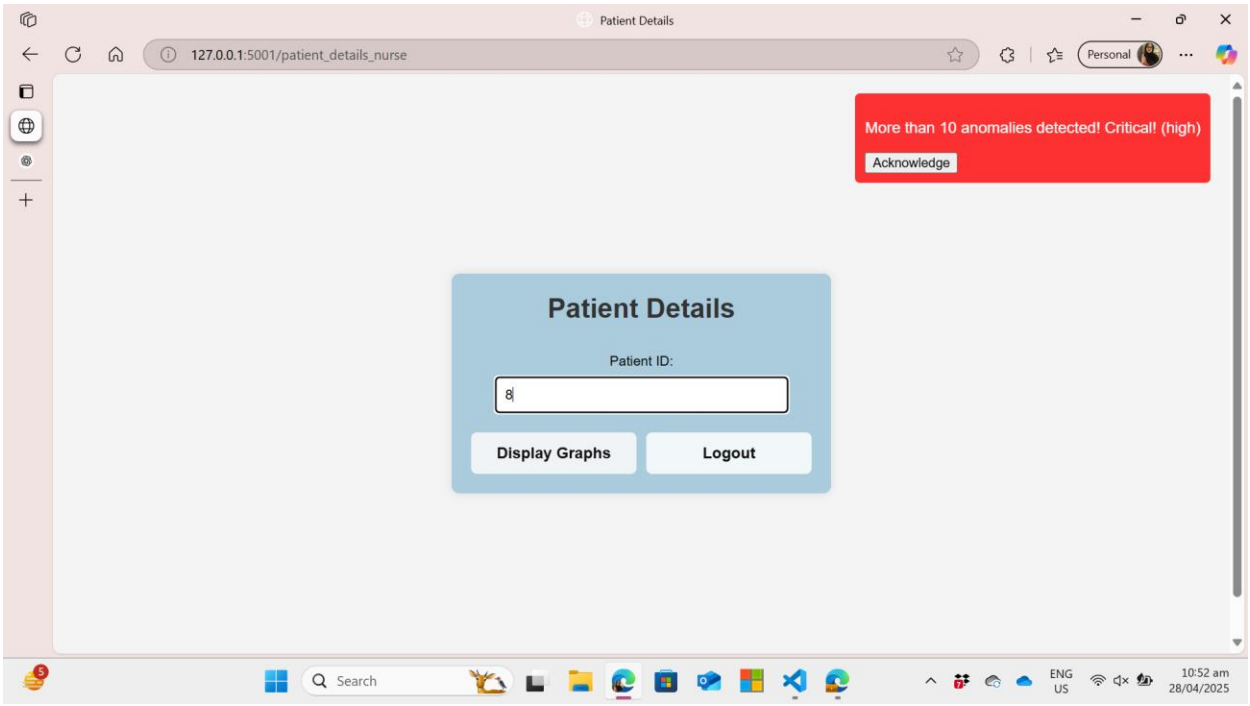



**Figure 4: Alert shows as pop-up (same on the doctor access page as well)**

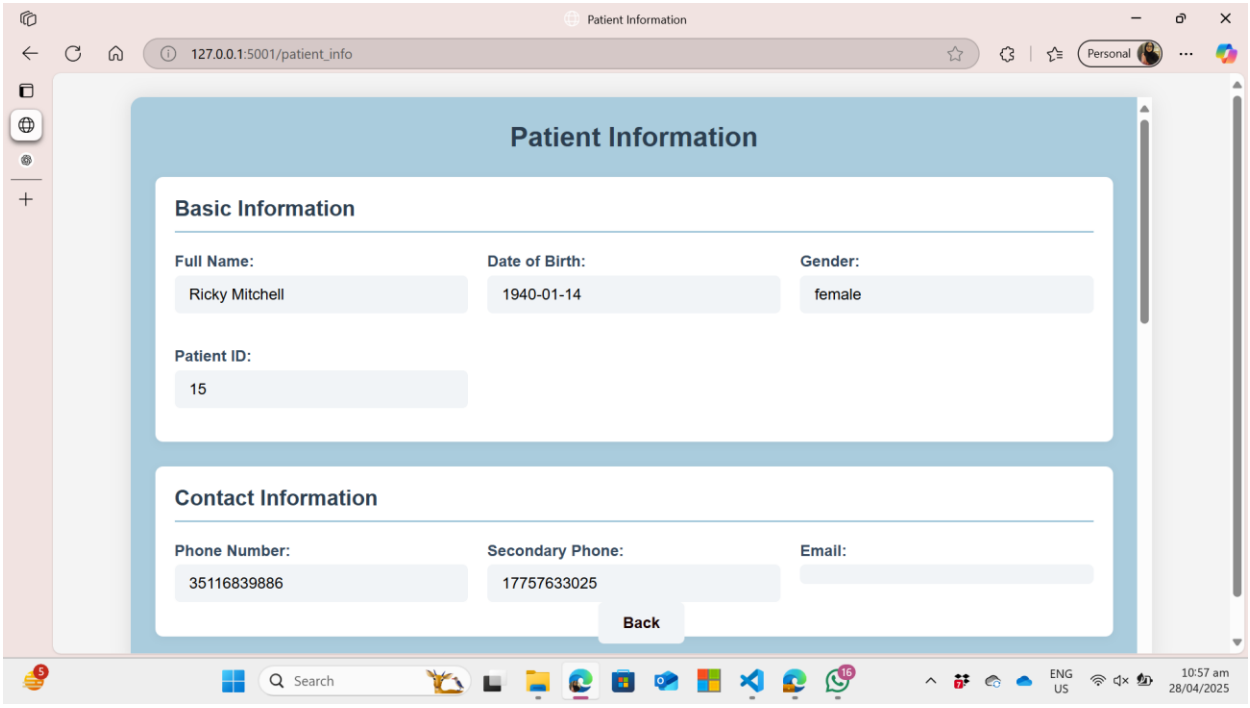



**Figure 5(a): Patient information/report display**

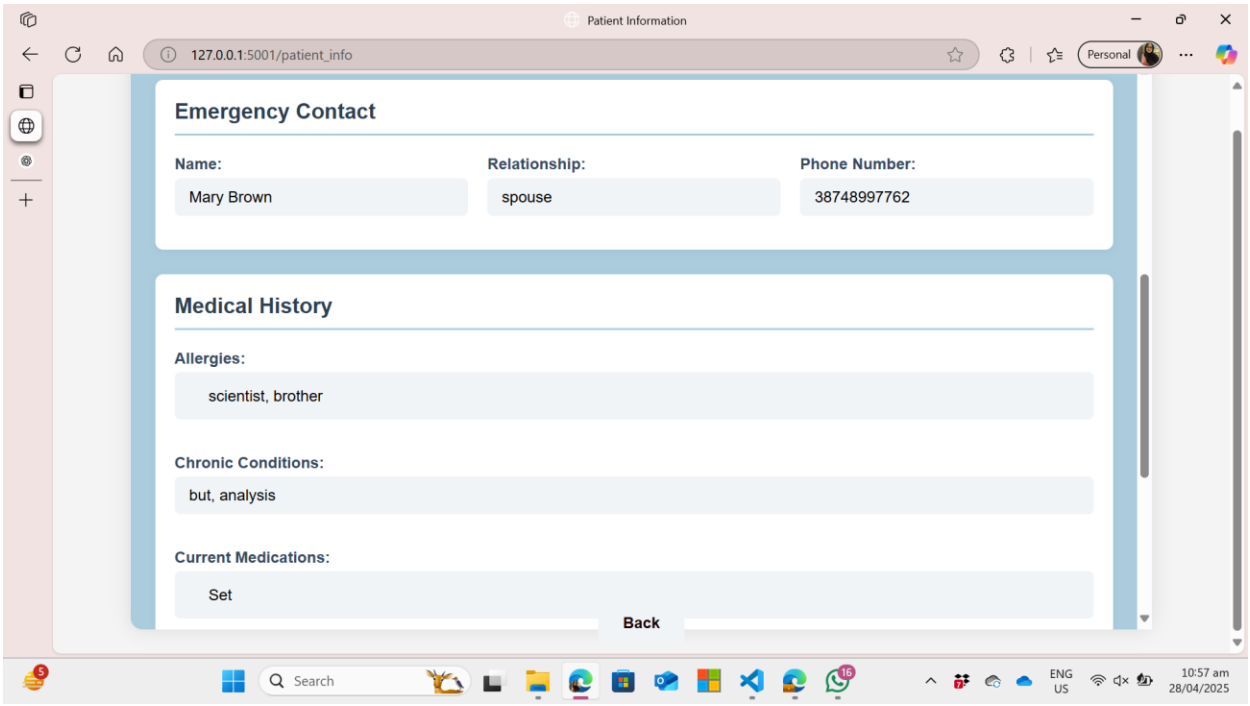



**Figure 5(b): Patient information/report display**

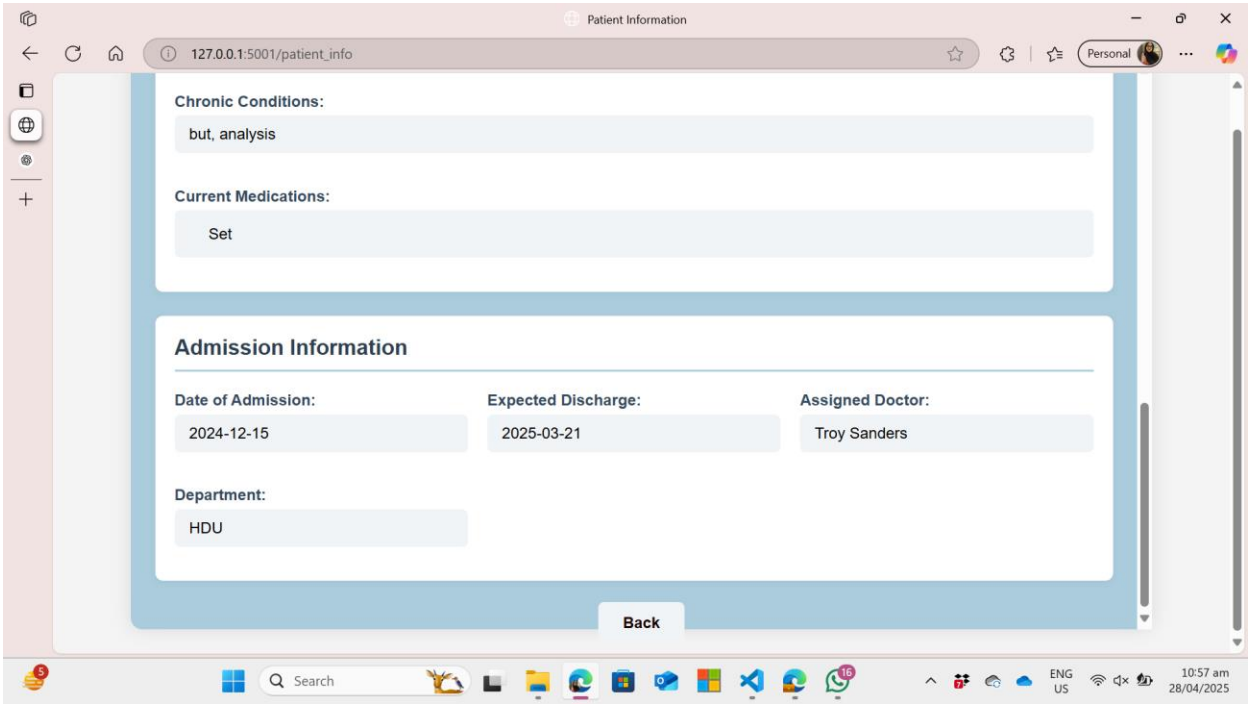



**Figure 5(c): Patient information/report display**

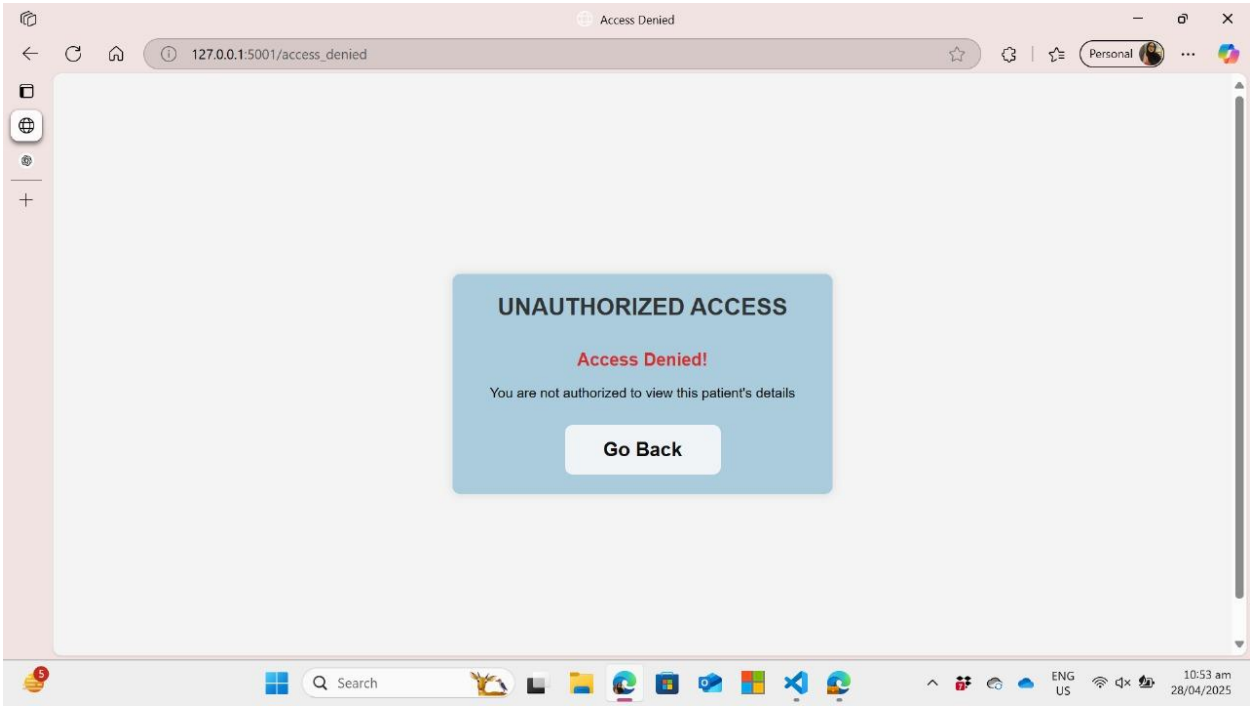



**Figure 6: Unauthorized access alert screen**

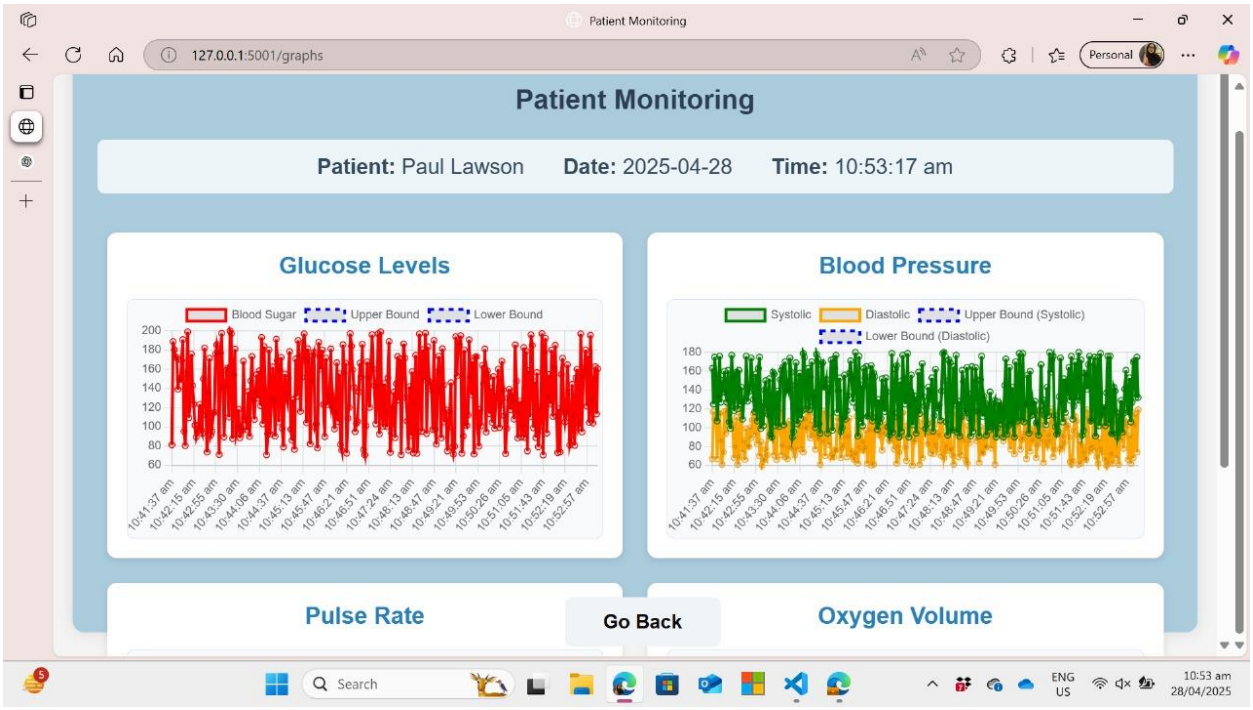



**Figure 7(a): Patient Vitals graph display screen**

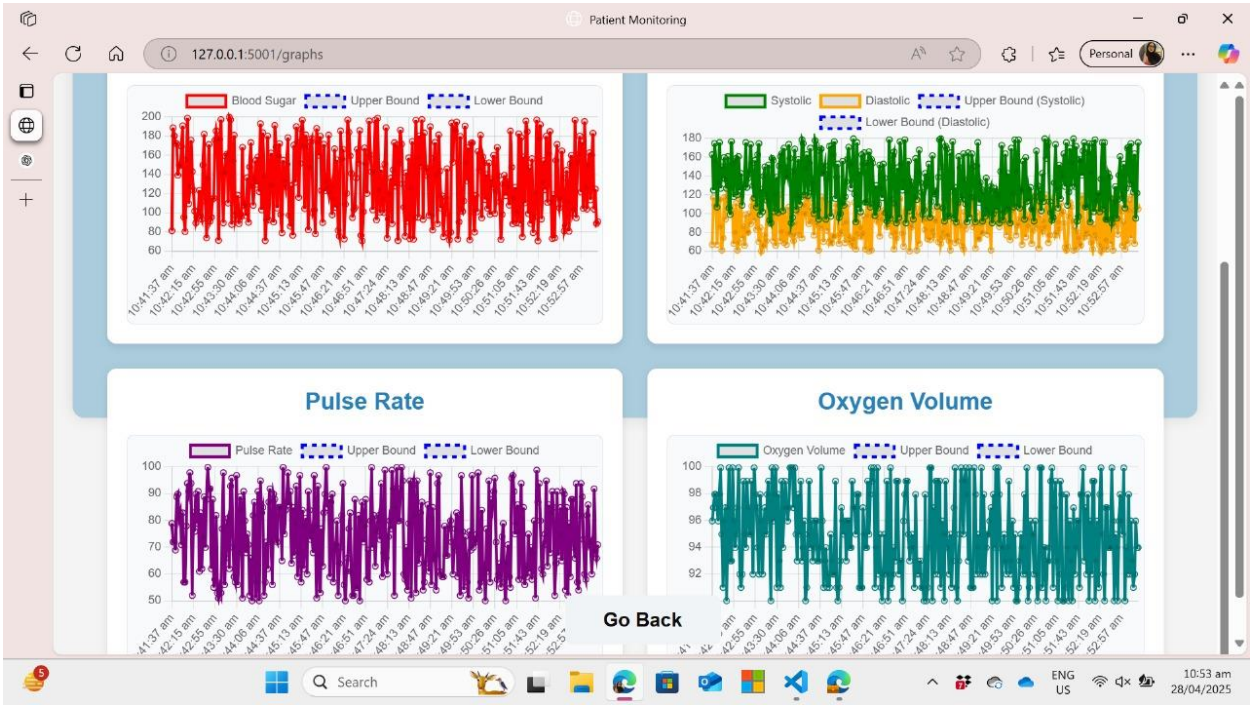



**Figure 7(b): Patient Vitals graph display screen**

# 3 Design Strategy

The system was developed as a web-based application using Python with Flask for the backend and modern HTML/CSS/JavaScript for the frontend. This approach was selected to ensure:

- Cross-platform accessibility from any modern web browser
- Real-time data visualization capabilities
- Secure role-based access control
- Easy deployment in hospital network environments
- The technology stack includes:
- Backend: Python 3.7+ with Flask framework
- Database: SQLite (with PostgreSQL compatibility)
- ORM: SQLAlchemy for database interactions
- Frontend: Chart.js for real-time vital sign visualization
- Authentication: Werkzeug security for password hashing

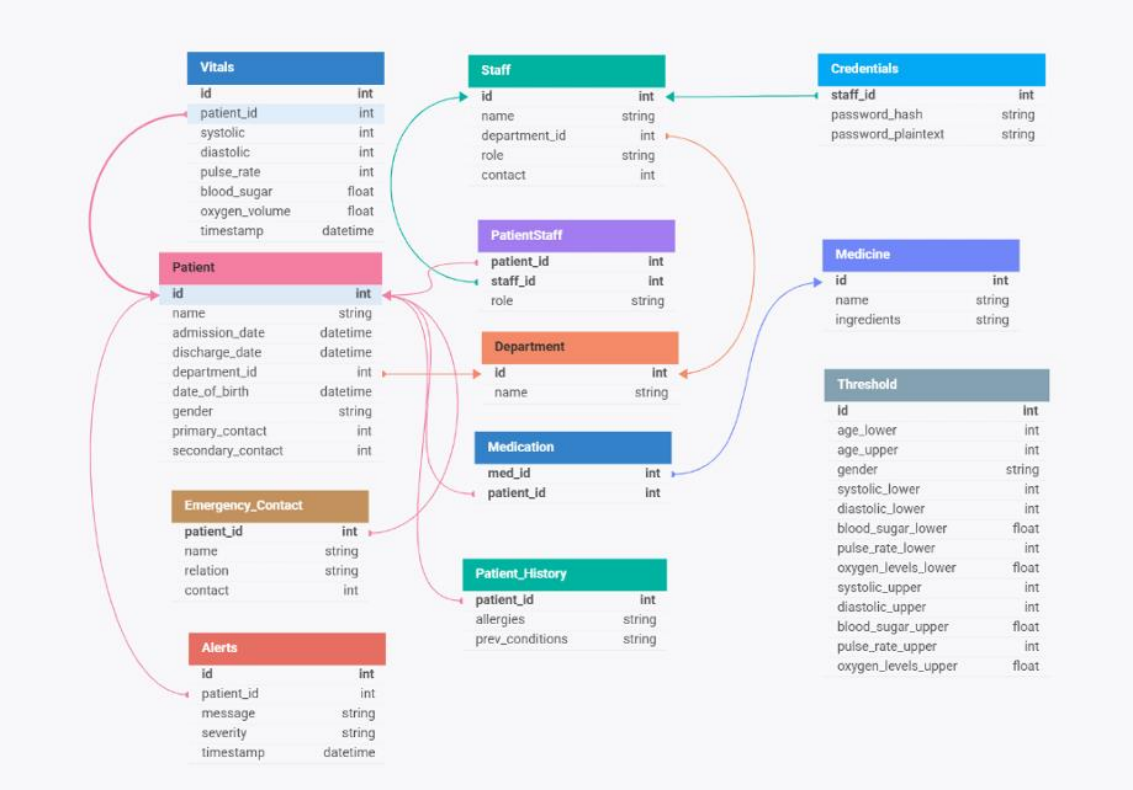
The system is designed with extensibility in mind, allowing for future enhancements such as:

- Mobile application integration
- EHR system interoperability
- Advanced analytics modules
- Multi-hospital deployment capabilities

# 4 Detailed System Design

## *4.1 Database Design*

### 4.1.1 ER Diagram



## 4.1.2 Data Dictionary

### 4.1.2.1 Data 1 – Patient Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INT | PK | unique patient identifier |
| name | VARCHAR(100) | not null | full name of patient |
| admission_date | DATETIME | not null | date/time of hospital admission |
| discharge_date | DATETIME | nullable | date/time of discharge (null if still admitted) |
| department_id | INT | FK | department where patient is admitted |
| date_of_birth | DATETIME | not null | patients birth date |
| gender | VARCHAR(10) | not null | patients gender |
| primary_contact | VARCHAR(15) | not null | primary phone no. |
| secondary_contact | VARCHAR(15) | nullable | secondary phone no. |

### 4.1.2.2 Data 2 – Vitals Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INT | PK | unique vital reading id |
| patient_id | INT | FK | associated patient id |
| systolic | INT | not null | systolic blood pressure |
| diastolic | INT | not null | diastolic blood pressure |
| pulse_rate | INT | not null | heart rate |
| blood_sugar | FLOAT | not null | blood gluscose level |
| oxygen_volume | FLOAT | not null | oxygen saturation |
| timestamp | DATETIME | not null | time of measurement |

### 4.1.2.3 Data 3 – Staff Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INT | PK | staff member id |
| name | VARCHAR(100) | not null | staff member name |
| department_id | INT | FK | department assignment |
| role | VARCHAR(20) | not null | role(doctor/nurse/admin) |
| contact | VARCHAR(15) | not null | contact number |

#### 4.1.2.4 Data 4 – Emergency_Contact Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| patient_id | INT | PK, FK | associated patient id |
| name | VARCHAR(100) | not null | contact persons name |
| relation | VARCHAR(50) | not null | relationship to patient |
| contact | VARCHAR(15) | not null | emergency contact number |

#### 4.1.2.5 Data 5 – Alert Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INT | PK | Alert id |
| patient_id | INT | FK | associated patient id |
| message | VARCHAR(255) | not null | alert description |
| severity | VARCHAR(10) | not null | severity |
| timestamp | DATETIME | not null | time of alert |

#### 4.1.2.6 Data 6 – PatientStaff Table (Assignment)

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| patient_id | INT | PK, FK | pateint id |
| staff_id | INT | PK, FK | staff id |
| role | VARCHAR(20) | not null | role |

#### 4.1.2.7 Data 7 – Threshold Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INT | PK | THRESHOLD ID |
| age_lower | INT | not null | minimum age for range |
| age_upper | INT | not null | minimum age for range |
| gender | VARCHAR(10) | not null | gender |
| systolic_lower | INT | not null | minimum normal systolic bp |
| systolic_upper | INT | not null | minimum normal systolic bp |
| diastolic_lower | INT | not null | minimum normal diastolic bp |
| diastolic_upper | INT | not null | minimum normal diastolic bp |
| blood_sugar_lower | FLOAT | not null | minimum normal gluscose |
| blood_sugar_upper | FLOAT | not null | minimum normal gluscose |
| pulse_rate_lower | INT | not null | minimum normal heart rate |
| pulse_rate_upper | INT | not null | minimum normal heart rate |
| oxygen_levels_lower | FLOAT | not null | minimum normal spO2 |
| oxygen_levels_upper | FLOAT | not null | minimum normal spO3 |

#### 4.1.2.8 Data 8 – Credentials Table

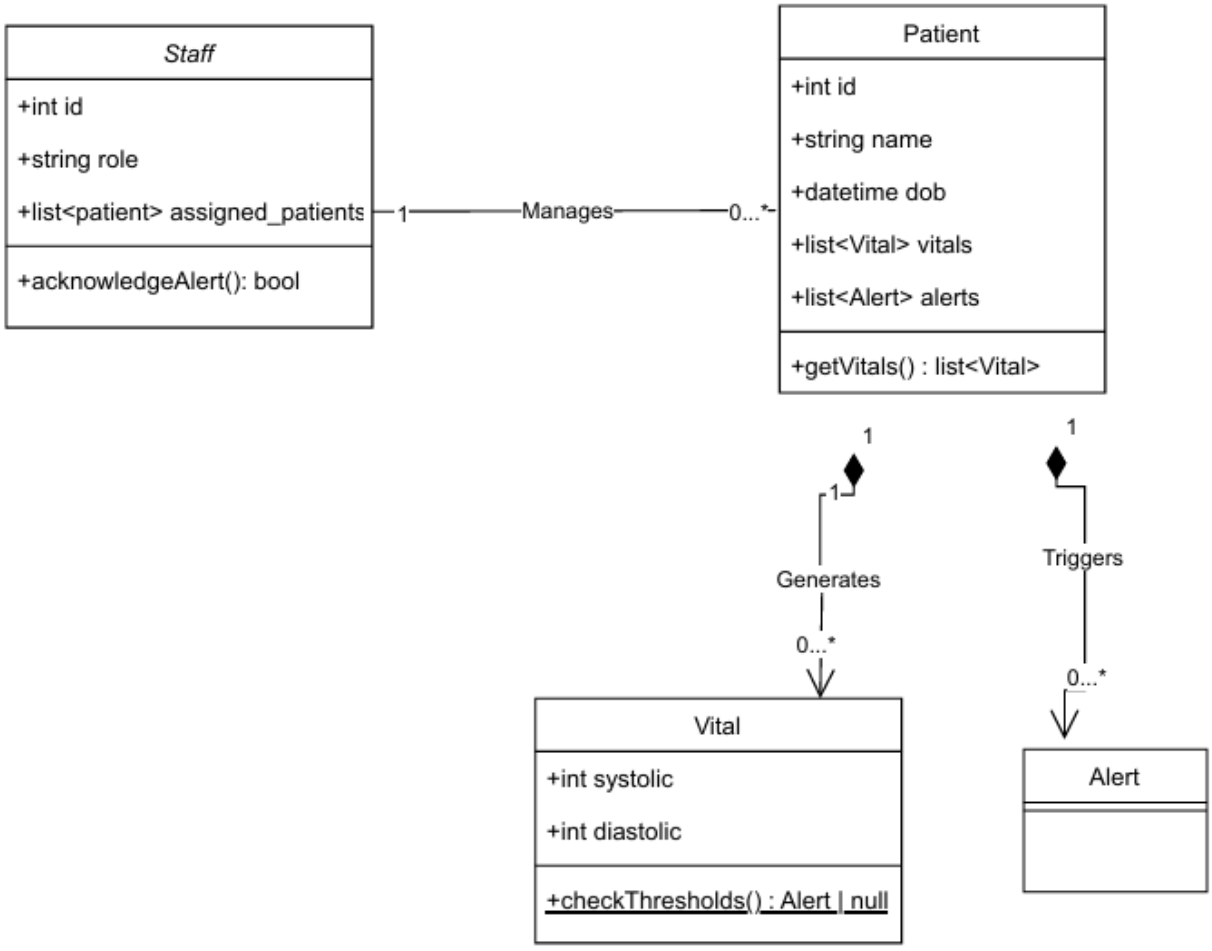
| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| staff_id | INT | PK, FK | staff member id |
| password_hash | VARCHAR(128) | not null | hashed password |
| password_plaintext | VARCHAR(50) | nullable | plain text password |

#### 4.1.2.9 Data 9 – Department Table

| Field Name | Data Type | Constraint | Description |
|---|---|---|---|
| id | INT | PK | department id |
| name | VARCHAR(50) | not null | department name |

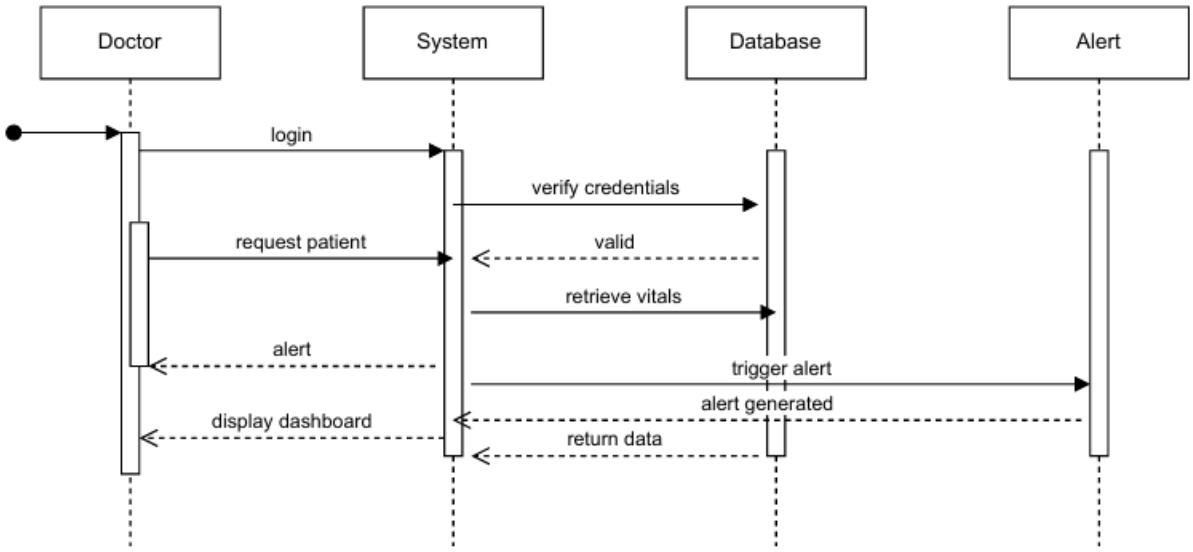
## *4.2 Application Design*

### 4.2.1 Class Diagram



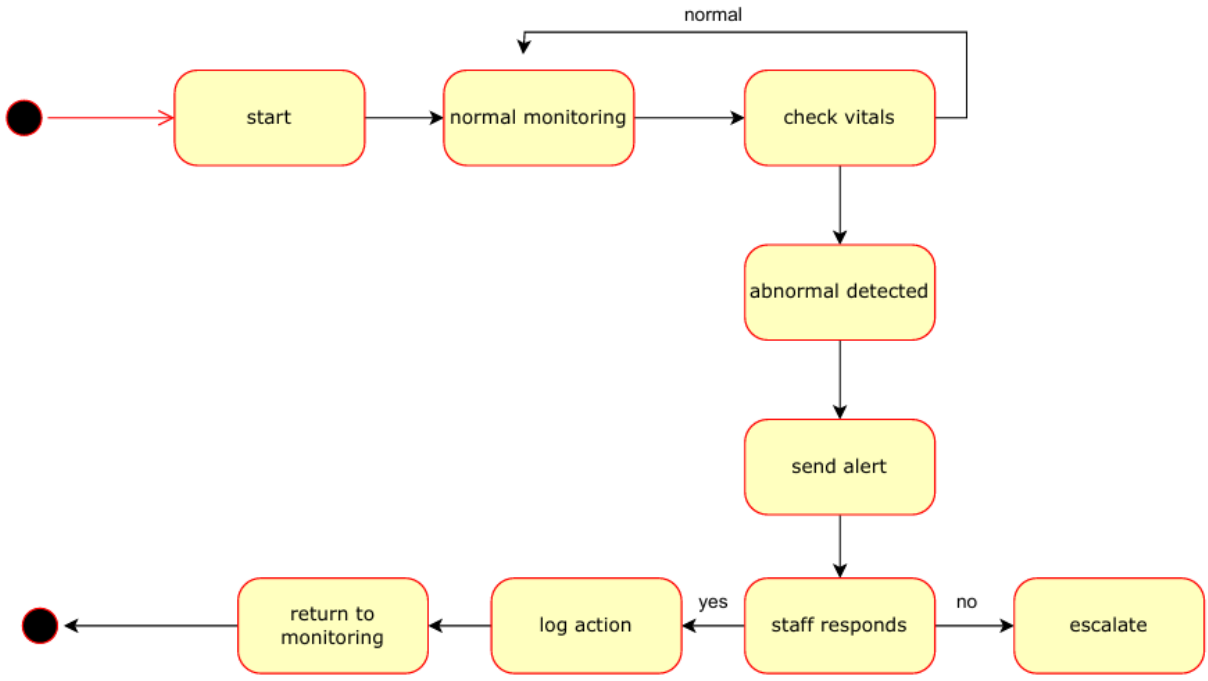
## 4.2.2 Sequence Diagram

### 4.2.2.1 Sequence Diagram 1 – Client Actor



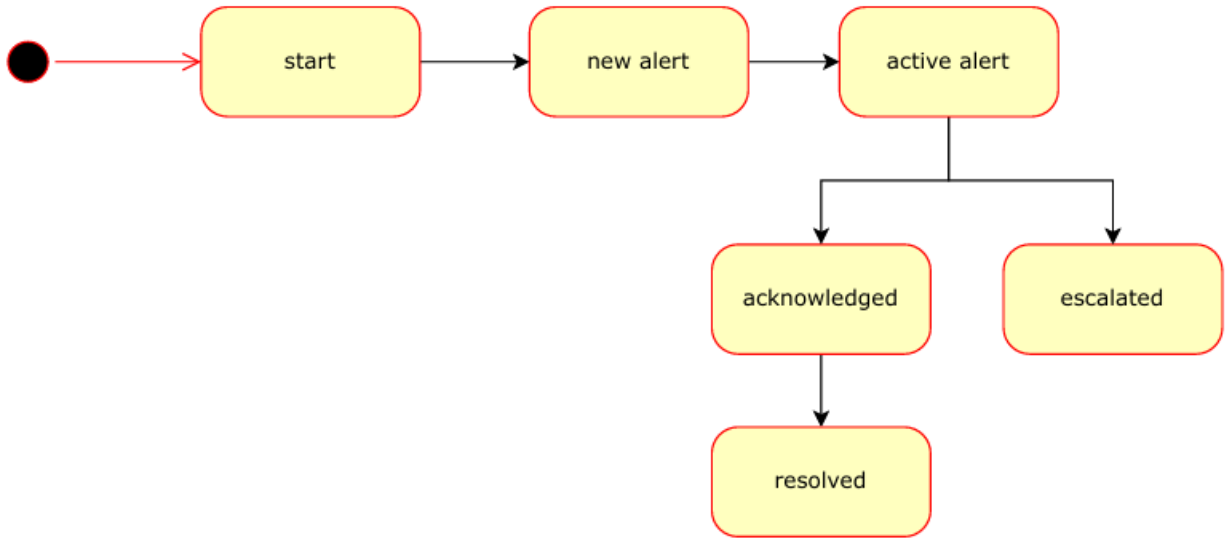
## 4.2.3 State Diagrams

### 4.2.3.1 Patient Vital Monitoring State




### 4.2.3.2 Alert state

.



### 4.2.4 System User Interface

Not Applicable.

# 5 References

The template for the SDS was given by Mr. Ateeb Ahmed, everything else is original.

# 6 Appendices

Not Applicable.