A PRELIMINARY PROJECT REPORT ON

HEALTHWATCH: ICU MONITORING AND CLOUD INTEGRATION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

of

SAVITRIBAI PHULE PUNE UNIVERSITY

By

MUKTA BAHULEKAR **Exam Seat No: B1902304221**

CHITRARTH KADEL **Exam Seat No: B1902304296**

SAHIL PHULE **Exam Seat No: B1902304362**

SWAYAM SONAR **Exam Seat No: B1902304381**

Under the guidance of

DR. M. P. WANKHADE

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING
SINHGAD COLLEGE OF ENGINEERING, PUNE-41
Accredited by NAAC with A+ Grade

2024-25

Sinhgad Technical Education Society,
Sinhgad College of Engineering, Pune-41
Department of Computer Engineering

Sinhgad Institutes

Date:

CERTIFICATE

This is to certify that the preliminary project report entitled

"HealthWatch: ICU Monitoring and Cloud Integration"

Submitted by

Mukta Bahulekar	**Exam Seat No : B1902304221**
Chitrarth Kadel	**Exam Seat No: B1902304296**
Sahil Phule	**Exam Seat No: B1902304362**
Swayam Sonar	**Exam Seat No : B1902304381**

is a bonafide work carried out by them under the supervision of Dr. M. P. Wankhade and it is approved for the partial fulfilment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2024-25.

Dr. M. P. Wankhade
Project Guide and Head Of Department
Department of Computer Engineering

Dr. S. D. Lokhande
Principal
Sinhgad College of Engineering

Acknowledgements

The success and outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project report. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank Dr. M.P. Wankhade, HOD of Computer Engineering, for providing us with this wonderful opportunity and providing us with all the support and guidance, which made us complete the project report duly. We are extremely thankful to him for providing support and guidance.

We are thankful and fortunate enough to get constant encouragement, support and guidance from all Teaching staff of the Department of Computer Engineering which helped us in completing our project. At last, we would like to thank all the unseen authors of various articles on the Internet, for helping us become aware of the research currently ongoing in this field and all my colleagues for providing help and support in our work.

Abstract

The healthcare sector increasingly relies on advanced technology to enhance patient care, especially in critical care environments such as Intensive Care Units (ICUs). HealthWatch is a web-based, cloud-integrated patient monitoring system designed to revolutionise ICU patient management by providing real-time vital sign monitoring, secure data storage, customisable alert system and advanced analytics. This system enables healthcare providers to access up-to-the-minute patient data from anywhere, reducing response times and improving overall patient outcomes.

HealthWatch integrates directly with ICU monitors to capture essential patient data, such as heart rate, ECG, blood pressure, temperature and oxygen levels and stores this information securely in a cloud-based system. The centralised dashboard provides a comprehensive view of all patients in the ICU, enabling doctors and nurses to monitor multiple patients simultaneously. The system features a customisable alert mechanism that notifies medical staff when a patient's vitals exceed predefined thresholds, allowing for rapid intervention.

A key feature of HealthWatch is its data analytics and machine learning capabilities, which analyse stored data to detect patterns and provide predictive insights. This functionality helps healthcare providers anticipate potential complications, improving decision-making and patient care. The system also incorporates role-based access control to ensure that sensitive patient data is only accessible to authorised personnel. By automating data collection, storage, and alerting processes, HealthWatch reduces the risk of manual errors, ensures data accuracy, and streamlines ICU workflows.

In summary, HealthWatch aims to enhance ICU operations by delivering a reliable, real-time, cloud-based monitoring solution that supports critical decision-making, ensures patient safety, and improves the efficiency of healthcare services.

List of Figures

List of Tables

Abbreviation

HTML	Hypertext Markup Language
CSS	Cascading Sheet Styles
JS	JavaScript
UI	User Interface
UML	Unified Modeling Language
ML	Machine Learning
SVM	Support Vector Machine
KNN	K Nearest Neighbor

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

1.1 Background and basics

In the rapidly evolving field of healthcare, the integration of technology is critical to ensuring patient safety and improving outcomes. This is especially true in Intensive Care Units (ICUs), where the most critically ill patients require constant and meticulous monitoring. ICUs are high-pressure environments where swift, informed decisions can mean the difference between life and death. Continuous monitoring of vital signs such as heart rate, blood pressure, respiratory rate, and oxygen levels is essential in detecting early signs of patient deterioration, allowing medical staff to intervene promptly. Accurate, real-time data monitoring is crucial for managing complex medical conditions, as even minor delays or manual errors can lead to adverse outcomes. As healthcare systems strive to improve operational efficiency and patient care, advanced solutions like HealthWatch: ICU Monitoring and Cloud Integration offer a powerful tool to automate vital monitoring, ensure data accuracy, and provide actionable insights to healthcare providers. By leveraging cloud technology and data analytics, HealthWatch enhances ICU operations, ensuring that critical patient information is always accessible, secure, and up-to-date.

1.2 Literature Survey

1.2.1 B. Joel, S Sibi Rajan, R Vibinanth, D. Pamela, and P. Manimegalai, "I-Doc – A Cloud Based Data Management System For Health Care,", Apr. 2022:

This paper discusses the development of a cloud-based system for managing patient data, which is closely aligned with our goal of integrating cloud storage for ICU patient monitoring. The emphasis on using a cloud-based Electronic Medical Record (EMR) system provides valuable insights into designing and implementing scalable cloud infrastructure [1].

1.2.1.1 Key Learnings:

- Cloud Integration: The paper demonstrates how hospitals can move from on-premise data centres to cloud solutions, which helps reduce maintenance costs and improve scalability—critical elements for HealthWatch.

- Data Security: The paper emphasises the importance of securing patient data in the cloud, which will guide us in applying advanced encryption and secure access protocols in our project.
- User Interface (UI): By suggesting the use of simplified user interfaces built with Bootstrap and Electron frameworks, this paper provides useful ideas on creating an intuitive and user-friendly design for healthcare professionals.

1.2.1.2 Shortcomings:

- The paper focuses on managing health records rather than real-time data monitoring. Our system must focus heavily on real-time data collection and analysis, especially since ICU environments demand immediate responses to changing patient conditions.
- The paper mentions data security but does not provide extensive solutions for scaling patient data across multiple healthcare facilities.

1.2.2 Donthula Akhil, Marepalli Vishnu Vardhan, K. V. Reddy, Chapala Preetham, and N Sangeeta, "Iot Based Icu Patient Monitoring Smart System," May 2024:

This paper presents an IoT-based system for monitoring ICU patients in real-time using sensors for temperature, ECG, and respiratory rates. It aligns directly with our project's objective of building a smart ICU monitoring system [2].

1.2.2.1 Key Learnings:

- Sensor Integration: The paper details how sensors are integrated with a microcontroller (ESP8266) for real-time data collection, which is very similar to our approach using ESP32. This helps us better understand sensor calibration and wireless data transmission protocols.
- Cloud Integration via AWS: Their use of AWS services, including DynamoDB, MQTT, and SNS, to store and analyse patient data provides a framework for our project's cloud-based architecture.
- Machine Learning for Prediction: The use of machine learning models to predict patient health outcomes based on sensor data offers valuable insights into how we can apply similar algorithms in HealthWatch to enhance predictive analysis.

1.2.2.2 Shortcomings:

- While this paper discusses the use of machine learning, it mainly focuses on alert systems rather than in-depth predictive analytics.

- The system described in this paper offers basic alerts based on patient vitals, without much flexibility for customization by healthcare providers.

1.2.3 M. D. Aldhoayan and Y. Aljubran, "Prediction of ICU Patients' Deterioration Using Machine Learning Techniques,", May 2023:

This paper evaluates the use of ML models (Gradient Boosting, KNN, SVM, Naive Bayes, etc.) to predict ICU patient deterioration based on vital signs such as blood pressure, heart rate, respiratory rate, and oxygen saturation [3].

1.2.3.1 Key Learnings:

- It gives insight into monitoring ICU patients and predicting deterioration using real-time vital signs, similar to HealthWatch's goals.
- It provides comparative performance insights, which can guide model selection for early ICU deterioration warnings.

1.2.3.2 Shortcomings:

- The ICU deterioration paper used static models on historical data with a small feature set.

1.2.4 C. V. Ravikanth, Kolangiammal, K. Chegondi, and V. Gowtham, "Cardio Monitoring and Cardiac disease prediction using Machine Learning,", Apr. 2023:

This paper focuses on integrating sensors (ECG, heart rate, temperature, glucose) with IoT platforms to monitor patient vitals and predict cardiac disease using ML models (KNN and SVM) [4].

1.2.4.1 Key Learnings:

- Sensor Integration Framework: It provides a roadmap on how to gather vital signs from sensors and transmit them to the cloud for processing.
- ML-based Cardiac Predictions: Using SVM and KNN to predict heart disease aligns with the goal of integrating cardiac monitoring in ICU patients.
- Data Preprocessing and Model Selection: It explores data preparation, training, and validation approaches, which are useful in refining HealthWatch's predictive models.

1.2.4.2 Shortcomings:

- The cardiac monitoring paper uses traditional ML models (SVM, KNN) with a relatively small dataset.
- Lack of real-time adaptive prediction.

1.2.5 On-site Hospital Survey

On August 28th, 2024, we carried out a detailed survey at Smt. Kashibai Navale Medical College and General Hospital. The survey lasted for about two hours, during which we visited various wards in the hospital, including the ENT ward, Medicine ward, Paediatric ward, General Surgery ward, and the ICU. During these visits, we gathered valuable insights that will significantly contribute to enhancing our project.

1.2.5.1 Findings

- The ICU uses a vital monitoring machine called Infinity Delta, developed by Draeger India Pvt. Ltd.
- This machine tracks critical vitals like ECG, heart rate, SpO2, pulse, and respiration, which are continuously monitored by nurses and doctors, as ICU patients require constant care.
- The data from these machines is manually entered into the hospital's main system.
- There is a centralised system in the ICU that displays basic information of all patients on a single screen.
- A basic alarm system is integrated into the machines, which alerts staff through beeping sounds if any abnormality in the vitals is detected.

1.2.5.2 Shortcomings

- Data is manually recorded and entered into systems, increasing the risk of errors and delays in patient care.
- Alerts are non-customizable, leading to delayed or missed critical responses in patient condition changes.
- Limited capacity for storing and analysing historical patient data, hindering trend analysis and long-term care decisions.
- The interface is non-intuitive and inflexible, slowing down workflow and making it harder to access critical information quickly.
- No effective remote monitoring capability restricts healthcare providers from accessing data outside the hospital.
- No advanced tools for analysing trends or predicting patient deterioration, limiting proactive healthcare interventions.

1.3 Project Undertaken

The main aim of the project is to address the lack of a centralised system for real-time ICU patient monitoring, data storage and analytics, alerting, and triggering. Additional features are added to streamline the entire patient monitoring process and make healthcare more efficient.

1.3.1 Problem Definition

To address the lack of a centralized system for real-time ICU patient monitoring, data storage & analytics, alerting & triggering.

1.3.2 Scope Statement

The project's scope encompasses the development of a sensor integration system, a database, and a web application designed for the following purposes:

1.3.2.1 Centralization of Various Healthcare Equipment:

Healthcare relies on numerous components supplied by different manufacturers, each with distinct interfaces, protocols, and output formats. Our system aims to centralise these devices, providing a cohesive platform for their interaction with minimal coding required for maintenance in the event of a malfunction.

1.3.2.2 Data Management:

Current systems often lack adequate data management and storage capabilities. Most machines store data locally for only two days, resulting in a manual process as records are not maintained on a centralised system. Furthermore, data logging is typically absent in general and non-emergency wards, making it crucial to document information to address any future health complications effectively.

1.3.2.3 Event Management:

In emergencies, a doctor may not be notified until their scheduled duty time, requiring another available physician to attend to the patient. Often, a specific doctor's familiarity with a patient’s medical history enables them to determine the best course of action. Our event management feature will facilitate notifications based on specific parameters set by physicians, ensuring they are alerted when necessary.

1.3.2.4 Data Analysis:

The data logged in the database can be invaluable for physicians encountering future complications. This information allows even new doctors to understand a patient's past conditions and treatments. Additionally, the data can be integrated into machine learning algorithms to enhance diagnostic processes, alerting doctors to potential complications that may have been overlooked.

1.4 Organization Of Project Report

This report is organised as follows,

Chapter 1 Introduces the Background and Basics of the Vital Monitoring system, the Literature Survey conducted, about the Project to be undertaken, the Problem Definition and the Scope of the Project.

Chapter 2 gives an overview of Project Planning and Management. It states the detailed Functional Requirements, Non-Functional Requirements, Deployment Requirements, External Interface Requirements and Other Requirements. This chapter also specifies the Project Process Model used to develop this Project, Cost and Effort estimates and Project Scheduling.

Chapter 3 gives the Analysis Study and Designing of the Project. It consists of the IDEA Matrix, Mathematical Model, Feasibility Analysis and all UML diagrams specifying how to build the required system

Chapter 4 gives an overview of Testing. It states which tests will be conducted to ensure proper functionality of the system. It will consist of Unit Testing, Integration testing and Acceptance testing.

Chapter 2

PROJECT PLANNING AND MANAGEMENT

2.1 Introduction

This chapter covers the project planning and management details. It also covers System Requirement specifications. SRS is considered as the base for effort estimations and project scheduling.

2.2 Detailed System Requirement Specification

This section gives a detailed description of all types of requirements to be satisfied by the System to be developed.

2.2.1 System Overview

HealthWatch: ICU Monitoring and Cloud Integration is an advanced system used to monitor, store and analyse patient data. This is a web-based application which will be available to doctors, nurses and other medical personnel, and can be accessed from anywhere. This system will extract data from ICU monitors, and send it to the server, from where the data will be stored on a cloud-based system. This data will be visualised and will be visible to the medical authority. This data will further be analysed using ML algorithms for data patterns and predictive analysis. Following are the main features of the project:

1. Centralised System
2. Easy and intuitive user interface
3. Long Term data storing
4. Customizable Alert System
5. Data Analysis
6. Role-Based Access Control
7. Remote access from any location
8. Message alert system for timely notifications

2.2.2 Functional Requirements

The features described below are organised based on their context of functionality. All the features have equal priority.

2.2.2.1 Real-Time Vital Monitoring

Description and Priority: This feature enables the system to collect, display, and update patient vitals in real time, ensuring that healthcare providers have the most current information available. A centralised system will display all patients' vitals on a single dashboard, making it easier for healthcare staff to monitor multiple patients simultaneously. By clicking on the required Bed ID, individual patient vitals assigned to that bed can be monitored.

Stimulus/Response Sequence: When a patient's vitals are recorded by the ICU monitors, the data is transmitted to the system, which processes and displays it on the centralised monitoring dashboard, allowing healthcare providers to monitor all patients in one place.

Functional Requirements:

- The system must collect patient data from ICU monitors every second.
- The main dashboard should show a concise view of all the beds present in that ICU ward.
- Individual vital monitoring can be done by clicking on the required bed.
- Patient vitals such as ECG, heart rate, blood pressure, SpO2, and respiratory rate must be displayed on the dashboard.
- The dashboard must automatically update to reflect any changes in patient vitals without needing a manual refresh.

Main Flow:

- The user logs into the system and is presented with the main console.
- The user clicks on the Vital Monitoring Dashboard, which brings them to the centralised monitoring dashboard.
- This dashboard displays real-time vitals for all patients, giving healthcare staff an overview of the entire ICU from a single screen.
- If the user wants to monitor a specific patient in detail, they click on the patient's bed ID.
- The system then directs them to the individual patient's vitals screen, where they can see detailed real-time data for that specific patient.

Exceptional Flow:

- The user enters their username and password on the login screen.
- The system checks the database to validate the entered credentials.
- If the username is not found in the database or the credentials are incorrect, the system

denies access.

- A message is displayed to the user: "Access denied. User not found or incorrect credentials. Please try again."
- The user is provided with an option to retry logging in or reset their password, if applicable.
- After a certain number of failed attempts, the system temporarily locks the account and displays an additional message: "Too many failed login attempts. Please try again later or contact the system administrator."

2.2.2.2 Registry

Description and Priority: This feature ensures that information such as patient vitals with patient information and doctor information is securely stored in a cloud-based database for future analysis and recordkeeping.

Stimulus/Response Sequence: Patient data is stored in the database during the admission. When patient vitals are captured, the system sends this data to the cloud-based database for storage. Healthcare providers can retrieve this data anytime to review historical trends.

Functional Requirements:

- The system must store patient data for a minimum of 30 days with the ability to extend the retention period.
- After 30 days, this information will be archived for easier storage.
- Data must be stored securely.
- This registry is only available to the system administrator and the doctor.

Main Flow:

- The user logs into the system and is welcomed by the main console.
- The user clicks on the registry, and the system redirects them to the registry page.
- The user enters the required information into the search fields.
- If the correct information is provided, the system retrieves and displays the requested patient's or doctor's information.

Exceptional Flow:

- The user logs into the system and is welcomed by the main console.

- The user clicks on the registry, and the system redirects them to the registry page.
- If incorrect or incomplete information is entered, the system displays a message indicating that no results were found or prompts the user to check the entered data.

2.2.2.3 Data Analytics and Predictive Insights

Description and Priority: The system uses machine learning algorithms to find trends in data and also analyse patient data and provide predictive insights into potential health risks.

Stimulus/Response Sequence: The doctor gets the overview of data after entering into the main console. To access an individual patient's analysis, the doctor clicks on the patient's bed id.

Functional Requirements:

- The system must apply machine learning algorithms to analyse patient data.
- Along with predictive analysis, the system should be able to recognize the trend in patients' data.
- The system must provide data visualisations such as graphs to illustrate trends.

Main Flow:

- The user logs into the system and is welcomed by the main console.
- The user clicks on the Data Monitors Analytics tab.
- The system generates and displays an overview report of the last 30 days, summarising key analytics data.
- To access individual patient data, the user clicks on the patient's Bed ID.
- The user clicks on the Analysis button to view detailed analytics for that specific patient.

Exceptional Flow:

- The user logs into the system and is welcomed by the main console.
- The user clicks on the Data Monitors Analytics tab.
- If the data for the last 30 days is unavailable or incomplete, the system displays a message: "No data available for the selected period. Please check the data source or select another timeframe."
- If the user clicks on an invalid or unassigned Bed ID, the system displays an error message: "Invalid Bed ID. Please select a valid patient ID."

2.2.2.4 Customizable Alerts and Notifications

Description and Priority: This feature allows healthcare providers to customise alert thresholds for patient vitals and receive notifications when these thresholds are exceeded.

Stimulus/Response Sequence: When patient vitals exceed preset thresholds, the system sends an alert to the responsible healthcare providers via Email or WhatsApp notifications.

Functional Requirements:

- The system must allow users to customise alert thresholds for each patient.
- Alerts must be sent via multiple channels (Email, WhatsApp) when a threshold is crossed.
- The system must log all alert events for future reference.

Main Flow:

- The user logs into the system and is welcomed by the main console.
- To admit a patient to the ICU, the user retrieves the patient's information from the database.
- The patient is then assigned a bed number in the ICU.
- The user has the option to set customizable thresholds for the patient's vitals.
- Once the patient's vitals are being monitored, if any vital crosses the set thresholds, the centralised system generates an alert.
- The alert is sent to the assigned doctor via email and WhatsApp, containing a link.
- The doctor clicks on the link, which redirects them to the patient's real-time vital screen for further monitoring and action.

Exceptional Flow:

- The user logs into the system and is welcomed by the main console.
- The user attempts to retrieve the patient's information from the database, but if the patient is not found, the system displays an error message.
- If an invalid or duplicate bed number is assigned, the system displays an error message.
- If customizable thresholds are not set and a critical vital is crossed, the system uses default thresholds and still generates an alert.
- In case of a communication failure (email or WhatsApp), the system logs the error and displays: "Alert delivery failed. Please check the communication settings." The system will retry sending the alert.

2.2.3 Non Functional Requirements

2.2.3.1 Performance Requirements

- The system must process and display patient vitals with a maximum latency of 1 second. This ensures that healthcare providers have immediate access to up-to-date information, which is critical in an ICU setting.
- The system must support monitoring up to 10 patients simultaneously. As patient load increases, the system should scale seamlessly without degradation in performance. It should be designed to accommodate additional ICU monitors and patients as needed.
- Historical patient data should be retrievable to allow healthcare providers quick access to records for analysis and decision-making.

2.2.3.2 Safety Requirements

- The system must accurately display real-time patient vitals such as heart rate, ECG, SpO2, blood pressure, and respiratory rate. Any errors in data capture, processing, or display must be flagged immediately, and appropriate notifications should be sent to healthcare providers. The accuracy of these vitals is critical to avoid misdiagnosis or delayed treatment, which could harm patients.
- The alert system must ensure that alarms and notifications for abnormal vitals (e.g., critical heart rate, low oxygen levels) are delivered promptly and reliably to the right healthcare personnel. Missed or delayed alerts can result in severe harm or even patient fatalities, especially in the ICU setting. The system should also log all alerts for auditing and follow-up purposes to ensure timely responses.
- The system must include backup mechanisms in case of hardware or software failures. For example, if the primary ICU monitor fails or loses connection, the system should alert staff immediately and attempt to switch to a backup system to maintain patient monitoring without interruption.

2.2.3.3 Security Requirements

- The system must implement Role-Based Access Control (RBAC) to restrict access to patient data based on user roles (doctors, nurses, administrators). Each user role must have defined permissions to access specific data and functionalities, ensuring that sensitive information is only accessible to authorised personnel.

- The system must maintain comprehensive audit logs of all access events, modifications, and alerts. These logs should be securely stored and easily retrievable for compliance audits and to investigate any suspicious activities.

2.2.3.4 Software Quality Attributes

- **Adaptability:** The system must be adaptable to changes in medical regulations and evolving hospital needs, such as adding new vital monitoring devices or integrating with new hospital management systems. The architecture should support the addition of new machine learning models or predictive algorithms as healthcare analytics evolve.
- **Availability**: The system must be designed for high availability, ensuring that it is operational and accessible all of the time. This includes mechanisms such as load balancing and failover processes to minimise downtime.
- **Scalability**: The system must support an increasing number of patients and ICU monitors without compromising performance. It should be designed to scale horizontally (adding more machines) and vertically (adding more power to existing machines) to meet future growth demands.
- **Correctness**: The system must provide accurate data monitoring and reporting. All calculations and data processing should be validated to ensure that the information presented to healthcare providers is reliable, and supports effective decision-making.
- **Reliability:** The system must ensure reliable performance, with real-time patient vitals consistently monitored and recorded without interruptions or errors. Redundancy mechanisms (e.g., data replication across cloud servers) should be in place to ensure that no data is lost during outages or failures. Regular system health checks and error detection mechanisms must be implemented to ensure reliable operation in high-pressure environments like ICUs.
- **Usability Requirements:** The system must provide an intuitive and user-friendly interface, minimising the need for extensive training. The layout should facilitate quick navigation to critical functions, such as accessing patient data, configuring alerts, and reviewing analytics. The web-based application should be accessible on various devices, including desktops, tablets, and smartphones. The design must accommodate users with different accessibility needs, ensuring compliance with accessibility standards.

2.2.3.5 Business Rules

- Only authorised users (sys admin, doctor, nurse) can log into the system.
- System admin and doctor have full access to all services: vitals monitoring, patient and doctor data, analytics, and alert management.
- Nurses can only access the vitals monitoring and alert services.
- Only the system admin and doctor can retrieve, update, or modify patient and doctor data from the database.
- Doctors can customise vital thresholds for patients; nurses can only view alerts but cannot set thresholds.
- Analytics reports for patient data are accessible only to system admin and doctors.
- Alerts are triggered when a patient's vitals cross the predefined threshold and sent to the doctor via email and WhatsApp.

2.2.4 Development Environment

2.2.4.1 Programming Languages and Frameworks:

- Frontend: HTML5, CSS3(using frameworks like Bootstrap), and JavaScript (using frameworks like React) for creating a responsive web interface [9].
- Backend: Node.js and Express.Js for server-side development [10].

2.2.4.2 Database Management System (DBMS)

A cloud-based relational database management system (MySQL) to

securely store and manage patient data.

2.2.4.3 Machine Learning Tools

Libraries such as Scikitlearn for data analysis and predictive modelling. Libraries like Seaborn and Pyplot will be utilized for visualization when necessary [11]. Algorithms such as Support Vector Machine (SVM)[12] and K-Nearest Neighbors (KNN)[13] will be employed for predictive analysis.

2.2.5 Deployment Environment

2.2.5.1 Web Server

A web server (e.g., Apache, Nginx, or Microsoft IIS) to serve web pages to client devices [14].

2.2.5.2 Cloud Infrastructure:

The following cloud technologies will be implemented in our project [15]:

- Security:- VPC will be used to ensure security and privacy.
- Hosting:- ECS & EC2 instances for hosting an application on the AWS Cloud
- Database:- RDS instance for storing data in SQL Database
- Storage:- S3 bucket for storing Application & Patient Files
- Analytics: Data Analytics will be used to identify patterns.
- Alerts: Triggers & Alerts will be used for Alerts generation.
- Sensor Data: IOT Cloud & Core will be used to process IOT Endpoint Data.

2.2.6 External Interface Environment

2.2.6.1 User Interface

The User Interface (UI) of the HealthWatch system is designed to provide an intuitive, user-friendly experience for healthcare providers. The following describes the logical characteristics of the UI:

1. **Main Console**:
- After login, all users are welcomed by the Main Console, a dashboard that provides a high-level overview of system options based on their roles.
- The Main Console contains tabs or navigation buttons for the available services: Vitals Monitoring, Patient and Doctor Registry, Analytics, and Alert Management.
2. **Vitals Monitoring Dashboard**:
- Displays real-time patient vitals in a centralised format, with data such as heart rate, ECG, blood pressure, SpO2, and respiratory rate.
- The user can click on individual bed IDs to view detailed patient data on a separate screen.
- Customizable layouts allow doctors and sys admins to prioritise which vitals are displayed prominently.
3. **Patient and Doctor Data Interface**:

- Accessible only to sys admins and doctors, this page allows users to retrieve, add, modify, and view patient and doctor records.
- Information is displayed in a clean tabular format, with searchable fields to quickly locate patient or doctor profiles.
- The interface includes form fields for adding or updating information, with dropdowns for selecting bed assignments and ICU status.

4. **Analytics Dashboard**:

- Provides an overview of patient data trends for the last 30 days, accessible to system admins and doctors.
- Displays graphical representations such as line charts, bar graphs, and tables showing trends in vitals, alert frequencies, and patient outcomes.
- Users can drill down into specific data by clicking on individual patient IDs for more detailed analytics.

5. **Alert Management Interface**:

- Available to doctors and nurses for managing patient alerts.
- Displays all active alerts in a list format, with details about the patient, the vital that triggered the alert, and the time of occurrence.

2.2.6.2 Hardware Interface

1. **ICU Monitors (Draeger)**:

- Data from ICU monitors will be extracted through an RS-232 port provided at each monitoring system. The data can be analog (e.g., ECG) or digital (e.g., pulse), depending on the type of sensor used [16].
- The analog data will be converted into digital values using the ESP32 microcontroller, which has a built-in Analog-to-Digital Converter (ADC).

2. **Microcontroller Interface**:

- The ESP32 microcontroller will act as a filter, collecting data from the ICU monitors and transmitting it to the master system via wired (cable) or wireless (Wi-Fi) connections [17].
- The data will be sent from the microcontrollers to the master system through a brute force method, where multiple microcontrollers can report to the master system concurrently.

3. **Master System**:

- The master system will receive filtered data from multiple ICU monitors, aggregating and processing it before passing it to the cloud-based system.
- Node-RED will act as the middleware, enabling communication between the hardware components (ICU monitors and microcontrollers) and the web-based system [18].

2.2.6.3 Software Interface

1. **Operating Systems**:

- **Server-Side**: Linux (Ubuntu or CentOS) or Windows Server to host the master system and manage cloud operations.
- **Client-Side**: Windows, macOS, and mobile operating systems (iOS, Android) to access the web-based application.

2. **Database Management System**:

- The system will use a cloud-based relational database (MySQL) to store patient data securely.
- Data will include patient vitals, alerts, and historical records.
- The database must support querying for real-time data as well as historical trend analysis.

3. **Middleware (Node-RED)**:

- Node-RED will serve as the intermediary between the hardware (ESP32 microcontrollers) and the web application.
- It will handle the collection, transformation, and forwarding of ICU monitor data to the web-based system for real-time display.

4. **Web Application**:

- The web-based interface will be built using JavaScript frameworks such as React along with HTML5 and CSS3 for a responsive design.
- Communication between the backend and frontend will be handled through RESTful APIs, facilitating data retrieval, alert configuration, and user management.

2.3 Project Process Model

The **Spiral Model** is particularly well-suited for our project HealthWatch, where the development of a healthcare monitoring system requires a balance between innovation, security, and compliance with industry standards. Given the complexity of integrating real-time patient data from the Dräger Infinity Delta monitoring system (such as pulse, SpO2,

respiration, and other vital signs), this model ensures a methodical approach that can address risks at every phase of development while being flexible to incorporate new requirements or technologies [19].

2.3.1 Planning Phase (Objective Setting and Requirement Analysis)

In the initial phase, our team would engage in thorough planning, where objectives are clearly defined, and all system requirements are gathered. For HealthWatch, these objectives would likely include:

- **Accurate and secure data collection**: Ensuring that data from multiple monitoring devices (pulse, SpO2, blood pressure) is transmitted reliably and without loss.
- **Cloud data storage and real-time access**: Data must be stored securely in the cloud and made available in real-time for healthcare providers to access critical patient information when needed.
- **System performance and scalability**: Considering that the system may need to handle high volumes of data, scalability and performance benchmarks must be clearly defined.

During this phase, we'd also map out project constraints such as time, budget, and resource limitations, as well as technical challenges like latency, hardware compatibility, and integration with existing hospital systems.

2.3.2 Risk Analysis Phase (Identifying and Mitigating Risks)

Once the planning is complete, the next step is identifying potential risks that could hinder the project's success. In HealthWatch, risks could include:

- **Data security**: Given the sensitivity of patient data, breaches or unauthorized access are significant risks.
- **System reliability**: A failure in the system could lead to inaccurate or missing patient data, which can have critical consequences in healthcare. Risk mitigation might include redundant data paths, robust error detection, and recovery mechanisms.
- **Interoperability issues**: Since the system must integrate with existing monitoring hardware (Dräger Infinity Delta), ensuring seamless hardware-software communication is

vital. If there are compatibility issues, the project could be delayed or result in an unusable product.

- **Performance under load**: The system may need to handle a high influx of data from multiple patients simultaneously, especially in larger healthcare facilities. If not properly managed, this could lead to data bottlenecks, system slowdowns, or even crashes.

For each identified risk, mitigation strategies are developed. Testing scenarios would be built around these risks to assess how the system performs under stress conditions.

2.3.3 Engineering Phase (Development and Testing)

During this phase, the actual development of HealthWatch begins. Based on the outcomes of the risk analysis, the team would start building system components in an incremental manner, which allows for testing and validation at each step. For example:

- **Prototyping a single data stream**: In early iterations, you may choose to focus on building the functionality for one type of data (e.g., pulse monitoring). This allows the team to assess how well the system collects, transmits, and stores pulse data in the cloud before adding other components like SpO2 or blood pressure monitoring.
- **Real-time performance checks**: Given the need for real-time data analysis, testing the system's performance under simulated load conditions is crucial. The system would be tested to ensure data from multiple patients can be transmitted, processed, and made available in real-time without delays.

Each increment focuses on a part of the system, so risks are addressed early, and feedback is incorporated. This phase may involve setting up a small-scale environment within a healthcare institution for live testing, allowing the team to observe how well the system performs under actual use conditions.

2.3.4 Evaluation Phase (Customer Feedback)

At the end of each spiral iteration, the system will undergo evaluation by the developers, including healthcare professionals. This feedback helps assess whether the system meets the healthcare providers' real-world needs and ensures it is compliant with regulatory requirements.

For HealthWatch, healthcare professionals might provide feedback on:

- **User interface**: Is the interface intuitive and user-friendly for quick access to patient data?
- **Data accuracy**: Are there any anomalies or delays in how patient data is presented in real-time?
- **System responsiveness**: Can the system handle the real-time nature of the monitoring without slowdowns or glitches?

Based on this feedback, any changes, improvements, or additional features can be integrated into the next development iteration. For instance, if the interface for data visualization is too complex for the concerned authority, the team would simplify it in the next iteration, making it easier for users to access critical patient information.

2.3.5 Continuous Iteration

One of the strengths of the Spiral Model is its **cyclic and iterative nature**. Each loop of the spiral represents a refinement of the system, allowing continuous improvement, risk reduction, and the ability to adapt to new requirements. This is particularly useful for a project like HealthWatch, where unforeseen challenges (such as changes in medical device standards) can arise, and features like machine learning for predictive health monitoring can be added.

By progressively refining each aspect of the project through these phases, the final product will be a well-tested, robust, and compliant system capable of handling real-time healthcare data and meeting the specific needs of healthcare providers.

2.4 Project Scheduling

2.4.1 Timeline Chart

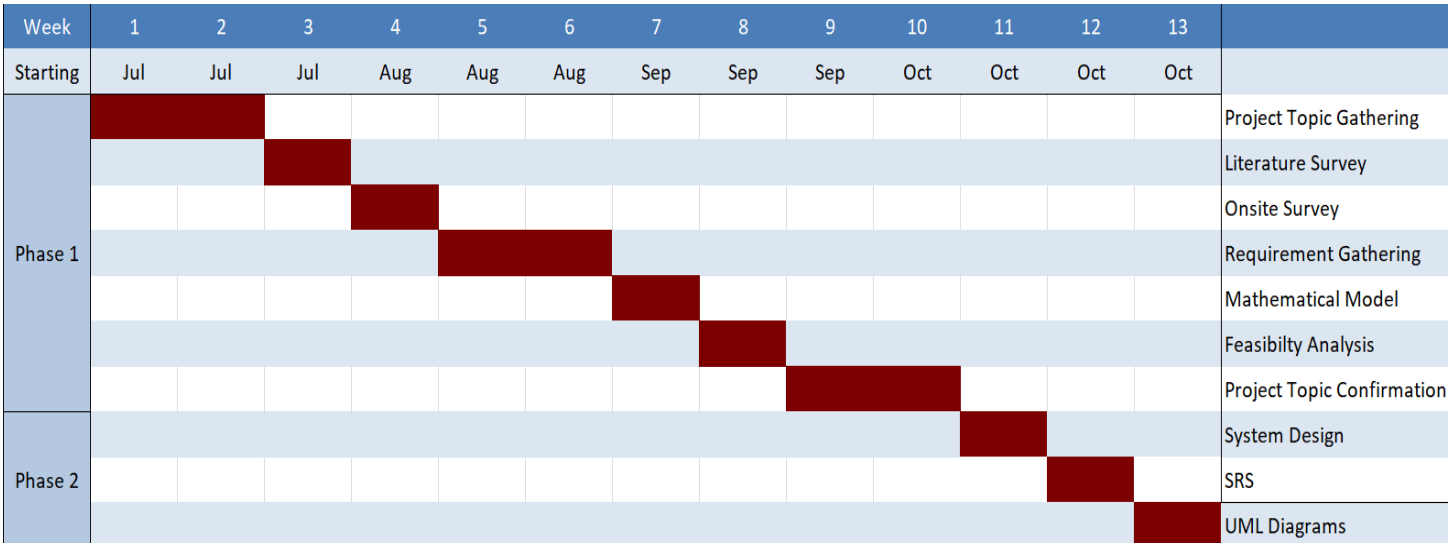

Figure 2.5: Timeline Chart

Chapter 3

ANALYSIS AND DESIGN

3.1 Introduction

This chapter covers the analysis and design of the considered system

3.2 IDEA Matrix

I	**Use**	**Parameters Affected**
Improve	Improve hospital efficiency	Productivity
Increase	Increased data storage time.	Data Accessibility
Innovation	The features introduced are innovative and not present in any of the existing Healthcare Systems.	Add-on Functionality

Table 3.1: I Elements

D	**Use**	**Parameters Affected**
Detect	Detect vital abnormalities	Memory Utilization
Decrease	Decrease the unpredictable risks.	Data Accessibility
Deliver	Deliver the urgent response to emergency situations.	Emergency Handling

Table 3.2: D Elements

E	Use	Parameters Affected
Enhance	Enhance the communication between doctor and patient	Effective Communication
Extract	Extract useful information from Data Analytics	Pattern Identification
Evaluate	Evaluate patients based on their medical history.	Patient Information

Table 3.3: E Elements

A	Use	Parameters Affected
Avoid	Avoid emergency situations	Reducing hurly burly
Assurance	Assurance to patients that continuous monitoring is present.	Patient De-stress
Associate	The developed application will be able to associate to previous healthcare system	Adaptability

Table 3.4: A Elements

3.3 Mathematical Model

Let S be the proposed system.

S = { Input, Functions, Output, Success, Failure }

Where,

Input = {I1, I2, I3, I4, I5}

Where,

I1 = Sensor data (pulse, SpO2, blood pressure, respiration rate, etc.)

I2 = Real-time monitoring alerts

I3 = Historical health data from centralised cloud storage

I4 = User input for trigger customization (doctor/nurse/concerned authority setting alert

conditions)
I5 = External system data (Draeger Infinity Delta monitoring system, patient records)

Functions = {F1, F2, F3, F4, F5}

Where,
F1 = Function to collect and transmit real-time sensor data to cloud storage
F2 = Function to trigger alerts based on custom-set thresholds
F3 = Function to analyze historical patient data to identify patterns (ML model for predicting patient deterioration)
F4 = Function to centralise and standardise the data from multiple devices
F5 = Function to provide real-time access to patient vitals on the web interface

Output = {O1, O2, O3, O4, O5, O6}

Where,
O1 = Accurate and real-time display of patient vitals on web interface
O2 = Trigger alerts for healthcare providers based on patient conditions
O3 = Long-term health patterns identified for individual patients
O4 = Remote monitoring capabilities enabled for doctors outside the hospital
O5 = Data stored securely in cloud storage for future analysis
O6 = Actionable insights for medical teams for prompt intervention

Success = {S1, S2, S3, S4}

Where,
S1 = Successful and real-time data transmission from ICU devices to cloud
S2 = Alerts generated successfully based on set thresholds
S3 = Identified patterns in historical data for predicting patient deterioration
S4 = Data successfully accessible remotely in real-time

Failure = {F1, F2, F3}

Where,
F1 = Failure in transmitting real-time data due to network/server issues
F2 = Inaccurate or missed alerts causing delayed medical responses
F3 = Failure in data synchronization between hospital devices and the centralised system.

3.4 Feasibility Analysis

- Data Collection and Transmission (F1: Function to collect and transmit real-time sensor data to cloud storage): **Time Complexity: O(1)**
- Alert Triggering (F2: Function to trigger alerts based on custom-set thresholds): **Time Complexity: O(n)**
- Analyzing Historical Data (F3: Function to analyse historical patient data to identify patterns): **Time Complexity: O(n*m)**
- Data Centralization (F4: Function to centralise and standardise data from multiple devices): **Time Complexity: O(k)**, where **k** is the number of device types, and it's generally constant.
- Real-time Access (F5: Function to provide real-time access to patient vitals on the web interface): **Time Complexity: O(1)**

The overall time complexity for the proposed system can be approximated as the sum of the complexities for each function:

$T(n) = O(1) + O(n) + O(n*m) + O(k) + O(1)$

Since O(n * m) dominates the other terms (especially as the number of historical records increases), the total time complexity of the system is approximately:

$T(n) = O(n * m)$

Where:

- n is the number of data records in the historical dataset.
- m is the complexity of the machine learning model used to analyze historical data.

System operates with a time complexity of O(n * m), which means that the most computationally expensive part of the system is the analysis of historical data for predicting patient deterioration. However, given the critical nature of ICU monitoring, this is a reasonable time complexity, ensuring that the system remains scalable as the number of patients and data records grows.

3.5 Architecture Diagram

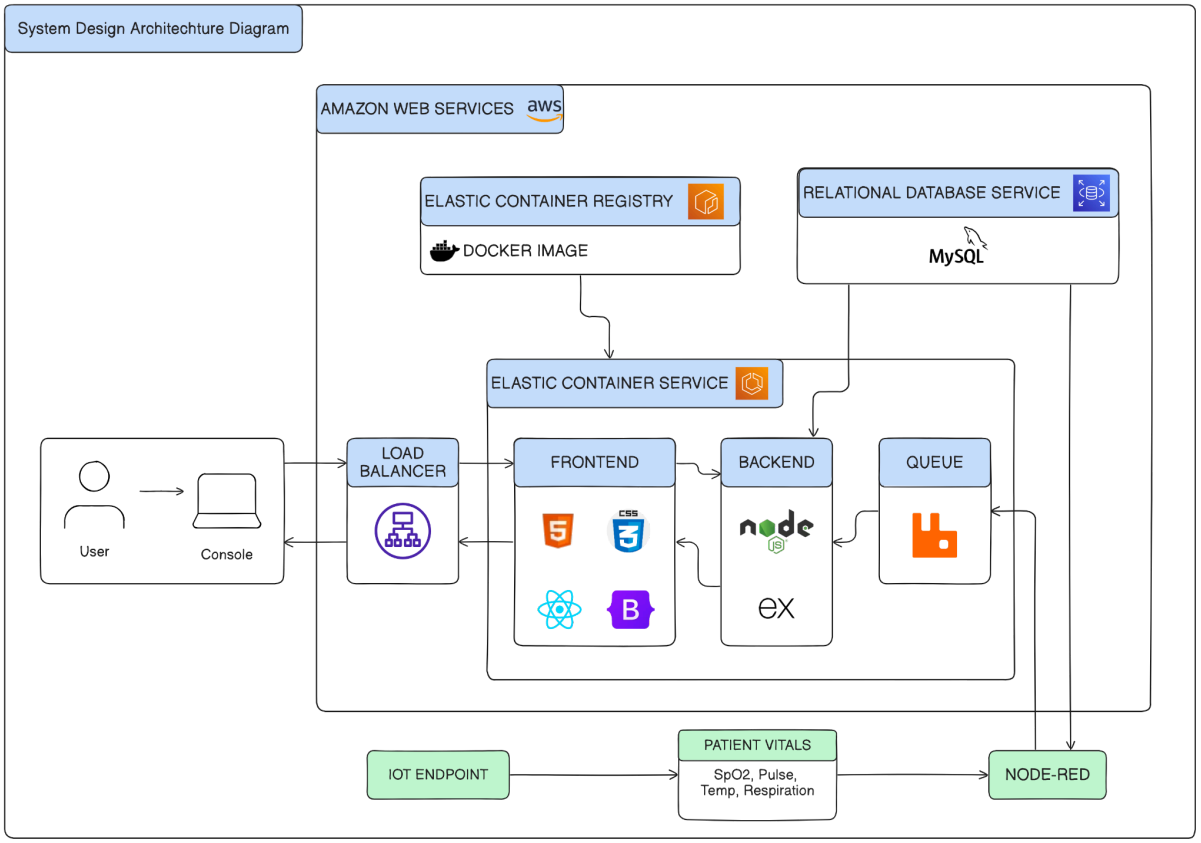

Figure 3.1: Architecture Diagram

3.6 UML Diagrams

3.6.1 Methodology Diagram

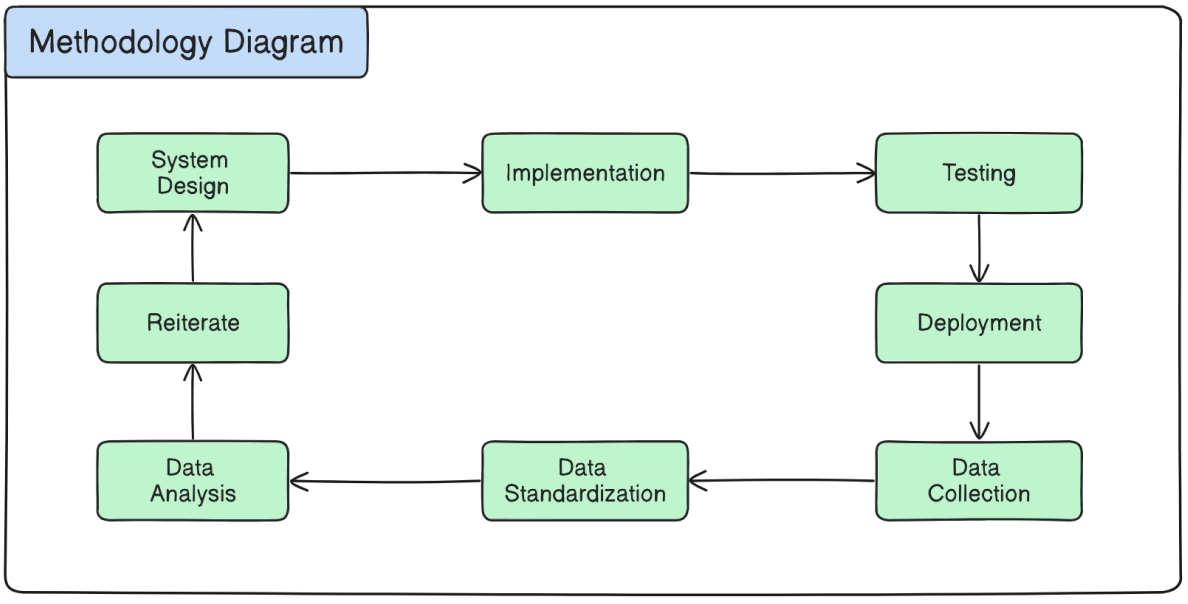

Figure 3.2: Methodology Diagram

3.6.2 Use-Case Diagram

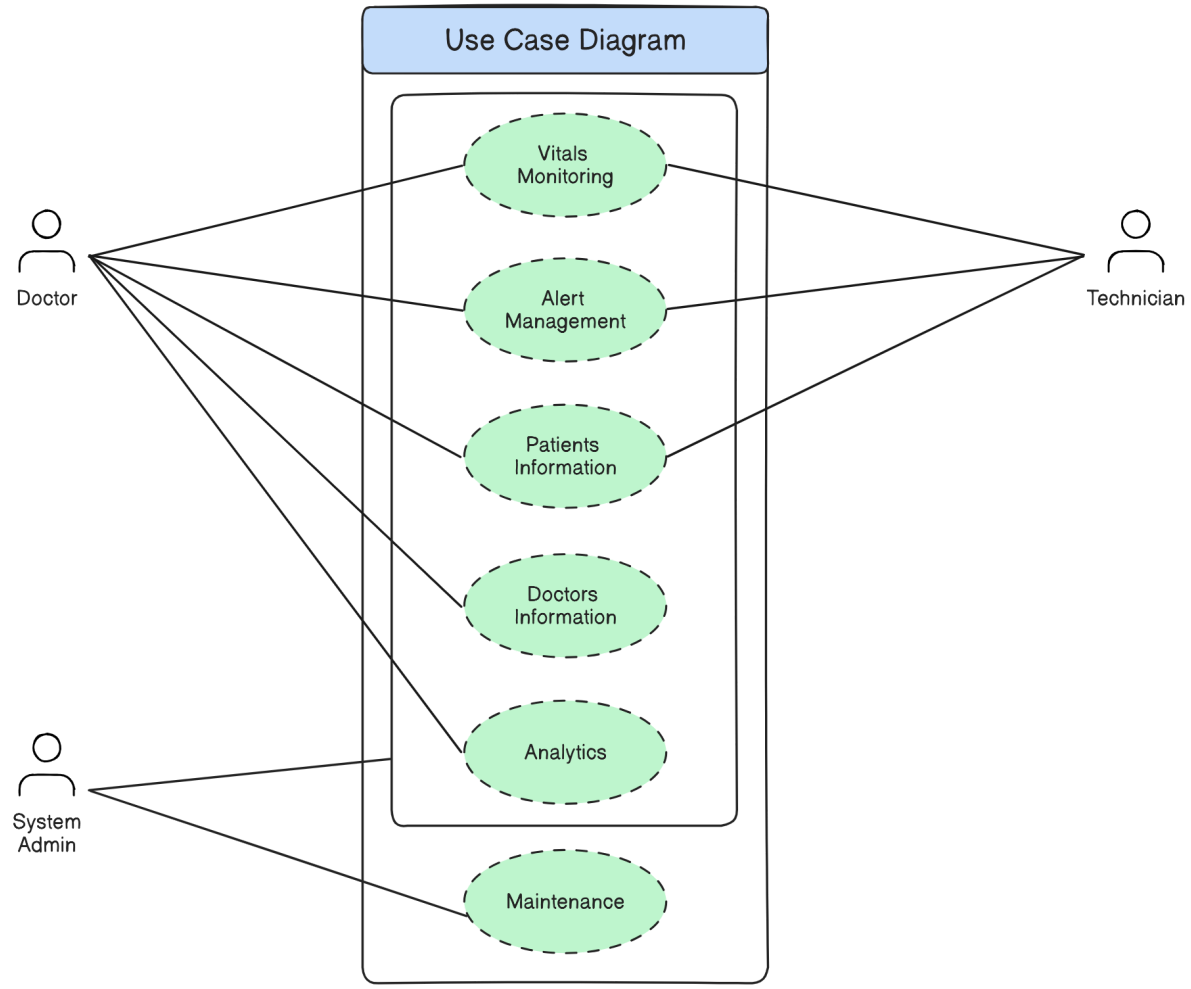

Figure 3.3: Use Case Diagram

The Use-Case diagram represents these relationships:

1. The Doctor interacts with the system to Monitor Vital Signs and receive alerts.
2. The Doctor and Nurse access the system for Accessing Patient History and respond to Alert Generation.
3. The System Administrator interacts with the system for Updating Information and ensuring Data Security.

3.6.3 Activity Diagram

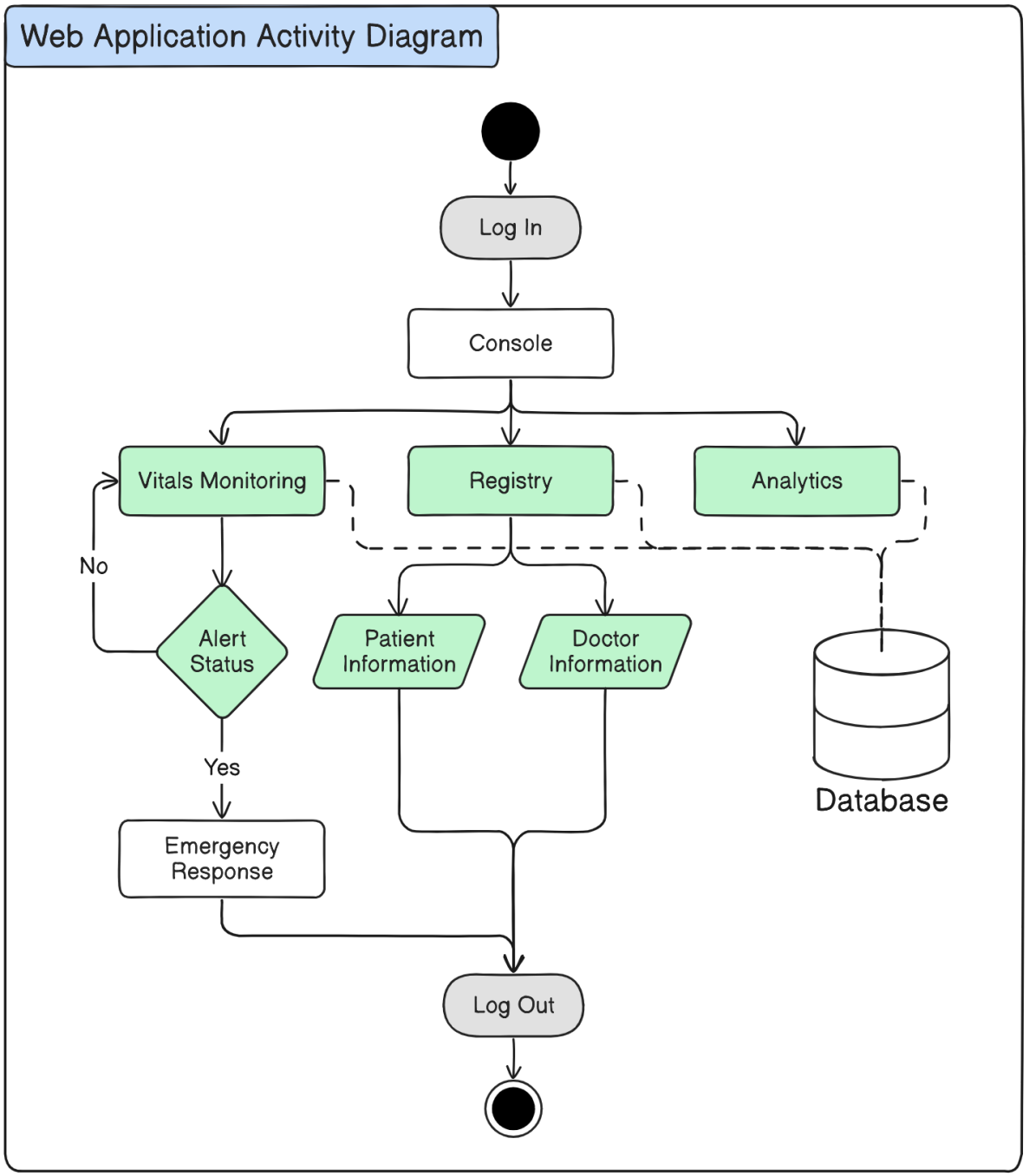

Figure 3.4: Activity Diagram

3.6.4 Sequence Diagram

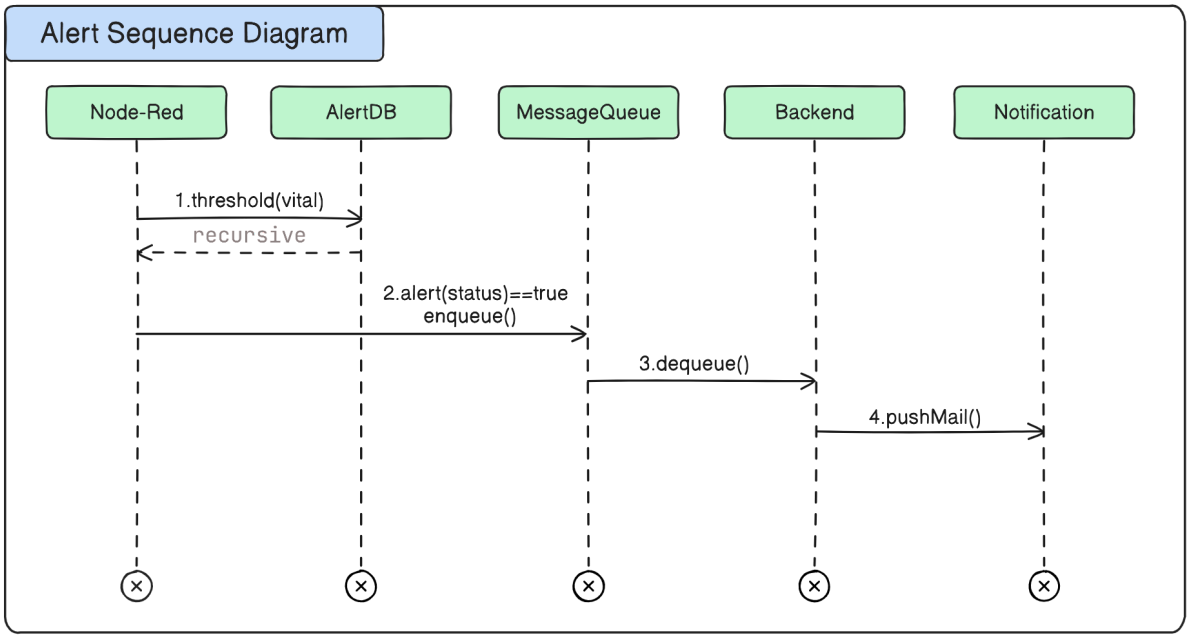

Figure 3.5: Alert Sequence Diagram

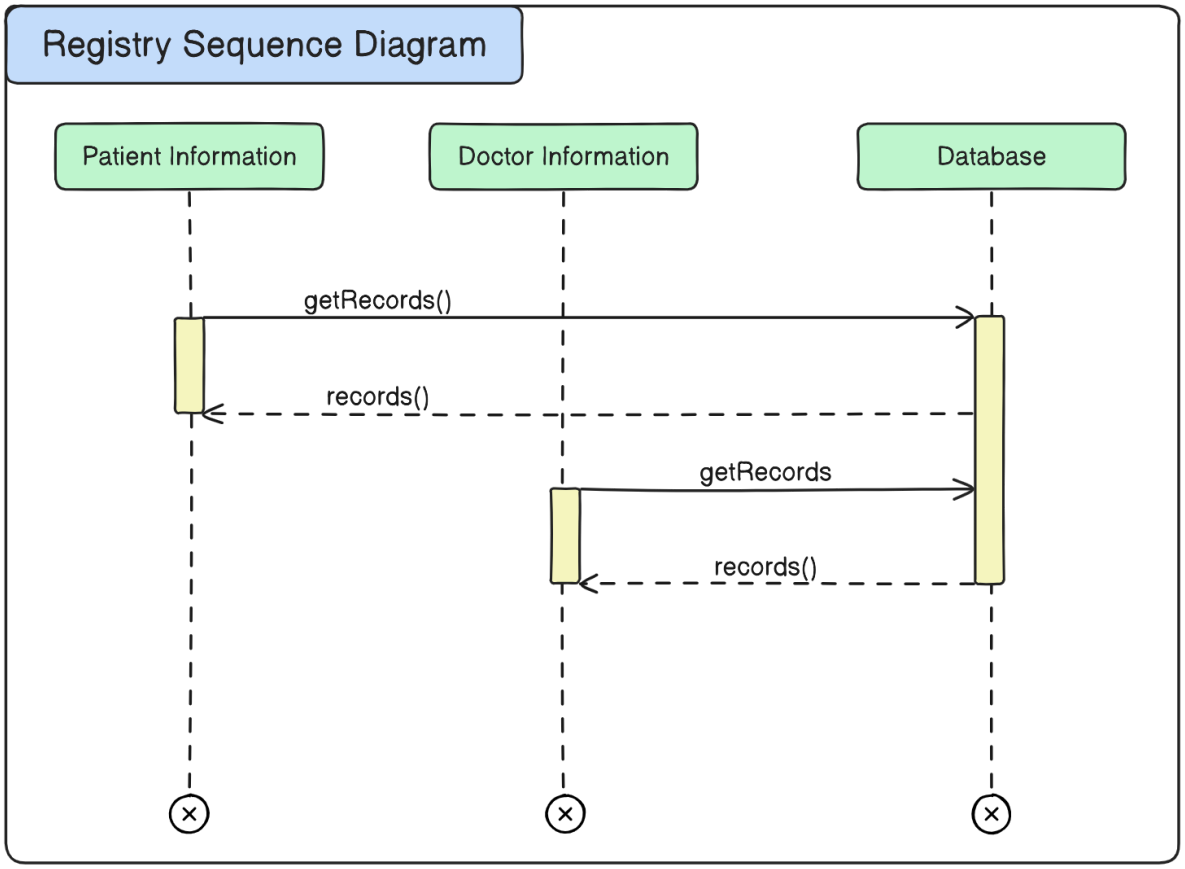

Figure 3.6: Registry Sequence Diagram

3.6.5 Class Diagram

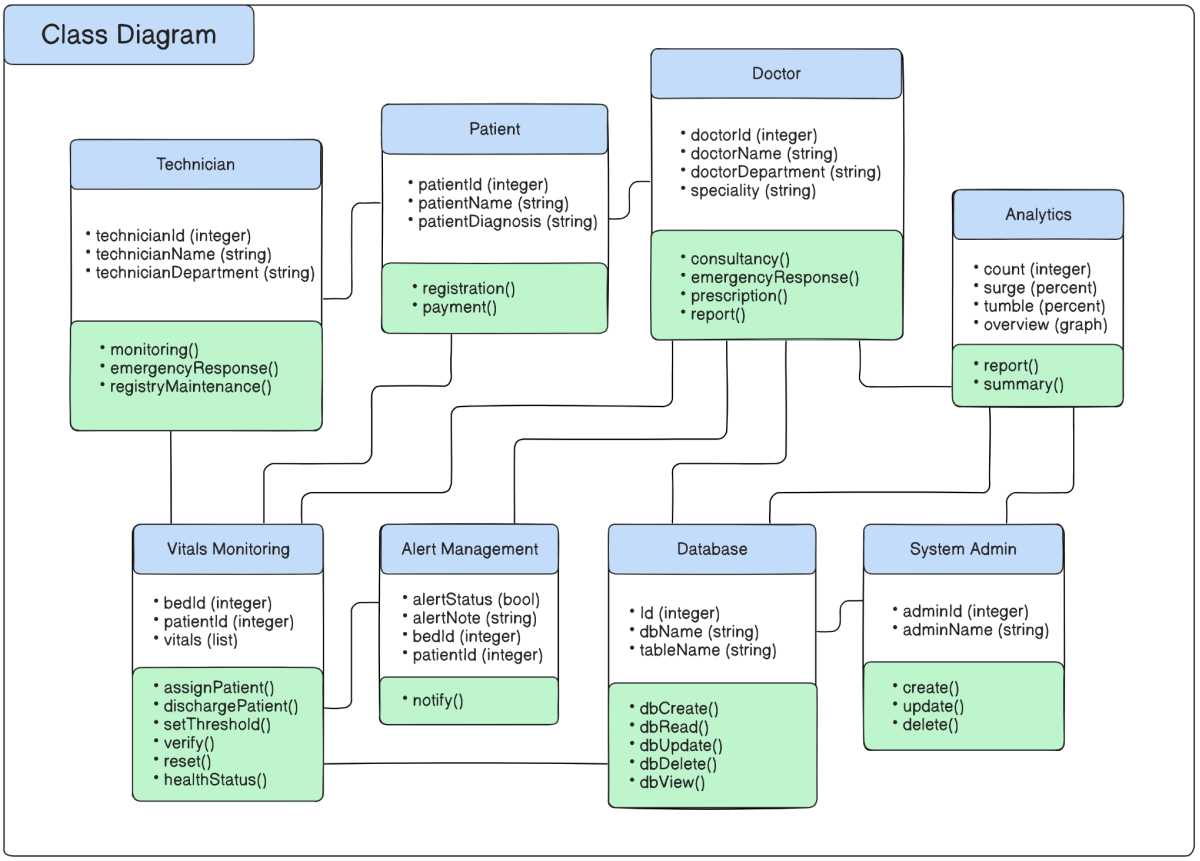

Figure 3.7: Class Diagram

3.6.6 ER Diagram

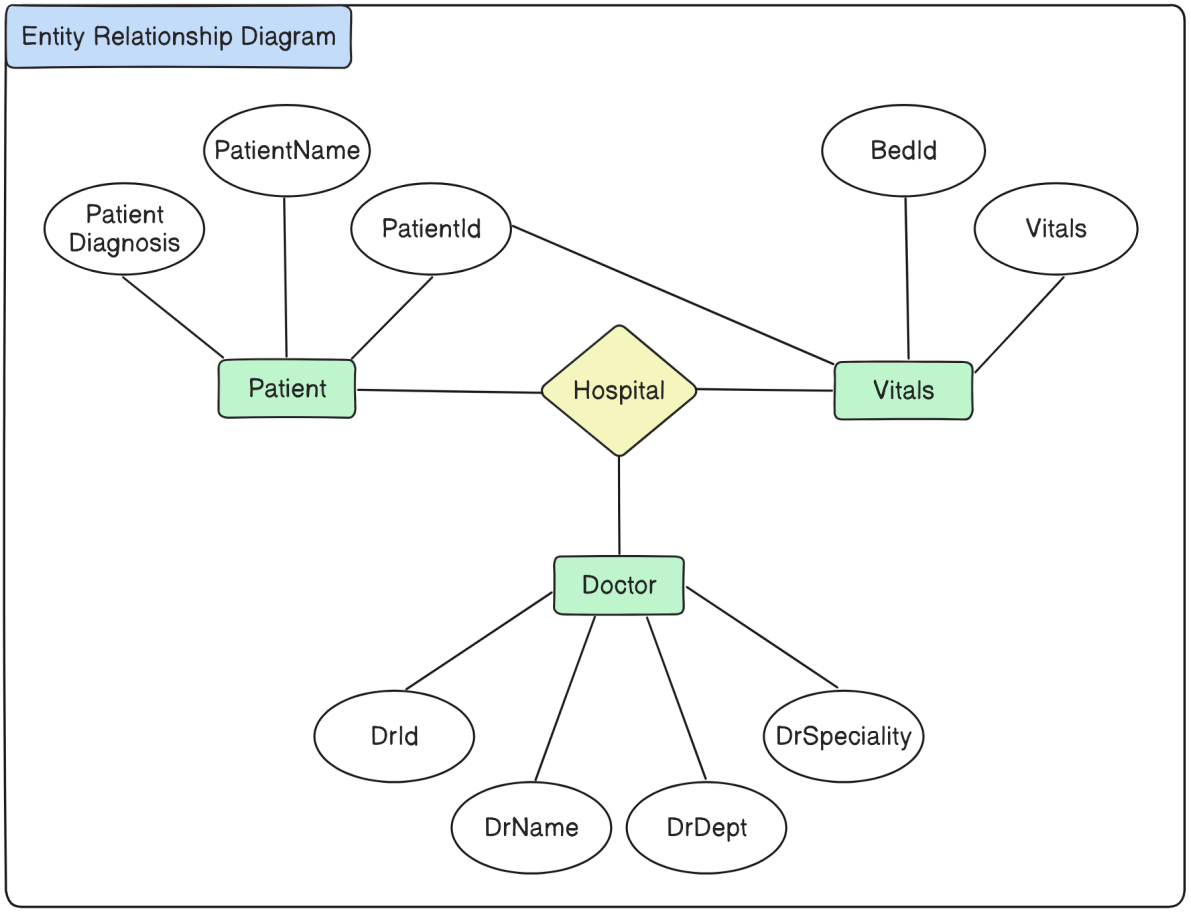

Figure 3.8: ER Diagram

Chapter 4
TESTING

4.1 Introduction

This chapter covers the testing approach used and the test cases [20].

4.2 Functional Testing

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements.

Sr. No.	Module	Description	Expected Result
1	Real-Time Patient Monitoring	Verify that the system collects patient vitals from ICU monitors every second.	Vitals such as ECG, heart rate, blood pressure, and SpO2 should update every second on the dashboard.
		Test the centralised dashboard displaying all patient vitals in real-time.	All patient vitals should be visible, and the dashboard should update automatically without manual refresh.
		Simulate a loss of connection to the ICU monitor.	The system should display an error message indicating the loss of data connection.
2	Data Storage	Verify that patient data is stored in the cloud for continuity.	Data stored for each patient should be accessible throughout time.

		Simulate a server connection failure.	The system should store data locally and sync with the cloud once the connection is restored.
3	Data Analytics and Predictive Insights	Test the system's ability to apply machine learning algorithms to analyse data.	The system should generate predictive insights and detect trends in patient data.
		Verify that the system generates alerts when the ML model predicts potential health risks.	Alerts should be raised based on predictive analysis, and trends should be displayed via visual graphs.
4	Customizable Alerts and Notifications	Test the system’s ability to allow users to customize alert thresholds for different patients.	The system should save custom alert thresholds and notify when thresholds are exceeded.
		Verify that alerts are sent via multiple channels (email, SMS, and in-app notifications).	Alerts should be received on the configured channels without delay.
5	Access Control and Security	Test role-based access control by attempting access with different user roles.	Users should only be able to access data and perform actions according to their assigned roles.
		Verify that unauthorized access attempts are logged.	Any attempt to access the system by unauthorized users should be logged and reported to IT administrators.

Table 4.1: Functional Testing Table

4.3 Non-Functional Testing

Non functional testing is a type of software testing that verifies non functional aspects of the product, such as performance, stability, and usability.

1	Performance Testing	Measure system performance with 10 patients being monitored simultaneously.	The system should handle all connections without performance degradation.
		Test the latency of patient vitals being updated on the dashboard.	Data should be displayed with a maximum latency of 1 second.
		Verify that historical patient data is retrieved within 2 seconds.	Healthcare providers should be able to access historical data promptly.
2	Security Testing	Test the encryption of data during transmission.	All data transferred between the system, cloud, and client devices should be encrypted using SSL/TLS protocols.
		Verify that all access events are logged.	The system should maintain audit logs for every access and modification event.
3	Usability Testing	Test the usability of the system across different devices (desktops, tablets, smartphones).	The interface should be responsive and easy to use on all devices.
		Evaluate accessibility for users with disabilities.	The system should comply with accessibility standards

			and be usable by users with different needs.

Table 4.2 Non-Functional Testing Table

4.4 System Scalability and Availability Testing

Scalability testing is a type of performance testing that evaluates how a system, application, or network can scale up or down to meet increased demands. Availability testing, also known as high availability (HA), measures how well an application remains available when one or more underlying components fail.

1	Scalability Testing	Stress tests the system by adding additional ICU monitors and patients.	The system should scale horizontally and vertically to accommodate increased data without performance loss.
2	Availability Testing	Test failover mechanisms by simulating a web server failure.	The system should continue functioning with minimal downtime by activating failover processes.

Table 4.3: System Scalability and Availability Testing Table

4.5 Integration and Interface Testing

Integration testing, also known as I&T, is a software testing method that verifies how different components of a software application work together.

1	External Interface Testing	Test secure communication between ICU monitors and the system.	Data from monitors should be securely transmitted using RS-232 and ESP32 without data loss.

		Test the integration of external notification systems	SMS and email notifications should be sent reliably via the integrated services.
2	Cloud Integration Testing	Test the interaction between the web application and AWS cloud services (EC2, RDS, S3).	All cloud-based services should function correctly, and data should be stored and retrieved seamlessly.

Table 4.4: Integration and Interface Testing Table

Chapter 5
CONCLUSION

This web application offers robust data management, event management, device compatibility, and storage solutions, addressing several challenges in the medical field, such as the incompatibility of various medical devices and data monitoring and storage issues. However, it does come with challenges, including the need for skilled technicians to set it up in hospitals. Changes in medical device communication protocols could lead to temporary system failures, potentially creating liability concerns. Continuous processing and updates are essential for its proper functioning. Additionally, patient data security could be compromised if the server is hacked, as hospitals often do not prioritise server management and protection.

Furthermore, this system will create a future-ready environment by integrating various databases and cloud systems, enabling advanced analysis for patient diagnoses using AI and ML algorithms. This will assist in identifying early signs of serious or complex diseases and help doctors recognize past complications that could lead to new issues. It can also track the effects of different treatments and medications on patients, preparing doctors in advance and facilitating comparisons of the effectiveness of various treatments. Despite its advantages and disadvantages, we believe this project will be highly beneficial to our society.

References

[1] B. Joel, S. Sibi Rajan, R. Vibinanth, D. Pamela, and P. Manimegalai, “I-Doc – A Cloud Based Data Management System For Health Care,” *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, Apr. 2022, doi: https://doi.org/10.1109/icdcs54290.2022.9780811.

[2] D. Akhil, M. V. Vardhan, K. V. Reddy, C. Preetham and N. Sangeeta, "Iot Based Icu Patient Monitoring Smart System," *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)*, vol. 13, pp. 1–6, May 2024, doi: https://doi.org/10.1109/aiiot58432.2024.10574787.

[3] M. D. Aldhoayan and Y. Aljubran, “Prediction of ICU Patients’ Deterioration Using Machine Learning Techniques,” Cureus, May 2023, doi: https://doi.org/10.7759/cureus.38659.

[4] C. V. Ravikanth, Kolangiammal, K. Chegondi, and V. Gowtham, “Cardio Monitoring and Cardiac disease prediction using Machine Learning,” *2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI),* pp. 1–6, Apr. 2023, doi: https://doi.org/10.1109/raeeucci57140.2023.10134081.

[5] S. C, V. R, K. Sreelatha and S. V, "SIPMS: IoT based Smart ICU Patient Monitoring System," 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-7, doi: https://doi.org/10.1109/iceconf57129.2023.10083603.

[6] A. Ahmed, M. T. Mahmud, and M. M. Khan, "Info Hospital: Web/Mobile Application based Health Care System," *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, 2022, pp. 1546–1552, doi: 10.1109/ICCMC53470.2022.9753895. Available: https://ieeexplore.ieee.org/document/9753895.

[7] R. M. H, S. Udupa, S. A. Bhagavath, Shreesha, and V. Rao, “Centralised and Automated Healthcare Systems: A Essential Smart Application Post Covid-19,” IEEE Xplore, Dec. 01, 2020. Available: https://ieeexplore.ieee.org/document/9316103.

[8] “Front-end web development.” Wikipedia. https://en.wikipedia.org/wiki/Front-end_web_development.

[9] “Back-End Architecture.” Codecadem y. https://www.codecademy.com/article/back-end-architecture#:~:text=The%20back%2Dend%20is%20the,the%20data%20for%20the%20application.

[10] “Data Visualization in Python.” Simplilearn. https://www.simplilearn.com/tutorials/python-tutorial/data-visualization-in-python.

[11] “Support Vector Machine Algorithm.” GeeksforGeeks. https://www.geeksforgeeks.org/support-vector-machine-algorithm/.

[12] “K-Nearest Neighbours.” GeeksforGeeks. https://www.geeksforgeeks.org/k-nearest-neighbours/.

[13] “Web Server and its Type.” GeeksforGeeks. https://www.geeksforgeeks.org/web-server-and-its-type/.

[14] “Cloud Computing.” Wikipedia. https://en.wikipedia.org/wiki/Cloud_computing.

[15] “Infinity Delta Series.” Draeger. https://www.draeger.com/en_in/Products/Infinity-Delta-Series.

[16] “ESP32.” Espressif Systems. https://www.espressif.com/en/products/socs/esp32.

[17] “Node-RED.” Node-RED. https://nodered.org/.

[18] “Software Engineering | Spiral Model.” GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-spiral-model/ .

[19] “Software Testing Basics.” GeeksforGeeks. https://www.geeksforgeeks.org/software-testing-basics/ .