HOSPICARE – HOSPITAL REQUIREMENT ANALYSER AND COMPLAINTS RESOLVER

A MINI PROJECT REPORT

Submitted by

BALA HARIHARAN B (720722104006)

KANDASAMY E (720722104018)

LINGESH R (720722104022)

SENTAMIL SELVAN P (720722104036)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Hindusthan College of Engineering and Technology

Approved by AICTE, New Delhi, Accredited with 'A++' Grade by NAAC

(An Autonomous Institution, Affiliated to Anna University, Chennai) Valley Campus, Pollachi Highway, Coimbatore – 641 032

DECEMBER 2024

Hindusthan College of Engineering and Technology
Approved by AICTE, New Delhi, Accredited with 'A++' Grade by NAAC
(An Autonomous Institution, Affiliated to Anna University, Chennai) Valley Campus, Pollachi Highway, Coimbatore – 641 032

BONAFIDE CERTIFICATE

Certified that this project report **"HOSPICARE – HOSPITAL REQUIREMENT ANALYSER AND COMPLAINTS RESOLVER"** is the Bonafide work of "**BALA HARIHARAN B(720722104006) KANDASAMY E(720722104018) LINGESH R (720722104022) SENTAMIL SELVAN P(720722104036)**" who carried out the project work under my supervision.

SIGNATURE

MR.M RAVIKUMAR, M.Tech., (Ph.D.)

SUPERVISOR

Assistant professor
Computer Science and Engineering
Hindusthan College of Engineering and Technology, Coimbatore-32

SIGNATURE

Dr.S SHANKAR ,M.E., Ph.D.

HEAD OF THE DEPARTMENT

Computer Science and Engineering
Hindusthan College of Engineering and Technology, Coimbatore-32

Submitted for the Autonomous Institution Mini Project Viva-Voce conducted on

_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We, hereby jointly declare that the project work entitled **"HOSPICARE - HOSPITAL REQUIREMENTS ANALYSER AND COMPLAINTS RESOLVER**", submitted to the Autonomous Institution Mini Project Viva voce-December 2024 in partial fulfilment for the award of the degree of **"BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING"**, is the report of the original project work done by us under the guidance of **Mr. M. RAVIKUMAR, M.TECH., (Ph.D.),** Assistant Professor, Department of Computer Science and Engineering, Hindusthan College of Engineering and Technology, Coimbatore.

NAME	SIGNATURE
1. BALA HARIHARAN B	_____
2. KANDASAMY E	_____
3. LINGESH R	_____
4. SENTAMIL SELVAN P	_____

I certify that the declaration made by the above candidates are true.

Project Guide,

Mr. M. RAVIKUMAR, M.Tech., (Ph.D.),
Assistant Professor,
Department of Computer Science and Engineering,
Hindusthan College of Engineering and Technology, Coimbatore –32

ACKNOWLEDGEMENT

We take this opportunity to express our wholehearted thanks and our profound respect to all those who guided and inspired us in the completion of this project work.

We extend our sincere thanks to the Founder and Chairman of Hindusthan Educational and Charitable Trust **Shri. T.S.R. KHANNAIYANN** and the Managing Trustee **Smt. SARASUWATHI KHANNAIYANN** and Executive Trustee & Secretary **Mrs. PRIYA SATISH PRABHU** for providing essential infrastructure.

We would like to reveal our profound thanks to our respected Principal, **Dr. J. JAYA, MTech, Ph.D.,** who happens to be striving force in all endeavours.

We would like to express our gratitude to the Head of the Department **Dr. S. SHANKAR, M.E., Ph.D.,** for bringing out the project successfully and for strengthening the ray of hope.

We would like to express our sincere thanks and deep sense of gratitude to our Class Advisor and Guide **Mr. M. RAVIKUMAR, M.Tech., (Ph.D.),** Assistant Professor, Department of Computer Science and Engineering, for his valuable guidance, suggestions and constant encouragement which paved way for the successful completion of the mini project work.

We express our immense pleasure and thankfulness to all our department faculty members, technical staffs and friends who helped us for the successful completion of this mini project. We express our gratitude to **Mr. BALAMURUGAN B** and **Mrs. SUMATHI B** parents of BALA HARIHARAN B, **Mr. ELANGOVAN S** and **Mrs. POOMARI E** parents of KANDASAMY E, **Mr. RAJESH KANNA T** and **Mrs. BALAPRIYA R** parents of LINGESH R, **Mr. PALANI N** and **Mrs. SUMATHI P** parents of SENTAMIL SELVAN P and our family members who encouraged us and strengthened us in perilous path, encountered during our task.

ABSTRACT

The **HOSPICARE - Hospital Requirements Analyser and Complaints Resolver** is designed to streamline the process of managing equipment needs and addressing complaints in healthcare institutions. The application provides a centralized platform for hospital staff to post equipment requirements and report issues related to hospital infrastructure or equipment quality.

The system features role-based access, enabling staff members to submit requests and complaints while allowing administrators to manage and track these submissions efficiently. Key functionalities include equipment request submission with detailed specifications, complaint registration with supporting evidence (such as images), real-time status tracking, and automated notifications for updates.

This project aims to improve operational efficiency by ensuring timely resolution of equipment shortages and complaints, fostering accountability, and enhancing the quality of healthcare services. By leveraging modern web technologies, this solution can cater to both small and large-scale healthcare institutions, addressing a critical aspect of hospital management.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION

Efficient management of resources and prompt resolution of issues are critical for ensuring the smooth functioning of hospitals and delivering quality healthcare services. Hospitals often face challenges such as delayed procurement of medical equipment, lack of proper communication channels for reporting issues, and difficulty in tracking the resolution status of complaints.

The **HOSPICARE - Hospital Requirements Analyser and Complaints Resolver** aims to address these challenges by providing a streamlined and user-friendly application. The platform enables hospital staff to submit equipment requirements and lodge complaints regarding hospital infrastructure or medical equipment. Administrators can efficiently manage these submissions, track their status, and ensure timely resolution.

This system ensures transparency, accountability, and effective communication between staff and administrators. By leveraging modern web technologies, it offers a scalable and robust solution tailored to meet the dynamic needs of healthcare institutions. The application's ability to simplify resource management and complaint handling significantly contributes to improving operational efficiency and patient care.

CHAPTER 2

SYSTEM ANALYSIS

In this chapter, we will discuss the **HOSPICARE - Hospital Requirements Analyser and Complaints Resolver**, a solution designed to address challenges in managing hospital resources and complaints. Hospitals face significant issues, including delays in procuring medical equipment, lack of efficient communication channels for complaints, and difficulty in tracking and resolving these issues. This system aims to streamline these processes, making hospital operations more efficient and transparent.

2.1 LITERATURE SURVEY

2.1.1 PURPOSE

The purpose is to streamline the process of managing equipment needs and complaints in hospitals. It ensures efficiency, transparency, and accountability by automating submissions and tracking resolutions in real-time. This system enhances communication and optimizes resource utilization to improve hospital operations and patient care

2.1.2 EXISTING SYSTEM

In the existing system, hospitals often rely on manual or semi-digital processes to handle equipment requests and complaints. These methods involve paperwork, emails, or spreadsheets, which are prone to delays, errors, and mismanagement. Communication gaps between staff and administrators lead to inefficiencies, while the lack of real-time tracking makes it difficult to prioritize urgent needs.

2.2 SYSTEM REQUIREMENTS

2.2.1 NON-FUNCTIONAL REQUIREMENTS

SN No.	Non-Functional Requirement	Description
1	**Performance**	The system should respond to user actions within a few seconds, ensuring quick load times and efficient operations.
2	**Security**	Sensitive data, such as user information and equipment details, must be encrypted. The system should include secure authentication and authorization mechanisms.
3	**Compatibility**	The application should be compatible with modern web browsers (Chrome, Firefox, Safari, etc.)
4	**Backup and Recovery**	Regular backups of system data should be scheduled to prevent data loss and ensure quick recovery in case of system failure.
5	**Usability**	The user interface should be simple, intuitive, and easy to navigate for both staff and administrators.
6	**Availability**	The system should be available 24/7 with minimal downtime.

2.2.2 FUNCTIONAL REQUIREMENTS

SN No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	**User Registration and Authentication**	Users (hospital staff, administrators) should be able to register and log in securely for the equipment registration.
2	**Equipment Request Management**	Users can submit detailed equipment requests including type, quantity, urgency, and description
3	**Complaint Management**	Users can submit complaints related to equipment quality, maintenance issues, or infrastructure problems
4	**Automated Notifications**	The system should notify users of any updates or changes regarding their submissions (e.g., request approval, complaint resolution)

2.3 SOFTWARE AND HARDWARE REQUIREMENTS

2.3.1 SOFTWARE REQUIREMENTS

1. Operating System:

   Ubuntu, Windows, or macOS (for development and deployment).

2. Frontend Development Tools:

   Framework: React.js (for building user interfaces).
   Languages: HTML, CSS, JavaScript.

3. Backend Development Tools:

   Framework: python like flask or Django (for server-side development).

   API Integration: RESTful API for communication between the frontend and backend.

4. Database:

   Database Management System: MySQL (for storing user data, user information, complaint and equipment records).

5. Integrated Development Environment (IDE):

   Visual Studio Code (for coding and debugging).

2.3.2 HARDWARE REQUIREMENTS

1. Processor:

Intel Core i3 or equivalent AMD Ryzen processors (for development and server hosting).

2. RAM:

Minimum 8 GB (recommended 16 GB for handling larger datasets and multiple processes).

3. Storage:

Solid-State Drive (SSD) with at least 512 GB of storage (for faster data access and system performance).

2.4 Hospital Requirements Analyser and Complaints Resolver Modules:

User Registration and Authentication Module:

Enables hospital staff and administrators to securely register and log in to the system. Role-based access ensures staff can submit requests or complaints, while administrators manage and resolve them

Equipment Request Module:

Allows users to submit detailed requests for hospital equipment, specifying quantity, urgency, and other relevant details. It ensures streamlined tracking and prioritization of equipment needs.

Complaint Management Module:

Facilitates the submission of complaints regarding equipment quality, maintenance issues, infrastructure problems. Users can attach supporting evidence like images or documents.

Request and Complaint Tracking Module:

Provides real-time updates on the status of requests and complaints, ensuring transparency and improving communication between staff and administrators.

Notification Module:

Sends automated notifications to users about the progress or resolution of their requests and complaints, ensuring timely updates.

Analytics and Reporting Module:

Generates insights and reports on trends in equipment requests and complaints, helping administrators optimize resource management and address recurring issues.

Administrator Dashboard Module:

Offers a centralized interface for administrators to manage and resolve submissions efficiently. It includes features like prioritizing requests, assigning tasks, and monitoring progress.

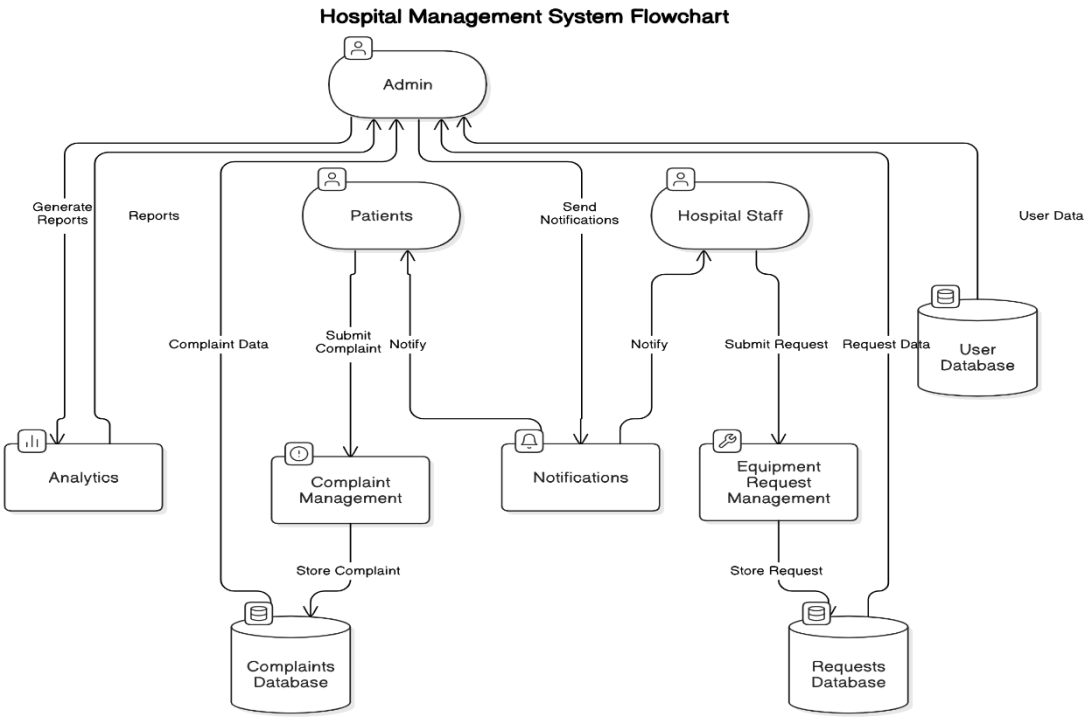
Integration Module:

Ensures seamless integration with existing hospital management systems or databases for better data synchronization and workflow continuity.

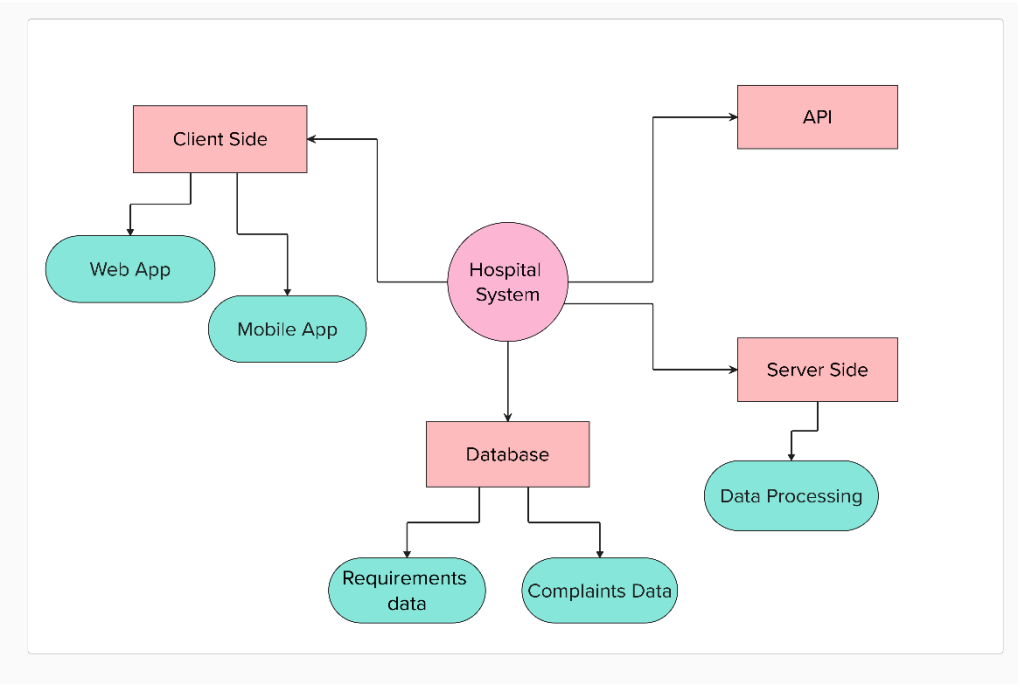
CHAPTER 3

SYSTEM DESIGN

3.1 DATA FLOW DIAGRAMS



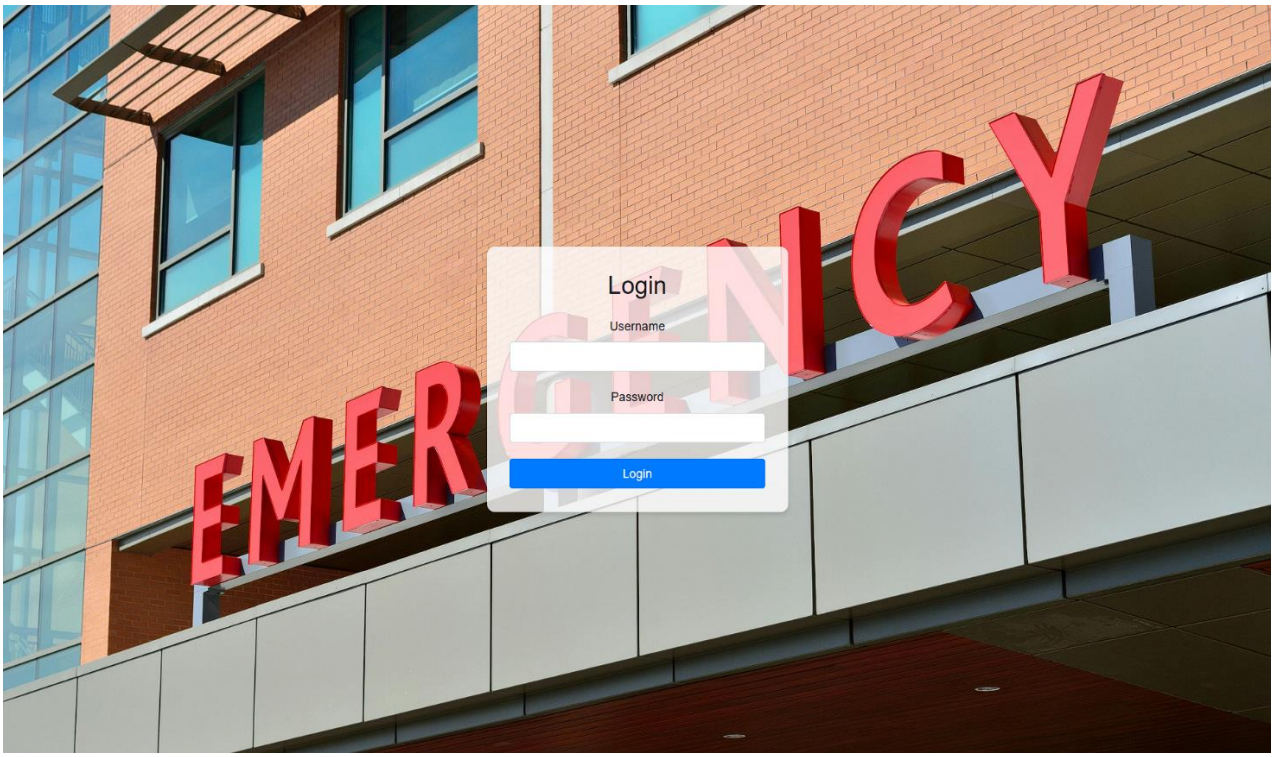
3.2 TECHNICAL ARCHITECTURE



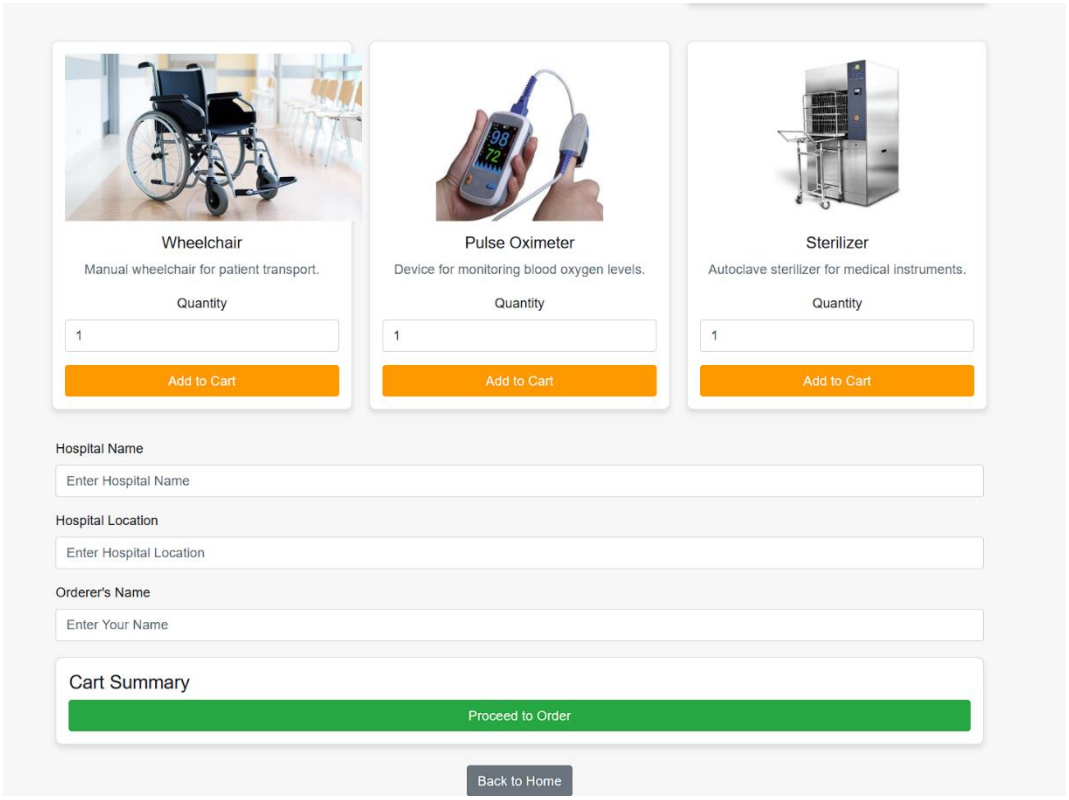
CHAPTER 4

4. SYSTEM IMPLEMENTATION

4.1 **Screenshot for Login Page**

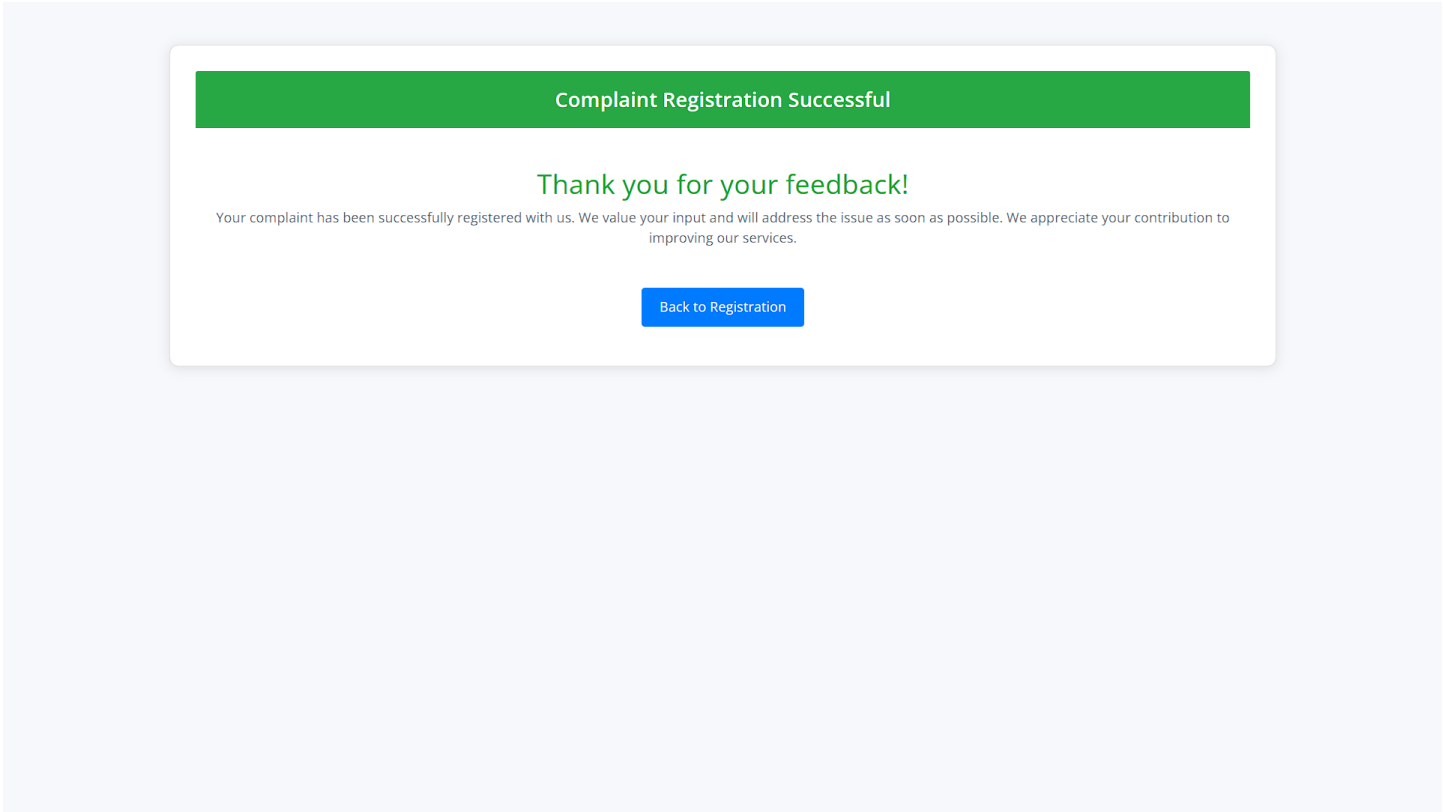


4.2 **Screenshot of Order Equipment**



4.3 **Screenshot of Complaint Registration**

4.4 **Screenshot of Registered Complaint**



CHAPTER 5

5. SYSTEM TESTING

System testing ensures that all components of the HOSPICARE - Hospital Requirements Analyser and Complaints Resolver function seamlessly and meet specified requirements. Below is an outline of the key testing procedures

1. **Functional Testing**

Functional testing ensures that the system performs all its intended functions as per the specifications. Key functionalities such as user registration, login, and role-based access are tested to validate access control mechanisms. The process of submitting equipment requests and complaints is verified to ensure smooth operation. Additionally, real-time status tracking for requests and complaints is tested to confirm accurate updates. Automated notifications for status changes are checked for reliability, and administrator functionalities, including resolving requests and generating reports, are thoroughly validated to ensure the system meets all operational requirements.

2. **Integration Testing**

Integration testing focuses on verifying smooth communication between the system's modules and subsystems. The interaction between the frontend (developed using React.js or Angular) and the backend (implemented with PHP or Python) is tested to ensure seamless data exchange. Database operations with MySQL or PostgreSQL are evaluated to confirm proper data handling. The integration of APIs for sending notifications, such as emails or SMS, is examined to ensure timely and accurate communication between components.

3. **Performance Testing**

Performance testing evaluates the system's responsiveness, speed, and stability under various workloads. Key metrics such as page load time, targeted to be under three seconds, are analyzed for optimal user experience. The system is also tested for scalability to handle high user loads, simulating up to 100,000 simultaneous users to ensure it performs well under peak conditions. This ensures that the system remains reliable and efficient regardless of user traffic.

4. **Security Testing**

Security testing is conducted to protect user data and address potential vulnerabilities. It includes ensuring secure login mechanisms with encryption and safeguarding data during storage and transmission using SSL/TLS protocols. The system is subjected to penetration tests to identify vulnerabilities such as SQL injection and cross-site scripting, ensuring robust security against attacks. This ensures that user information remains secure and the system is resilient to threats.

5. **Usability Testing**

Usability testing assesses the system's ease of use and accessibility. The platform is evaluated for intuitive navigation to ensure staff and administrators can easily perform their tasks. Processes such as submitting and tracking requests or complaints are checked for simplicity and efficiency. Accessibility features are also reviewed to ensure the system caters to diverse user groups, including those with disabilities, enhancing the overall user experience.

6. **Regression Testing**

Regression testing ensures that updates, bug fixes, or enhancements do not negatively impact existing functionalities. Core system features are retested after updates to confirm they function as expected. Compatibility with new hardware or software environments is also verified, ensuring the system remains reliable and efficient even after changes or upgrades. This testing guarantees a seamless experience for users over time.

CHAPTER 6

CONCLUSION

HOSPICARE - Hospital Requirements Analyser and Complaints Resolver streamlines hospital operations by automating equipment requests and complaint management. It ensures timely and transparent processes through real-time tracking, automated notifications, and analytics. The system improves communication, enhances efficiency, and empowers staff to address issues proactively. Administrators gain tools for resource management and data-driven decisions, with robust security ensuring data integrity. Future scope includes AI integration for predictive analytics, blockchain for secure and transparent data management, and mobile app support for accessibility. These advancements will further optimize operations, enhance scalability, and improve patient care outcomes.

CHAPTER 7

REFERENCES

1. **Books and Journals**:

   K. S. Sandhu, *Hospital Management System Design: Approaches and Challenges*, Springer, 2018.

   R. Goyal, *Modern Hospital Management: Strategies and Innovations*, Elsevier, 2021.

2. **World Health Organization (WHO):**
   Guidelines on Equipment Maintenance and Management. Available at https://www.who.int

3. **Research Paper**:
   Automating Equipment Requests in Healthcare Facilities, *Journal of Health Informatics Research*, 2019.

   Improving Complaint Management Systems for Hospitals, *International Journal of Healthcare IT*, 2020.

APPENDIX

INDEX.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Complaint Registration</title>
<linkrel="stylesheet"href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
</head>
<body>
  <div class="container">
    <h1 class="mt-5">Complaint Registration Form</h1>
    <form id="complaintForm" method="POST" action="/register" class="mt-4" enctype="multipart/form-data">
      <!-- Complaint Title -->
      <div class="form-group">
        <label for="complaint_title">Complaint Title</label>
<inputtype="text"class="form-control"id="complaint_title"name="complaint_title" required>
      </div>

      <!-- Description of Complaint -->
      <div class="form-group">
        <label for="complaint_description">Description</label>
        <textarea class="form-control" id="complaint_description" name="complaint_description" rows="4" required></textarea>
      </div>

      <!-- Your Name -->
      <div class="form-group">
        <label for="complainant_name">Your Name</label>
        <input type="text" class="form-control" id="complainant_name" name="complainant_name" required>
      </div>

      <!-- Email Address -->
      <div class="form-group">
        <label for="email">Email Address</label>
        <input type="email" class="form-control" id="email" name="email" required>
      </div>

      <!-- Phone Number -->
      <div class="form-group">
        <label for="phone">Phone Number</label>
        <input type="text" class="form-control" id="phone" name="phone" maxlength="10" pattern="[0-9]{10}" title="Please enter a valid 10-digit phone number" required>
      </div>

      <!-- Aadhar Number -->
```

```
        <div class="form-group">
            <label for="aadhar">Aadhar Number</label>
            <input type="text" class="form-control" id="aadhar" name="aadhar" maxlength="12" pattern="[0-9]{12}" title="Please enter a valid 12-digit Aadhar number" required>
        </div>

        <!-- Complaint Date -->
        <div class="form-group">
            <label for="complaint_date">Date of Complaint</label>
            <input type="date" class="form-control" id="complaint_date" name="complaint_date" required>
        </div>

        <!-- Photo Upload -->
        <div class="form-group">
            <label for="complaint_photo">Upload a Photo (optional)</label>
            <input type="file" class="form-control" id="complaint_photo" name="complaint_photo">
        </div>

        <!-- Submit Button -->
        <button type="submit" class="btn btn-primary">Submit Complaint</button>
    </form>
</div>

<!-- JavaScript for Validation -->
<script>
    document.getElementById('complaintForm').addEventListener('submit', function(event) {
        // Get the complaint date value
        const complaintDate = document.getElementById('complaint_date').value;

        // Get the current date in the format YYYY-MM-DD
        const currentDate = new Date().toISOString().split('T')[0];

        // Check if the selected complaint date is equal to today's date
        if (complaintDate !== currentDate) {
            alert("The complaint date must be today's date.");
            event.preventDefault();  // Prevent form submission
            return;
        }
```

```
            // Get the phone number
            const phoneInput = document.getElementById('phone').value;

            // Validate phone number
            const phonePattern = /^[0-9]{10}$/;
            if (!phonePattern.test(phoneInput)) {
                alert("Please enter a valid 10-digit phone number.");
                event.preventDefault(); // Prevent form submission
                return;
            }

            // Get the Aadhar number
            const aadharInput = document.getElementById('aadhar').value;

            // Validate Aadhar number
            const aadharPattern = /^[0-9]{12}$/;
            if (!aadharPattern.test(aadharInput)) {
                alert("Please enter a valid 12-digit Aadhar number.");
                event.preventDefault(); // Prevent form submission
                return;
            }
        });
    </script>

    <!-- Include Bootstrap JS (optional, for responsive behavior) -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></scrip
t>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

PYTHON FLASK

```
from flask import Flask, render_template, request, redirect, url_for
import mysql.connector
from mysql.connector import Error
import os
```

```
app = Flask(__name__)

# Set the folder for file uploads
UPLOAD_FOLDER = 'uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# MySQL connection parameters
host = 'localhost'
user = 'root'
password = 'Nagasrinivas@11'
database = 'hospital_db'

# Ensure the upload folder exists
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Function to connect to MySQL database
def get_db_connection():
    try:
        # Establish connection to MySQL
        conn = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )
        if conn.is_connected():
            return conn
        else:
            return None
    except Error as e:
        print(f"Error: {e}")
        return None

# Function to initialize the database and table
def initialize_db():
    try:
        # Connect to MySQL server (without database)
        conn = mysql.connector.connect(
```

```
            host=host,
            user=user,
            password=password
        )
        if conn.is_connected():
            print("Connected to MySQL server")

            cursor = conn.cursor()
            # Create the database if it doesn't exist
            cursor.execute(f"CREATE DATABASE IF NOT EXISTS {database}")
            cursor.execute(f"USE {database}")

            # Create the complaintregister table if it doesn't exist
            cursor.execute('''
            CREATE TABLE IF NOT EXISTS complaintregister (
                id INT AUTO_INCREMENT PRIMARY KEY,
                complaint_title VARCHAR(255) NOT NULL,
                complaint_description TEXT NOT NULL,
                complainant_name VARCHAR(255) NOT NULL,
                email VARCHAR(255) NOT NULL,
                phone VARCHAR(15) NOT NULL,
                aadhar VARCHAR(12) NOT NULL,  -- Aadhar field
                complaint_date DATE NOT NULL,
                complaint_photo VARCHAR(255)  -- Path to the uploaded photo
            )
            ''')

            print("Database and table created successfully.")
            cursor.close()
            conn.close()
    except Error as e:
        print(f"Error initializing database: {e}")

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/register', methods=['POST'])
def register_complaint():
```

```
    complaint_title = request.form['complaint_title']
    complaint_description = request.form['complaint_description']
    complainant_name = request.form['complainant_name']
    email = request.form['email']
    phone = request.form['phone']
    aadhar = request.form['aadhar']
    complaint_date = request.form['complaint_date']

    # Handle photo upload
    complaint_photo = request.files.get('complaint_photo')
    photo_filename = None
    if complaint_photo:
        # Save the photo to the server
        photo_filename = os.path.join(app.config['UPLOAD_FOLDER'],
complaint_photo.filename)
        complaint_photo.save(photo_filename)

    # Connect to MySQL database
    conn = get_db_connection()

    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute('''
                INSERT INTO complaintregister (complaint_title, complaint_description,
complainant_name, email, phone, aadhar, complaint_date, complaint_photo)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
            ''', (complaint_title, complaint_description, complainant_name, email, phone,
aadhar, complaint_date, photo_filename))
            conn.commit()
            return redirect(url_for('view_complaints'))
        except Error as e:
            return f"Database error: {e}"
        finally:
            if conn.is_connected():
                cursor.close()
                conn.close()

    return "Unable to connect to the database."
```

```
@app.route('/complaints')
def view_complaints():
    conn = get_db_connection()

    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM complaintregister")
            complaints = cursor.fetchall()  # Fetch all rows

            return render_template('complaints.html', complaints=complaints)
        except Error as e:
            return f"Database error: {e}"
        finally:
            if conn.is_connected():
                cursor.close()
                conn.close()

    return "Unable to connect to the database."

if __name__ == '__main__':
    # Initialize the database and table when the app starts
    initialize_db()
    app.run(debug=True)
```

REQUIREMENT.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hospital Equipment Order Form</title>
    <link  href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
```

```
<style>
    body {
        font-family: Arial, sans-serif;
        background-color: #f7f7f7;
    }
    .container {
        margin-top: 30px;
    }
    .product-card {
        border: 1px solid #ddd;
        border-radius: 8px;
        background-color: #fff;
        padding: 15px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        text-align: center;
    }
    .product-card img {
        max-height: 200px;
        object-fit: cover;
        margin-bottom: 15px;
    }
    .product-card h5 {
        font-size: 1.2rem;
    }
    .product-card .btn-add-to-cart {
        background-color: #ff9900;
        color: white;
    }
    .product-card .btn-add-to-cart:hover {
```

```
    background-color: #e68900;
}
.cart-summary {
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 15px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    margin-top: 20px;
}
.cart-summary h4 {
    font-size: 1.4rem;
}
.cart-summary .btn-proceed {
    background-color: #28a745;
    color: white;
}
.cart-summary .btn-proceed:hover {
    background-color: #218838;
}
.cart-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 10px;
}
.cart-item button {
    background-color: #ff5733;
    color: white;
```

```
        border: none;
        cursor: pointer;
    }
    .cart-item button:hover {
        background-color: #ff2d1b;
    }
    /* Responsive grid */
    .row {
        display: flex;
        flex-wrap: wrap;
    }
    .col-md-4 {
        flex: 0 0 33.333%;
        max-width: 33.333%;
        padding: 10px;
    }
    @media (max-width: 768px) {
        .col-md-4 {
            flex: 0 0 50%;
            max-width: 50%;
        }
    }
    @media (max-width: 480px) {
        .col-md-4 {
            flex: 0 0 100%;
            max-width: 100%;
        }
    }
</style>
```

```
</head>
<body>

<div class="container">
  <h2 class="text-center mb-4">Order Required Equipment</h2>

  <div class="row" id="product-list">
    <!-- Dynamic product cards will be injected here -->
  </div>

  <!-- Hospital Information Form -->
  <div class="form-group">
    <label for="hospitalName">Hospital Name</label>
    <input type="text" class="form-control" id="hospitalName" placeholder="Enter Hospital Name" required>
  </div>
  <div class="form-group">
    <label for="hospitalLocation">Hospital Location</label>
    <input type="text" class="form-control" id="hospitalLocation" placeholder="Enter Hospital Location" required>
  </div>
  <div class="form-group">
    <label for="ordererName">Orderer's Name</label>
    <input type="text" class="form-control" id="ordererName" placeholder="Enter Your Name" required>
  </div>

  <div class="cart-summary">
    <h4>Cart Summary</h4>
    <div id="cartItemsList"></div>
```

<button class="btn btn-proceed btn-block" onclick="submitOrder()">Proceed to Order</button>

 Back to Home

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<script>

let cart = [];

const products = [

{ name: "Oxygen Concentrator", image: "oxygen.jpg", description: "Portable oxygen concentrator for hospital use." },

{ name: "Patient Bed", image: "bed.jpg", description: "Adjustable hospital bed for patients." },

{ name: "Infusion Pump", image: "pump.jpg", description: "Precise infusion pump for controlled drug administration." },

{ name: "ECG Machine", image: "ecg.jpg", description: "Electrocardiogram machine for heart monitoring." },

{ name: "Blood Pressure Monitor", image: "bpmonitor.jpg", description: "Automatic blood pressure measuring device." },

{ name: "Ventilator", image: "ventilator.jpg", description: "Medical ventilator for patient respiratory support." },

{ name: "Surgical Light", image: "light.jpg", description: "High-intensity light for surgical procedures." },

{ name: "X-Ray Machine", image: "xray.jpg", description: "X-Ray imaging equipment for diagnostic use." },

```
    { name: "Defibrillator", image: "defibrillator.jpg", description: "Device for emergency heart rhythm correction." },
    { name: "Wheelchair", image: "wheelchair.jpg", description: "Manual wheelchair for patient transport." },
    { name: "Pulse Oximeter", image: "pulse.jpg", description: "Device for monitoring blood oxygen levels." },
    { name: "Sterilizer", image: "sterilizer.jpg", description: "Autoclave sterilizer for medical instruments." }
  ];

  function addToCart(itemName, id) {
    const quantity = document.getElementById('quantity' + id).value;
    cart.push({ itemName: itemName, quantity: quantity });
    updateCart();
  }

  function removeFromCart(index) {
    cart.splice(index, 1);
    updateCart();
  }

  function updateCart() {
    const cartList = document.getElementById('cartItemsList');
    cartList.innerHTML = '';
    cart.forEach((item, index) => {
      const cartItemDiv = document.createElement('div');
      cartItemDiv.classList.add('cart-item');
      cartItemDiv.innerHTML = `
        <span>${item.itemName} - Quantity: ${item.quantity}</span>
        <button onclick="removeFromCart(${index})">Remove</button>
```

```
        `;
        cartList.appendChild(cartItemDiv);
    });
}

function submitOrder() {
    if (cart.length === 0) {
        alert('Please add items to your cart before proceeding.');
        return;
    }

    // Collect the hospital and orderer details
    const hospitalName = document.getElementById('hospitalName').value;
    const hospitalLocation = document.getElementById('hospitalLocation').value;
    const ordererName = document.getElementById('ordererName').value;

    if (!hospitalName || !hospitalLocation || !ordererName) {
        alert('Please fill in all the hospital details.');
        return;
    }

    const orderData = {
        hospitalName: hospitalName,
        hospitalLocation: hospitalLocation,
        ordererName: ordererName,
        cartItems: cart
    };

    // Send the order data to the Flask backend
```

```
    fetch('/submit_order', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(orderData)
    })
    .then(response => response.json())
    .then(data => {
        alert(data.message || 'Order submitted successfully!');
    })
    .catch(error => {
        alert('Error submitting order: ' + error);
    });
}

function loadProducts() {
    const productListDiv = document.getElementById('product-list');
    products.forEach((product, index) => {
        const productCard = document.createElement('div');
        productCard.classList.add('col-md-4', 'mb-4');
        productCard.innerHTML = `
            <div class="product-card">
                <img src="/static/images/${product.image}" alt="${product.name}">
                <h5>${product.name}</h5>
                <p class="text-muted">${product.description}</p>
                <div class="form-group">
                    <label for="quantity${index + 1}">Quantity</label>
                    <input type="number" class="form-control" id="quantity${index + 1}"
name="quantity${index + 1}" min="1" value="1">
                </div>
```

```
                <button class="btn btn-add-to-cart btn-block" onclick="addToCart('${product.name}', ${index + 1})">Add to Cart</button>
            </div>
        `;
        productListDiv.appendChild(productCard);
    });
}

window.onload = loadProducts;
</script>
</body>
</html>
```