RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS23332 DATABASE MANAGEMENT SYSTEMS LAB

EQUIPMENT MANAGEMENT SYSTEM
A MINI PROJECT REPORT

Submitted by

MONISH S	**231501103**
MOUNESH KUMARAN	**231501104**
NANDHINI PRAKASH	**231501105**

In partial fulfilment for the award of the degree of
BACHELOR OF ENGINEERING IN
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM
CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report "**EQUIPMENT MANAGEMENT SYSTEM**" is the Bonafide work of **"MONISH S (231501103), MOUNESH KUMARAN (231501104), NANDHINI PRAKASH (231501105)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on ________________

SIGNATURE
Mr. U. Kumaran,
Assistant Professor (SS)
AIML,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This paper presents the design and implementation of an Equipment Management System, a web-based application that automates various hospital equipment operations. The system leverages a MySQL database to store and manage critical information, including equipment details, orders, and service history.

The user-friendly interface, built using HTML and CSS, enables efficient equipment management, order placement, usage tracking, and notification delivery. By automating routine tasks and providing valuable insights, the system aims to streamline hospital operations, improve equipment availability, and enhance overall service efficiency. The Equipment Management System is a simple console application with a MySQL database for the backend and HTML and CSS for the frontend.

The user can perform basic equipment management operations like adding equipment, placing orders, removing or updating equipment details, and tracking service status.
Each piece of equipment in the system has a unique identification number.

The user can request equipment based on availability and order equipment from a list of suggestions provided by the admin. The system also automatically adds equipment to the order list when stock levels are low.
When the equipment status indicates service or replacement, the administrator can send a notification to the relevant personnel about the required action, ensuring timely maintenance or replenishment.

TABLE OF CONTENTS

INTRODUCTION

1.1 INTRODUCTION

In the modern healthcare sector, medical equipment plays a crucial role in delivering high-quality care. Hospitals and healthcare facilities rely on various equipment types, ranging from basic diagnostic tools to advanced imaging and surgical instruments. However, managing this equipment effectively poses significant challenges. Ensuring availability, monitoring usage, tracking maintenance, and maintaining accurate records are essential to maximize equipment efficiency, minimize downtime, and reduce operational costs.

This project aims to develop a Medical Equipment Management System (MEMS) using a database management system (DBMS). The system provides a centralized platform to store and manage information about medical equipment, including equipment details, purchase history, location, current status, and maintenance schedules. By leveraging a DBMS, the project seeks to improve data accuracy, streamline record-keeping processes, and facilitate easy access to crucial information for decision-makers. This report documents the development process, database design, system functionalities, and testing results of the MEMS, demonstrating its potential to enhance equipment management in healthcare settings.

1.2 OBJECTIVE

The objective of this project is to design and implement a robust and efficient Medical Equipment Management System (MEMS) that leverages database management principles to streamline the tracking, maintenance, and utilization of medical equipment within healthcare facilities.

1.3 MODULES

- Equipment Management Module
- Maintenance and Scheduling Module
- Inventory and Allocation Module
- Supplier and Vendor Management Module
- User Access and Authentication Module

SURVEY OF TECHNOLOGY

2.1 SOFTWARE DESCRIPTION

VISUAL STUDIO CODE

Visual Studio Code, commonly referred to as VS Code, is a powerful, open-source code editor developed by Microsoft. Designed with flexibility and productivity in mind, VS Code combines the simplicity of a text editor with advanced developer tooling, making it suitable fora wide range of programming tasks across different languages and frameworks. It has gained immense popularity in the developer community due to its user-friendly interface, extensibility, and efficient features that streamline the development process.

KEY FEATURES AND BENEFITS

1. **INTELLISENSE CODE COMPLETION:**

   VS Code's IntelliSense offers intelligent autocompletion for variables, methods, and functions, speeding up the coding process and reducing errors by suggesting the correct syntax and function names.

2. **BUILT-IN DEBUGGING:**

   VS Code includes integrated debugging tools, allowing developers to set breakpoints and inspect code execution, making bug fixing faster and more efficient without switching between tools.

3. **CUSTOMIZATION AND EXTENSIONS:**

   With a vast marketplace of extensions, VS Code can be tailored to specific needs, enhancing functionality with support for new languages, Git integration, and tools for testing and deployment.

4. **VERSION CONTROL WITH GIT INTEGRATION:**

   VS Code’s built-in Git support enables easy version control, allowing developersto manage code, commit changes, and push updates directly within the editor.

LANGUAGES

2.2.1 HTML (HYPERTEXT MARKUP LANGUAGE) – FRONT END

HTML, or HyperText Markup Language, is the standard language used to structure and layout web pages. It forms the backbone of all web content, defining the organization of text, images, links, and multimedia on a webpage. Through various HTML tags, developers can create different types of content—such as headings, paragraphs, lists, forms, tables, and hyperlinks—that define the basic structure of any website. HTML is integral in web development, as it allows browsers to render and display content in a structured, readable format for users.

PURPOSE IN EDUEASE:

HTML (HyperText Markup Language) serves as the foundational language for creating and structuring content on the web. Its main purposes are:

1. **Structure Web Content**: HTML defines the structure of web pages by using elements like headings, paragraphs, lists, links, images, and tables.
2. **Content Formatting**: It allows basic styling of content (though CSS is typically used for more advanced styling), like bold text, italics, and alignment.
3. **Navigation**: HTML enables the creation of hyperlinks, allowing users to navigate between different pages or websites.
4. **Embedding Media**: It allows embedding multimedia content such as images, videos, and audio files within web pages.
5. **Forms & Input**: HTML provides forms for collecting user data, such as login credentials, surveys, and contact information.
6. **Accessibility**: It supports accessibility features like screen reader compatibility for users with disabilities.

2.2.2 CSS (CASCADING STYLE SHEETS) – FRONT END

CSS, or Cascading Style Sheets, is the primary styling language used to define the visual presentation of web applications. It brings HTML elements to life, dictating the layout, colors, fonts, spacing, and responsiveness that collectively create an engaging and accessible interface. CSS is essential for transforming a basic HTML structure into a polished, visually appealing, and user-friendly application. By managing style elements, CSS helps create a cohesive, consistent design across devices, ensuring that applications remain visually consistentand accessible on various platforms.

PURPOSE OF CSS IN THE EDUEASE APPLICATION:

In the **Medical Equipment Management System**, **CSS** is essential for designing and styling the user interface. Its purposes include:

1. **Layout and Structure**: CSS organizes the placement and alignment of elements on the page, such as menus, buttons, tables, and forms, to create a clean and organized layout.
2. **Visual Design**: CSS controls colors, fonts, borders, and backgrounds, making the interface visually appealing and aligned with the branding, which is crucial for a professional, trustworthy look.
3. **Responsive Design**: CSS ensures the application looks and functions well on different devices and screen sizes, including desktops, tablets, and smartphones, making it accessible for all users.
4. **User Experience Enhancements**: CSS adds visual cues like hover effects on buttons, animations, and transitions, helping users navigate the system smoothly and intuitively.
5. **Consistency Across Pages**: CSS maintains a uniform style across all pages, providing a cohesive and polished look, which improves usability and professionalism.

2.2.3 JAVASCRIPT (JS) – FRONT END

In frontend development, **JavaScript** plays a critical role in making web pages dynamic, engaging, and responsive. Its main functions include:

1. **Creating Interactivity**: JavaScript powers interactive elements such as dropdown menus, sliders, modals, and animations, helping users interact with the website intuitively.

2. **Real-Time Updates**: It enables live updates to page content without reloading, allowing for responsive notifications, live form validations, and dynamic data displays.

3. **Validating User Input**: JavaScript checks the accuracy and completeness of user inputs (like required fields, email formats, and password strength) before sending data to the server, reducing errors and improving usability.

4. **Enhancing Responsive Design**: JavaScript, in combination with CSS, helps adjust the layout and functionality based on screen sizes or devices, ensuring an optimal experience across desktops, tablets, and smartphones.

5. **Handling Asynchronous Requests**: Using AJAX or Fetch API, JavaScript can request data from the server in the background, allowing smooth page transitions, faster updates, and overall better performance.

JavaScript is essential in transforming static HTML and CSS layouts into a seamless, dynamic experience that today’s users expect on modern websites and applications.

2.2.4 MySQL - DATABASE

MySQL is an open-source **Relational Database Management System (RDBMS)** that uses Structured Query Language (SQL) to manage, organize, and retrieve data. It is widely used for web and application development due to its reliability, scalability, and compatibility with various platforms.

Key Characteristics:

1. **Data Storage**: MySQL organizes data in **tables** consisting of rows and columns, making it easy to manage large amounts of structured data.
2. **Relational Structure**: It supports relationships between tables, enabling complex data connections (e.g., linking orders to specific equipment in an inventory system).
3. **SQL Support**: MySQL uses SQL commands like SELECT, INSERT, UPDATE, and DELETE to interact with data.
4. **Data Integrity and Security**: It provides features like transactions, indexing, and ACID (Atomicity, Consistency, Isolation, Durability) compliance, ensuring data reliability and security.
5. **Scalability**: MySQL is highly scalable, supporting everything from small applications to large enterprise databases with high transaction volumes.

PURPOSE OF MySQL:

In a **Medical Equipment Management System (MEMS)**, **MySQL** plays a crucial role in:

1. **Data Storage**: Stores inventory, maintenance, and usage data for medical equipment.
2. **Querying**: Allows quick retrieval of equipment details and service history.
3. **Inventory Management**: Tracks equipment status and quantity.
4. **Scheduling & Alerts**: Manages maintenance schedules and notifications.
5. **Access Control**: Ensures secure user permissions for data.
6. **Reporting**: Generates usage, maintenance, and cost reports.
7. **Compliance**: Ensures data meets regulatory standards for audits.
8. **Scalability & Integration**: Supports system growth and integration with other healthcare systems.

2.2.5 FLASK – PYTHON (BACKEND)

Flask is a lightweight Python web framework ideal for building dynamic web applications. In the **Equipment Management System (P2)**, Flask enables seamless interaction between the frontend and backend for efficient equipment tracking, order management, and notifications. Its simplicity and flexibility make it suitable for implementing user-specific features and managing hospital operations effectively.

Key Characteristics of Flask:

1. **Lightweight Framework**: Minimal overhead for easy customization.
2. **Dynamic Routing**: Manages user-specific views like requests and approvals.
3. **Database Integration**: Ensures efficient data handling with MySQL.
4. **Template Rendering**: Displays real-time data using Jinja2 templates.
5. **Built-in Server**: Simplifies testing and debugging.
6. **Extensibility**: Supports secure authentication and notifications.

Purpose of Flask:

1. **User Role Management**: Differentiates access for members, managers, and admins.
2. **Order Processing**: Enables placing, approving, or rejecting equipment orders.
3. **Notifications**: Automates alerts for low stock or service needs.
4. **Real-Time Data Display**: Shows equipment status and updates dynamically.
5. **Scalable Architecture**: Supports future enhancements without major changes.

REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION

3.1.1 FUNCTIONAL REQUIREMENTS:

1. **Equipment Request Submission:** Healthcare staff can easily submit equipment requests through a user-friendly form built with HTML and React. The form allows users to select equipment from a catalog, specify the quantity needed, and provide the purpose or reason for the request. This system ensures that all data is collected accurately and submitted seamlessly.

2. **Approval and Allocation Management:** Administrators or authorized staff can review and approve equipment requests through an intuitive approval interface, built with React. The system allows users to approve, reject, or modify requests with a single click, ensuring fast and efficient decision-making. The workflow streamlines the approval process and reduces the time spent on administrative tasks.

3. **Notification System:** The system features a real-time notification service, powered by JavaScript and React, which instantly notifies staff about the status of equipment requests. Both requesters and approvers are informed of approvals, rejections, or any status changes, ensuring transparent communication and timely updates.

4. **Dashboard and Record Management:** The platform includes a React-based dashboard for both staff and administrators, enabling them to view, track, and manage equipment requests. The system stores all requests, approval statuses, and equipment usage data in a MongoDB database, ensuring that all records are organized, searchable, and easily retrievable for audits and future reference.

3.1.2 NON-FUNCTIONAL REQUIREMENTS:

1. **High Accessibility:** The Medical Equipment Management System, built using HTML, CSS, and React, is designed to be fully responsive, ensuring accessibility across a wide range of devices, including desktops, tablets, and mobile phones. The responsive design adjusts to different screen sizes, providing an optimal user experience across all platforms. This ensures that staff and administrators can access the system anytime, anywhere, improving convenience and operational efficiency.

2. **Secure Database:** The system’s database, powered by MongoDB, ensures secure and encrypted storage of sensitive data related to medical equipment, staff, and requests. All personal information, equipment records, and request details are stored with encryption and secure access protocols, protecting against unauthorized access. MongoDB's security features maintain the confidentiality and integrity of the data, helping to comply with health data protection regulations.

3. **Scalability:** The platform is designed with scalability in mind to accommodate increasing data and usage as the organization grows. MongoDB’s flexible schema and horizontal scaling capabilities allow the system to handle an expanding volume of equipment requests and user data efficiently. As the demand for medical equipment management increases, the system can scale seamlessly, ensuring continued reliability and performance.

HARDWARE AND SOFTWARE REQUIREMENTS

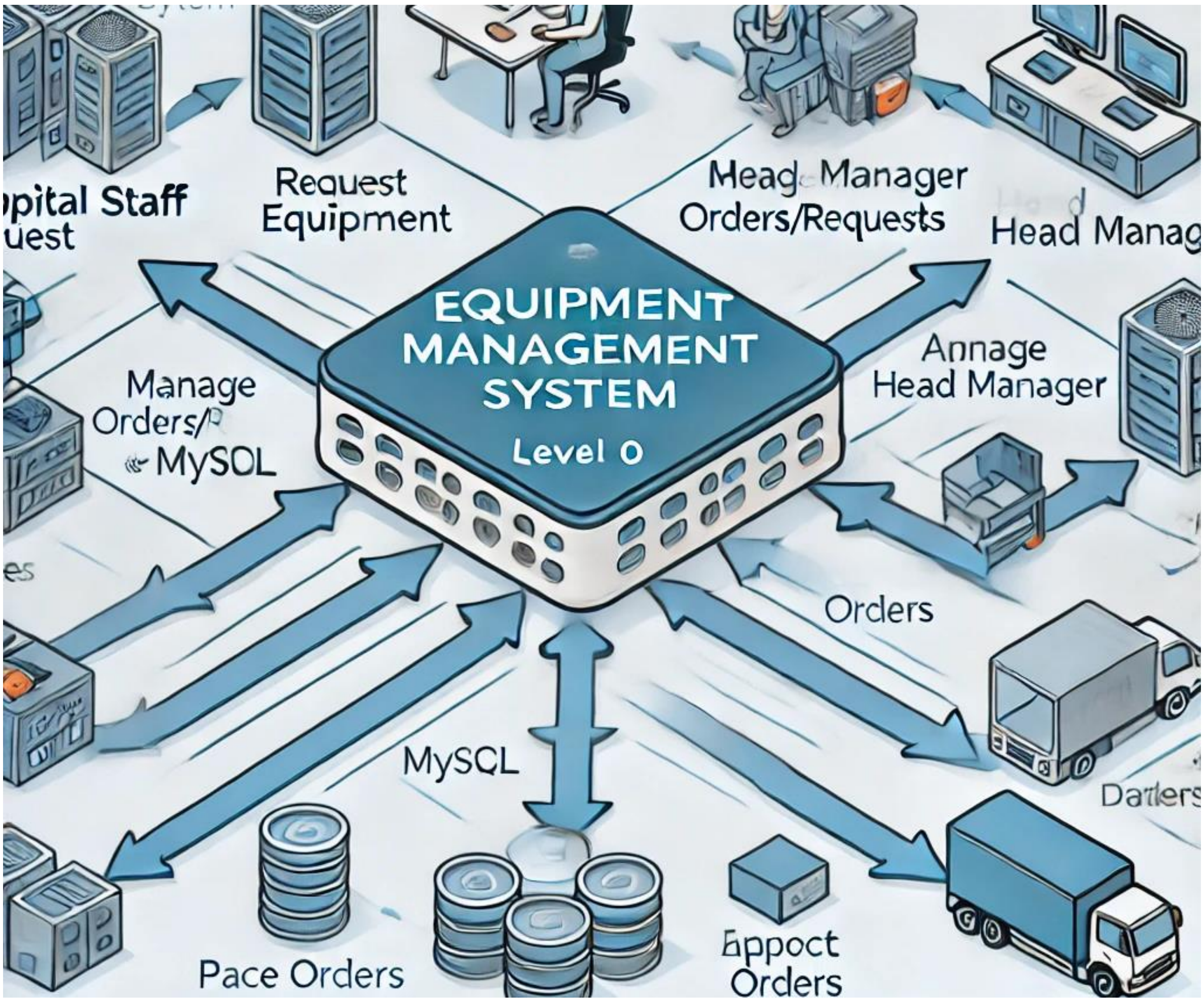
HARDWARE REQUIREMENTS:

- **Computer with Windows 10/Linux Operating System:** Required for both development and server deployment, ensuring the system can run development environments and production environments effectively.
- **Web Server or Cloud-Based Server:** Necessary for hosting the Medical Equipment Management System and providing real-time access to all users, ensuring the application is available at all times.
- **Minimum 4 GB RAM and 500 GB Storage:** Ensures that the application runs efficiently, supporting data storage, real-time processing, and the high-performance requirements of equipment management and tracking.

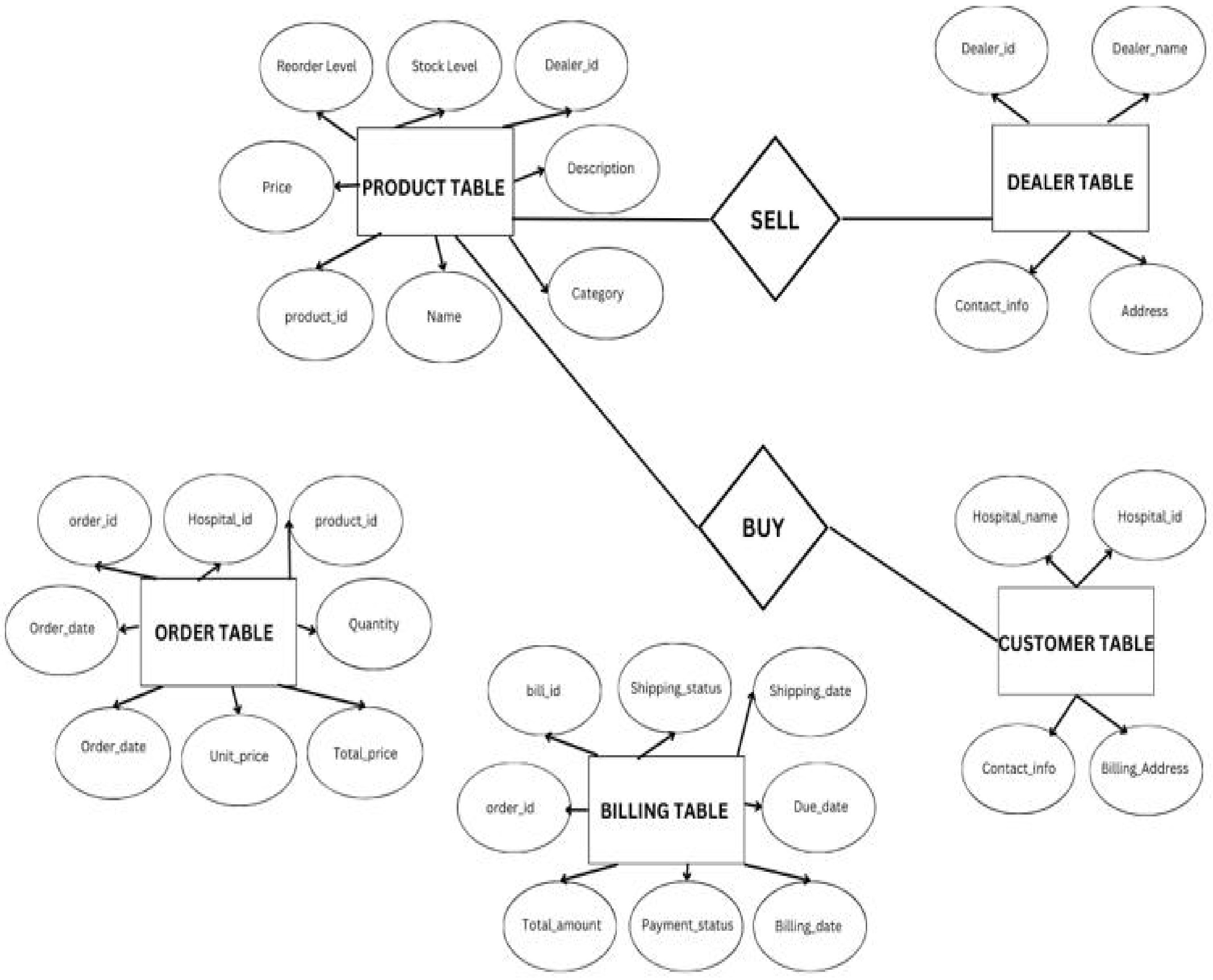
SOFTWARE REQUIREMENTS:

1. **HTML, CSS, and React (Front-End):** Used for creating a responsive and interactive interface. React, along with HTML and CSS, forms the core front-end technologies for developing a user-friendly experience for staff and administrators.
2. **MongoDB (Back-End):** A NoSQL database chosen for its flexibility and scalability, ideal for storing medical equipment data, inventory records, maintenance logs, and request histories. It supports rapid data retrieval and efficient management of large datasets.
3. **JavaScript (Back-End):** Powers the interactive and dynamic functionalities of the system, including real-time notifications, equipment request tracking, and event handling. When combined with Node.js, it supports server-side logic and operations.

ARCHITECTURE DIAGRAM OF MEDICAL EQUIPMENT MANAGEMENT



ER DIAGRAM


Reorder Level
Stock Level
Dealer_id
Price
PRODUCT TABLE
Description
product_id
Name
Category
SELL
Dealer_id
Dealer_name
DEALER TABLE
Contact_info
Address
BUY
Hospital_name
Hospital_id
CUSTOMER TABLE
Contact_info
Billing_Address
order_id
Hospital_id
product_id
Order_date
ORDER TABLE
Quantity
Order_date
Unit_price
Total_price
bill_id
Shipping_status
Shipping_date
order_id
BILLING TABLE
Due_date
Total_amount
Payment_status
Billing_date

PROGRAM CODE

ARENA FOR CODING: VISUAL STUDIO CODE

CODE FOR SIGN UP PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Sign Up</h1>
    <form action="/signup" method="POST">
        <input type="text" name="name" placeholder="Name" required>
        <input type="password" name="password" placeholder="Password" required>
        <select name="role" required>
            <option value="member">Member</option>
            <option value="manager">Manager</option>
        </select>
        <button type="submit">Sign Up</button>
    </form>
</body>
</html>
```

CODE FOR LOGIN PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-6">
                <h2 class="text-center">Login</h2>
                <form action="{{ url_for('login') }}" method="POST">
                    <div class="form-group">
                        <label for="username">Username</label>
                        <input type="text" class="form-control" id="username" name="username" required>
                    </div>
                    <div class="form-group">
                        <label for="password">Password</label>
                        <input type="password" class="form-control" id="password" name="password" required>
                    </div>
                    <button type="submit" class="btn btn-primary btn-block">Login</button>
                    <br>
                    <button onclick="location.href='/signup'">Sign Up</button>
            </form>
            {% with messages = get_flashed_messages() %}
              {% if messages %}
                <div class="alert alert-danger mt-3">
                  {{ messages[0] }}
                </div>
              {% endif %}
            {% endwith %}
          </div>
        </div>
    </div>
</body>
</html>
```

CODE FOR ORDER PLACING

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Orders to be Placed</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Orders to be Placed</h1>
    {% for order in orders %}
    <p><strong>Order ID:</strong> {{ order[0] }}</p>
    <p><strong>Order Date:</strong> {{ order[1] }}</p>
    <p><strong>Recommended Dealer:</strong> {{ order[2] }}</p>
    <p><strong>Dealer Contact:</strong> {{ order[3] }}</p>
    <p><strong>Dealer Address:</strong> {{ order[4] }}</p>
    <p><strong>Equipment Name:</strong> {{ order[5] }}</p>
    <br><hr>
{% endfor %}

</body>
</html>
```

CODE FOR VIEWING ALL ORDERS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>All Orders</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Orders:</h1>
    <table>
        <tr>
            <th>Order ID</th>
            <th>Equipment Name</th>
            <th>Quantity Ordered</th>
            <th>Order Date</th>
            <th>Room Requested</th>
            <th>Approval Status</th>
        </tr>
        {% for order in orders %}
        <tr>
            <td>{{ order[0] }}</td>
            <td>{{ order[1] }}</td>
            <td>{{ order[2] }}</td>
            <td>{{ order[3] }}</td>
            <td>{{ order[4] }}</td>
            <td>{{ order[5] }}</td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```

CODE FOR EQUIPMENT SEARCH AND REQUEST USAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Equipment Search</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <form action="/equipment_search" method="POST" id="search-form">
        <input type="text" id="search-name" name="search_name" placeholder="Search by equipment name"
required>
        <button type="submit">Search</button>
    </form>

    {% if equipment %}
        <h2>Search Results:</h2>
        <ul>
        {% for item in equipment %}
            <li>
                <p><strong>ID:</strong> {{ item[0] }}</p>
                <p><strong>Name:</strong> {{ item[1] }}</p>
                <form action="/request_usage" method="POST">
                    <input type="hidden" name="equipment_id" value="{{ item[0] }}">
                    <input type="text" name="room_requested" placeholder="Room/Patient ID" required>
                    <button type="submit">Request Usage</button>
                </form>
            </li>
        {% endfor %}
        </ul>
    {% endif %}

    <div id="search-results"></div>
</body>
</html>
```

CODE FOR VIEWING PENDING ORDERS

```
<!-- view_orders.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Order List</title>
    <link rel="stylesheet" href="styles.css">
    <script>
        function approveOrder(orderId) {
            fetch(`/approve_order/${orderId}`, { method: 'POST' })
                .then(response => response.json())
                .then(data => {
                    if (data.success) {
                        alert("Sent the details to warehouse");
                        document.getElementById(`order-row-${orderId}`).remove();
                    } else {
                        alert("Failed to approve order");
                    }
                });
        }
        function rejectOrder(orderId) {
            fetch(`/reject_order/${orderId}`, { method: 'POST' })
                .then(response => response.json())
                .then(data => {
                    if (data.success) {
                        alert("Order rejected");
                        document.getElementById(`order-row-${orderId}`).remove();
                    } else {
```

```
                    alert("Failed to reject order");
                }
            });
        }
    </script>
</head>
<body>
    <h1>Pending Orders</h1>
    <table>
        <tr>
            <th>Order ID</th>
            <th>Equipment Name</th>
            <th>Quantity Ordered</th>
            <th>Order Date</th>
            <th>Room Requested</th>
            <th>Requested By</th>
            <th>Approval Status</th>
            <th>Action</th>
        </tr>
        {% for order in orders %}
        <tr id="order-row-{{ order.order_id }}">
            <td>{{ order.order_id }}</td>
            <td>{{ order.equipment_name }}</td>
            <td>{{ order.quantity_ordered }}</td>
            <td>{{ order.order_date }}</td>
            <td>{{ order.room_requested }}</td>
            <td>{{ order.requested_by }}</td>
            <td>{{ order.approval_status }}</td>
            <td>
                <button onclick="approveOrder({{ order.order_id }})">Approve</button>
                <button onclick="rejectOrder({{ order.order_id }})">Reject</button>
            </td>
        </tr>
        {% endfor %}
    </table>
    <button onclick="location.href='/manager/orders'">View All Orders</button>
    <button onclick="location.href='/manager/orders_to_place'">View Orders to Place</button>

</body>
</html>
```

CODE FOR CONNECTING MySQL SERVER TO THE FRONT-END

```
import os

from flask import Flask, render_template, request, redirect, url_for, jsonify, flash, session

from flask_mysqldb import MySQL

from datetime import datetime, date

import mysql.connector

from flask import render_template

import bcrypt

import MySQLdb.cursors


app = Flask(__name__)

app.secret_key = 'ff80b591e8778b991e81613ad5b641182fddbe7769eb1dba'


# Database connection

app.config['MYSQL_HOST'] = 'localhost'

app.config['MYSQL_USER'] = 'root'

app.config['MYSQL_PASSWORD'] = 'nandy'

app.config['MYSQL_DB'] = 'hospital_inventory'


mysql = MySQL(app)


@app.route('/')

def index():

    return render_template('login.html')
```

```
@app.route('/login', methods=['GET', 'POST'])

def login():

    username = request.form.get('username')

    password = request.form.get('password')



    cursor = mysql.connection.cursor()

    cursor.execute("SELECT id, password, role FROM Users WHERE name = %s", (username,))

    user = cursor.fetchone()

    cursor.close()



    if user and bcrypt.checkpw(password.encode('utf-8'), user[1].encode('utf-8')):

        session['user_id'] = user[0]

        session['role'] = user[2]



        if user[2] == 'member':

            return redirect(url_for('equipment_search'))

        elif user[2] == 'manager':

            return redirect(url_for('view_orders'))



    return "Invalid credentials", 401



@app.route('/signup', methods=['GET', 'POST'])

def signup():

    if request.method == 'POST':

        name = request.form['name']

        password = request.form['password']

        role = request.form['role']
        if role not in ['member', 'manager']:
            return "Invalid role", 400

        hashed_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

        cursor = mysql.connection.cursor()
        cursor.execute("INSERT INTO user1 (name, password, role) VALUES (%s, %s, %s)", (name,
hashed_password, role))
        mysql.connection.commit()
        cursor.close()

        return redirect(url_for('login'))  # Redirect to login page

    return render_template('signup.html')

@app.route('/equipment_search', methods=['GET', 'POST'])
def equipment_search():
    if request.method == 'POST':
        name = request.form['search_name']
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM Equipment WHERE name LIKE %s AND quantity > threshold",
(f"%{name}%",))
        equipment = cursor.fetchall()
        cursor.close()
        if not equipment:
            return "NO Stock"
        return render_template('equipment_search.html', equipment=equipment)

    return render_template('equipment_search.html')

@app.route('/request_usage', methods=['POST'])
def request_usage():
    equipment_id = request.form['equipment_id']
    room_requested = request.form['room_requested']
    requested_by = session['user_id']  # Assuming this will be the user_id of the member

    cursor = mysql.connection.cursor()
    cursor.execute("SELECT quantity, threshold FROM Equipment WHERE equipment_id = %s",
(equipment_id,))
    equipment = cursor.fetchone()

    if not equipment:
        return jsonify({"error": "Equipment not found"}), 404

    quantity, threshold = equipment
    if quantity <= threshold:
        cursor.execute("""
            INSERT INTO Orders (equipment_id, order_type, quantity_ordered, order_date, requested_by,
room_requested, status)
            VALUES (%s, 'New Order', %s, NOW(), %s, %s, 'Pending')
        """, (equipment_id, threshold - quantity + 1, requested_by, room_requested))
    else:
        cursor.execute("""
            INSERT INTO Orders (equipment_id, order_type, quantity_ordered, order_date, requested_by,
room_requested, status)
            VALUES (%s, 'Usage Request', 1, NOW(), %s, %s, 'Pending')
        """, (equipment_id, requested_by, room_requested))

    mysql.connection.commit()
    cursor.close()
    return redirect(url_for('equipment_search'))

@app.route('/view_orders')
def view_orders():
    if session.get('role') != 'manager':
        return "Access denied", 403

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute("""
        SELECT o.order_id, o.equipment_id, e.name AS equipment_name, o.quantity_ordered, o.order_date,
o.room_requested, o.approval_status
        FROM Orders o
        JOIN Equipment e ON o.equipment_id = e.equipment_id
        WHERE o.approval_status = 'Pending'
    """)
    orders = cursor.fetchall()
    cursor.close()
    return render_template('view_orders.html', orders=orders)
```

```
@app.route('/approve_order/<int:order_id>', methods=['POST'])
def approve_order(order_id):
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    # Retrieve the order to get the equipment ID and quantity ordered
    cursor.execute("SELECT equipment_id, quantity_ordered FROM Orders WHERE order_id = %s",
(order_id,))
    order = cursor.fetchone()

    if order:
        equipment_id = order['equipment_id']
        quantity_ordered = order['quantity_ordered']

        # Update the order's approval status
        cursor.execute("UPDATE Orders SET approval_status = 'Approved' WHERE order_id = %s",
(order_id,))

        # Deduct the quantity from the Equipment table
        cursor.execute("UPDATE Equipment SET quantity = quantity - %s WHERE equipment_id = %s",
(quantity_ordered, equipment_id))

        mysql.connection.commit()
        cursor.close()

        # Return success response
        return jsonify({"success": True})

    cursor.close()
    return jsonify({"success": False}), 400


@app.route('/reject_order/<int:order_id>', methods=['POST'])
def reject_order(order_id):
    cursor = mysql.connection.cursor()
    cursor.execute("UPDATE Orders SET approval_status = 'Rejected' WHERE order_id = %s",
(order_id,))
    mysql.connection.commit()
    cursor.close()


    # Respond with success message
    return jsonify({'success': True})


@app.route('/manager/orders')
def view_all_orders():
    cursor = mysql.connection.cursor()
    query = '''SELECT o.order_id, e.name AS equipment_name, o.quantity_ordered, o.order_date,
o.room_requested, o.approval_status
        FROM Orders o
        JOIN Equipment e ON o.equipment_id = e.equipment_id'''
    cursor.execute(query)
    orders = cursor.fetchall()
    cursor.close()
    return render_template('all_orders.html', orders=orders)


# Route to display orders that require placement with recommended dealers
@app.route('/manager/orders_to_place')
```

```
def view_orders_to_place():
    cursor = mysql.connection.cursor()
    query = '''SELECT
            o.order_id, o.order_date, d.name, d.contact_info, d.address, e.name
            FROM Orders o
            LEFT JOIN Dealers d ON o.equipment_id = d.dealer_id
            LEFT JOIN Equipment e ON o.equipment_id = e.equipment_id
            WHERE o.approval_status = 'Pending' && e.threshold > e.quantity'''
    cursor.execute(query)
    orders_to_place = cursor.fetchall()
    print(orders_to_place)
    cursor.close()
    return render_template('orders_to_place.html', orders=orders_to_place)
if __name__ == "__main__":
    app.run(debug=True)
```

CODE FOR APP BUILDING

```
CREATE TABLE Equipment (
    equipment_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    quantity INT NOT NULL,
    threshold INT NOT NULL,
    condition1 VARCHAR(50) NOT NULL,
    status VARCHAR(50),
    last_serviced_date DATE
);
```

```
CREATE TABLE user1 (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('member', 'manager') NOT NULL
);
-- Table: Orders
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    equipment_id INT,
    order_type ENUM('New Order', 'Usage Request') NOT NULL,
    quantity_ordered INT NOT NULL,
    order_date DATE NOT NULL,
    requested_by INT NOT NULL,
    status ENUM('Pending', 'Approved', 'Rejected') DEFAULT 'Pending',
    approval_date DATE,
    dealer_id INT,
    room_requested VARCHAR(255),
    FOREIGN KEY (equipment_id) REFERENCES Equipment(equipment_id),
    FOREIGN KEY (dealer_id) REFERENCES Dealers(dealer_id)
);
-- Table: Service
CREATE TABLE Service (
    service_id INT AUTO_INCREMENT PRIMARY KEY,
    equipment_id INT,
    service_date DATE NOT NULL,
    issue_description TEXT,
    status ENUM('Pending', 'In Progress', 'Completed') DEFAULT 'Pending',
    technician_assigned VARCHAR(255),
    completion_date DATE,
    FOREIGN KEY (equipment_id) REFERENCES Equipment(equipment_id)
);
-- Table: Dealers
CREATE TABLE Dealers (
    dealer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    contact_info VARCHAR(255),
    address TEXT,
    equipment_types TEXT
);
-- Table: Management_Team
CREATE TABLE Management_Team (
    member_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    role ENUM('Hospital Member', 'Manager') NOT NULL,
    contact_info VARCHAR(255),
    last_login DATETIME,
    actions_taken TEXT
);
USE hospital_inventory;
INSERT INTO Equipment (name, quantity, threshold, condition1, status, last_serviced_date)
VALUES
('Heart Monitor', 10, 2, 'Good', 'Operational', '2024-10-10'),
('Ultrasound Machine', 3, 1, 'Fair', 'In Service', '2024-09-20'),
('ECG Machine', 5, 1, 'Excellent', 'Operational', '2024-08-15'),
('X-Ray Machine', 2, 1, 'Poor', 'Needs Repair', '2024-07-30'),
('Defibrillator', 8, 3, 'Good', 'Operational', '2024-10-01'),
('Ventilator', 4, 2, 'Fair', 'Operational', '2024-09-25'),
('Blood Pressure Monitor', 15, 5, 'Good', 'Operational', '2024-10-05'),
('Surgical Light', 6, 2, 'Excellent', 'Operational', '2024-09-10'),
('Infusion Pump', 7, 3, 'Good', 'Operational', '2024-10-12'),
('Oxygen Concentrator', 5, 2, 'Fair', 'Operational', '2024-09-22');
INSERT INTO Dealers (dealer_id, name, contact_info, address) VALUES
(1, 'Medical Supply Co.', '123-456-7890', '123 Medical St, City, State, 12345'),
(2, 'Healthcare Solutions', '234-567-8901', '456 Healthcare Ave, City, State, 23456'),
(3, 'Global Medical Supplies', '345-678-9012', '789 Global Rd, City, State, 34567'),
(4, 'Innovative Health Supplies', '456-789-0123', '321 Innovative Blvd, City, State, 45678'),
(5, 'Vital Equipment Providers', '567-890-1234', '654 Vital Way, City, State, 56789');
INSERT INTO Service (equipment_id, service_date, issue_description, status, technician_assigned,
completion_date)
VALUES
(1, '2024-10-15', 'Routine maintenance', 'Completed', 'John Doe', '2024-10-16'),
(2, '2024-10-20', 'Repair broken display', 'In Progress', 'Jane Smith', NULL),
(3, '2024-10-25', 'Battery replacement', 'Pending', 'Mike Johnson', NULL),
(4, '2024-10-28', 'Calibration issue', 'Completed', 'Emily Davis', '2024-10-29'),
(5, '2024-11-01', 'Software update required', 'Pending', NULL, NULL);
INSERT INTO user1 (name, password, role) VALUES ('John Doe', 'password123', 'member');
INSERT INTO user1 (name, password, role) VALUES ('Jane Smith', 'mypassword', 'manager');
INSERT INTO Users (name, password, role) VALUES ('Alice Johnson', 'alicepass', 'member');
ALTER TABLE Orders
ADD COLUMN approval_status ENUM('Pending', 'Approved', 'Rejected') DEFAULT 'Pending',
ADD COLUMN description TEXT;

SELECT * FROM orders;

SELECT * FROM user1;

INSERT INTO Equipment (name, quantity, threshold, condition1, status, last_serviced_date)

VALUES

('pulse', 1, 2, 'Good', 'Operational', '2024-10-10');
update equipment set quantity = 5 where equipment_id = 1;
```

CODE FOR DESIGNING INDEX INTERFACE

```
/* General Reset */
*,
*::before,
*::after {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Basic Styles */
body {
    font-family: Arial, sans-serif;
    background-color: #f2f6fc;
    display: flex;
    flex-direction: column;
    align-items: center;
    min-height: 100vh;
    margin: 0;
    padding: 20px;
    color: #333;
}

h1, h2 {
    color: #1d3a73;
    text-align: center;
    margin: 20px 0;
}

h1 {
    font-size: 2.5rem;
    text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.1);
}

h2 {
    font-size: 2rem;
}

/* Table Styling */
table {
    width: 100%;
    max-width: 800px;
    border-collapse: collapse;
    background-color: #ffffff;
    margin: 20px auto;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

th, td {
    padding: 12px 15px;
    text-align: left;
    font-size: 0.95rem;
}

th {
    background-color: #1d3a73;
    color: #ffffff;
    font-weight: bold;
    text-transform: uppercase;
    letter-spacing: 0.03em;
}

td {
    color: #555;
    border-bottom: 1px solid #ddd;
}

tr:nth-child(even) {
    background-color: #f9fafc;
}

tr:hover {
    background-color: #eaf1fd;
    cursor: pointer;
}

/* Form Styling */
form {
    width: 100%;
    max-width: 500px;
    background-color: #ffffff;
    padding: 20px;
    border-radius: 12px;
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    text-align: center;
    margin-bottom: 20px;
    transition: transform 0.3s ease;
}

form:hover {
    transform: scale(1.01);
}

input[type="text"],
input[type="hidden"],
select,
button {
    width: 100%;
    padding: 12px;
    margin: 10px 0;
    border-radius: 5px;
    font-size: 1rem;
    border: 1px solid #ddd;
    transition: border-color 0.3s, box-shadow 0.3s;
}

input[type="text"]:focus,
select:focus {
    border-color: #1d3a73;
    outline: none;
    box-shadow: 0 0 8px rgba(29, 58, 115, 0.3);
}

/* Button Styling */
button {
    background-color: #1d3a73;
    color: #ffffff;
    font-weight: bold;
    cursor: pointer;
    border: none;
    transition: background-color 0.3s, transform 0.2s, box-shadow 0.3s;
}

button:hover {
    background-color: #335a9c;
    transform: scale(1.02);
    box-shadow: 0 4px 8px rgba(51, 90, 156, 0.3);
}

button:active {
    transform: scale(0.98);
```

```
    box-shadow: none;
}

/* Order Container Styling */
.order-container {
    width: 90%;
    max-width: 800px;
    background-color: #ffffff;
    padding: 20px;
    margin: 15px 0;
    border-radius: 8px;
    box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);
    border-left: 5px solid #3498db;
}

.order-container p {
    font-size: 1em;
    color: #555;
    margin: 8px 0;
}

.order-container p strong {
    color: #2c3e50;
}

/* Search Result Item */
.search-result {
    display: flex;
    flex-direction: column;
    gap: 10px;
    padding: 15px;
    margin: 10px;
    background-color: #ffffff;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
```

```
}

.search-result form {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.search-result button {
    align-self: stretch;
}

/* Tooltip Styling */
.tooltip {
    position: relative;
    display: inline-block;
    cursor: pointer;
    margin-left: 5px;
    color: #1d3a73;
    font-weight: bold;
}

.tooltip .tooltiptext {
    visibility: hidden;
    width: 140px;
    background-color: #555;
    color: #fff;
    text-align: center;
    padding: 5px;
    border-radius: 5px;
    position: absolute;
    bottom: 125%;
    left: 50%;
    transform: translateX(-50%);
    opacity: 0;
    transition: opacity 0.3s;
```

```
}

.tooltip:hover .tooltiptext {
   visibility: visible;
   opacity: 1;
}

/* Keyframes for Animations */
@keyframes fadeIn {
   from { opacity: 0; }
   to { opacity: 1; }
}

/* Responsive Adjustments */
@media (max-width: 768px) {
   h1, h2 {
      font-size: 1.8rem;
   }

   form, .order-container, table {
      width: 95%;
   }

   button {
      font-size: 0.85em;
      padding: 8px 12px;
   }
}

@media (max-width: 500px) {
   h1 {
      font-size: 1.5rem;
   }

   th, td {
      font-size: 0.85rem;
```

```
        padding: 8px;
    }

    input[type="text"], select, button {
        font-size: 0.85rem;
        padding: 10px;
    }
}
```

CODE FOR VIEWING MAIN FILE

```
/* General Reset */
*,
*::before,
*::after {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Basic Styles */
body {
    font-family: Arial, sans-serif;
    background-color: #f2f6fc;
    display: flex;
    flex-direction: column;
    align-items: center;
    min-height: 100vh;
    margin: 0;
    padding: 20px;
    color: #333;
}

h1, h2 {
    color: #1d3a73;
    text-align: center;
    margin: 20px 0;
}

h1 {
    font-size: 2.5rem;
    text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.1);
}

h2 {
    font-size: 2rem;
}

/* Table Styling */
table {
    width: 100%;
    max-width: 800px;
    border-collapse: collapse;
    background-color: #ffffff;
    margin: 20px auto;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

th, td {
    padding: 12px 15px;
    text-align: left;
    font-size: 0.95rem;
}

th {
    background-color: #1d3a73;
    color: #ffffff;
    font-weight: bold;
    text-transform: uppercase;
    letter-spacing: 0.03em;
```

```
}

td {
   color: #555;
   border-bottom: 1px solid #ddd;
}

tr:nth-child(even) {
   background-color: #f9fafc;
}

tr:hover {
   background-color: #eaf1fd;
   cursor: pointer;
}

/* Form Styling */
form {
   width: 100%;
   max-width: 500px;
   background-color: #ffffff;
   padding: 20px;
   border-radius: 12px;
   box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
   text-align: center;
   margin-bottom: 20px;
   transition: transform 0.3s ease;
}

form:hover {
   transform: scale(1.01);
}

input[type="text"],
input[type="hidden"],
select,
button {
   width: 100%;
   padding: 12px;
   margin: 10px 0;
   border-radius: 5px;
   font-size: 1rem;
   border: 1px solid #ddd;
   transition: border-color 0.3s, box-shadow 0.3s;
}

input[type="text"]:focus,
select:focus {
   border-color: #1d3a73;
   outline: none;
   box-shadow: 0 0 8px rgba(29, 58, 115, 0.3);
}

/* Button Styling */
button {
   background-color: #1d3a73;
   color: #ffffff;
   font-weight: bold;
   cursor: pointer;
   border: none;
   transition: background-color 0.3s, transform 0.2s, box-shadow 0.3s;
}

button:hover {
   background-color: #335a9c;
   transform: scale(1.02);
   box-shadow: 0 4px 8px rgba(51, 90, 156, 0.3);
}

button:active {
   transform: scale(0.98);
   box-shadow: none;
}

/* Order Container Styling */
.order-container {
   width: 90%;
   max-width: 800px;
   background-color: #ffffff;
   padding: 20px;
   margin: 15px 0;
   border-radius: 8px;
   box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);
   border-left: 5px solid #3498db;
}

.order-container p {
   font-size: 1em;
   color: #555;
   margin: 8px 0;
}

.order-container p strong {
   color: #2c3e50;
}

/* Search Result Item */
.search-result {
   display: flex;
   flex-direction: column;
   gap: 10px;
   padding: 15px;
   margin: 10px;
   background-color: #ffffff;
   border: 1px solid #ddd;
   border-radius: 8px;
   box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
}
```

```
.search-result form {
   display: flex;
   flex-direction: column;
   gap: 10px;
}

.search-result button {
   align-self: stretch;
}

/* Tooltip Styling */
.tooltip {
   position: relative;
   display: inline-block;
   cursor: pointer;
   margin-left: 5px;
   color: #1d3a73;
   font-weight: bold;
}

.tooltip .tooltiptext {
   visibility: hidden;
   width: 140px;
   background-color: #555;
   color: #fff;
   text-align: center;
   padding: 5px;
   border-radius: 5px;
   position: absolute;
   bottom: 125%;
   left: 50%;
   transform: translateX(-50%);
   opacity: 0;
   transition: opacity 0.3s;
}
```

```
.tooltip:hover .tooltiptext {
   visibility: visible;
   opacity: 1;
}

/* Keyframes for Animations */
@keyframes fadeIn {
   from { opacity: 0; }
   to { opacity: 1; }
}

/* Responsive Adjustments */
@media (max-width: 768px) {
   h1, h2 {
      font-size: 1.8rem;
   }

   form, .order-container, table {
      width: 95%;
   }

   button {
      font-size: 0.85em;
      padding: 8px 12px;
   }
}

@media (max-width: 500px) {
   h1 {
      font-size: 1.5rem;
   }

   th, td {
      font-size: 0.85rem;
      padding: 8px;
```

```
    }

    input[type="text"], select, button {
        font-size: 0.85rem;
        padding: 10px;
    }
}
```

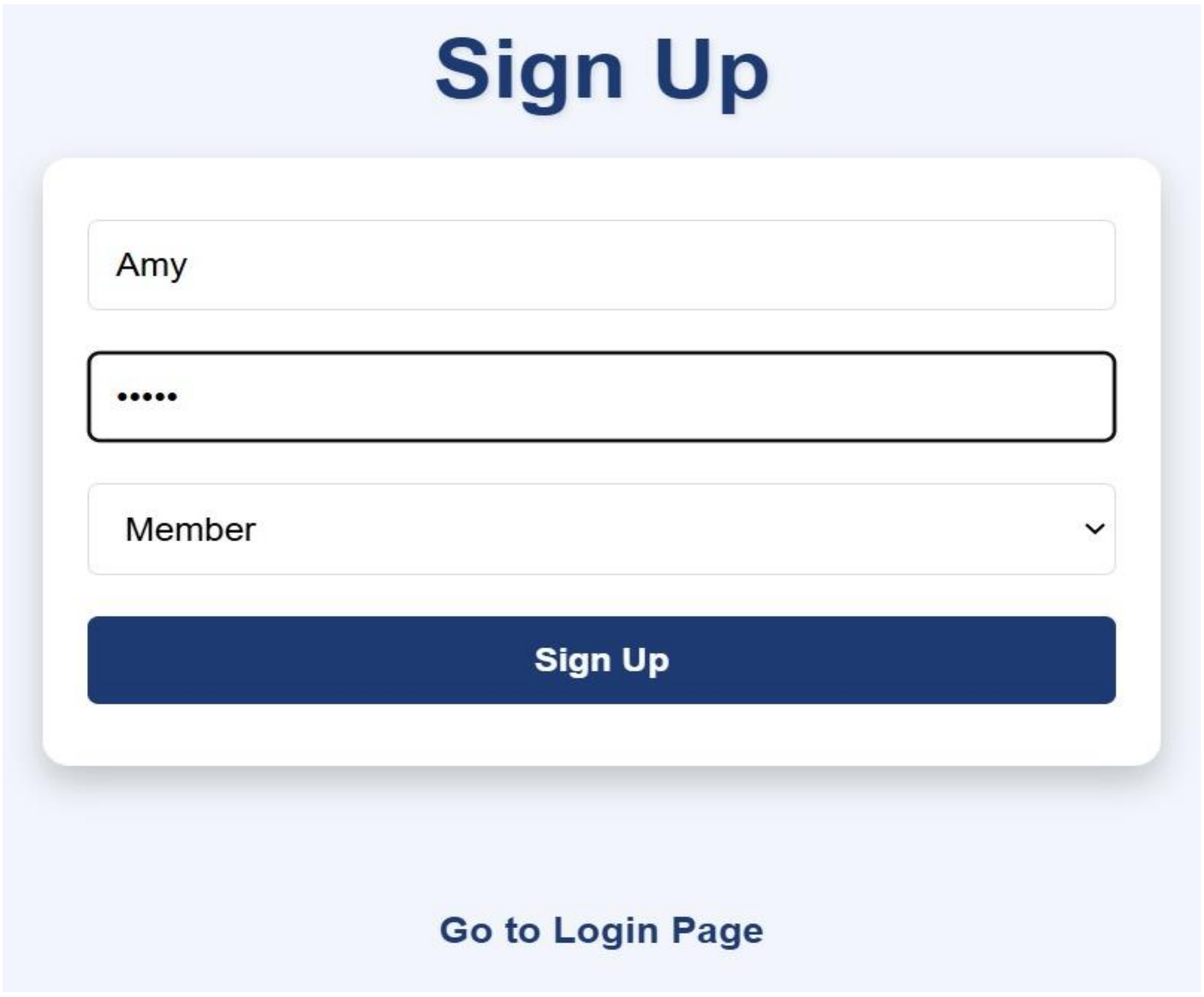
CODE FOR JAVASCRIPT

```
// scripts.js

document.getElementById("search-form").onsubmit = function (e) {
    e.preventDefault();
    let name = document.getElementById("search-name").value;

    fetch(`/search_equipment?name=${name}`)
        .then(response => response.json())
        .then(data => {
            let resultDiv = document.getElementById("search-results");
            if (data === "NO Stock") {
                // If "NO Stock" is returned, show the message
                resultDiv.innerHTML = "<p>No equipment in stock.</p>";
            }
            else if (data.length > 0) {
                resultDiv.innerHTML = data.map(item => `
                    <p><strong>ID:</strong> ${item.equipment_id}, <strong>Name:</strong> ${item.name},
<strong>Quantity:</strong> ${item.quantity}</p>
                    <form action="/request_usage" method="POST">
                        <input type="hidden" name="equipment_id" value="${item.equipment_id}">
                        <input type="text" name="room_requested" placeholder="Room/Patient ID" required>
                        <button type="submit">Request Usage</button>
                    </form>
                    <hr>
                `).join("");
            } else {
                resultDiv.innerHTML = "<p>No equipment found.</p>";
            }
        })
        .catch(error => {
            console.error("Error fetching equipment:", error);
        });
};
```

```
document.getElementById("request-form").onsubmit = function (e) {
    e.preventDefault();
    let equipment_id = document.getElementById("request-equipment-id").value;
    let room_requested = document.getElementById("request-room").value;

    fetch('/request_use', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ equipment_id: equipment_id, room_requested: room_requested,
requested_by: 1 })
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById("request-result").innerText = data.message;
    });
};

function requestEquipmentUse(equipmentId, roomRequested, requestedBy) {
    fetch('/request_use', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            equipment_id: equipmentId,
            room_requested: roomRequested,
            requested_by: requestedBy
        })
    })
    .then(response => response.json())
    .then(data => {
        // Display confirmation message
        alert(data.message);
    })
    .catch(error => {
        console.error('Error:', error);
    });
}

document.getElementById("add-equipment-form").onsubmit = function (e) {
    e.preventDefault();
    let name = document.getElementById("equipment-name").value;
    let quantity = document.getElementById("equipment-quantity").value;
    let threshold = document.getElementById("equipment-threshold").value;
    let condition = document.getElementById("equipment-condition").value;

    fetch('/add_equipment', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ name: name, quantity: quantity, threshold: threshold, condition: condition })
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById("add-result").innerText = data.message;
    });
};
```

PROJECT SCREENSHOTS

Sign UP – Member

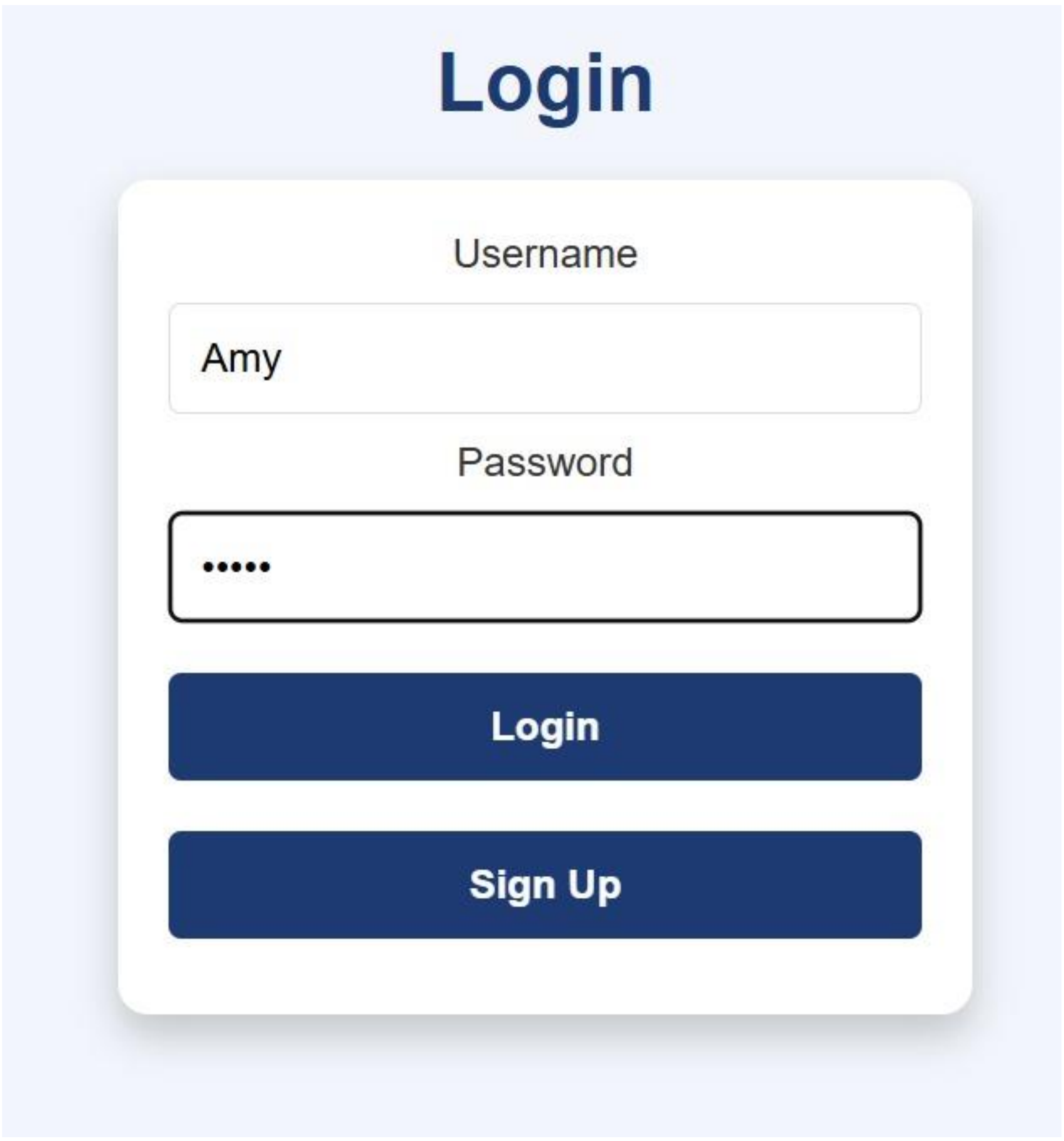

Sign Up

Amy

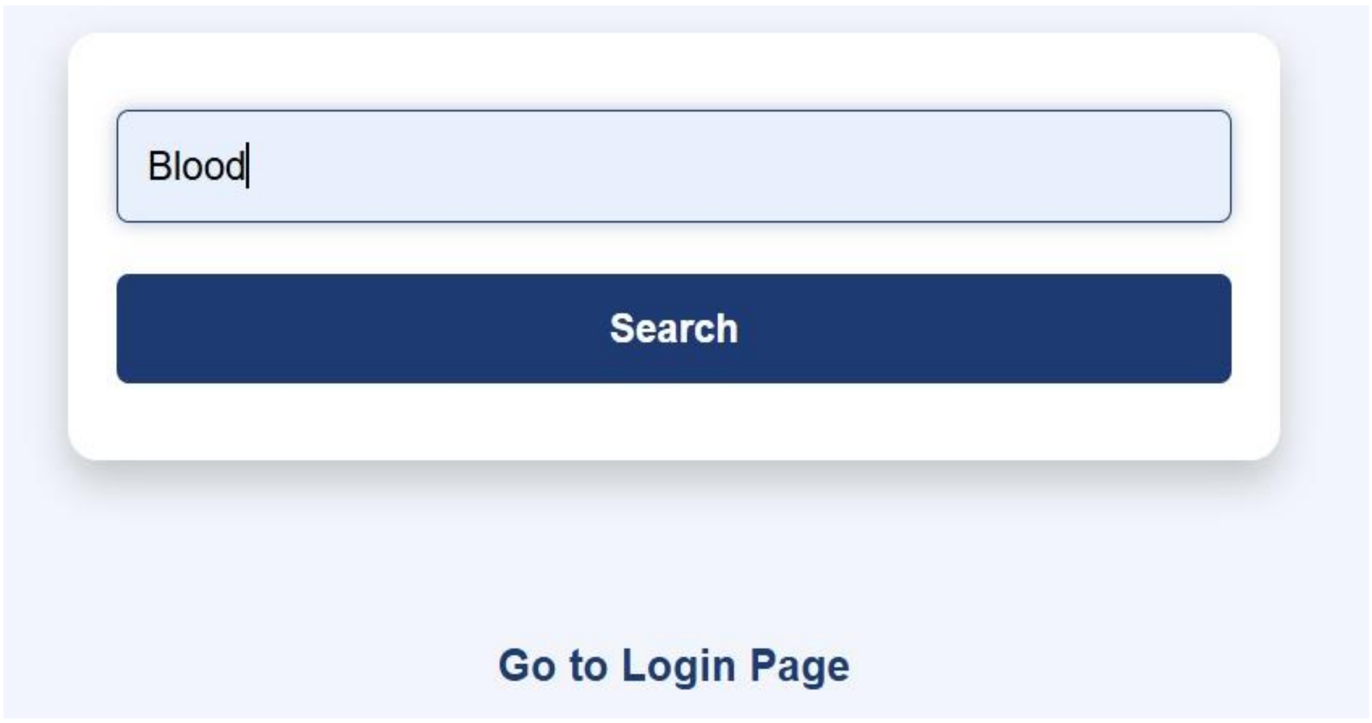
•••••

Member

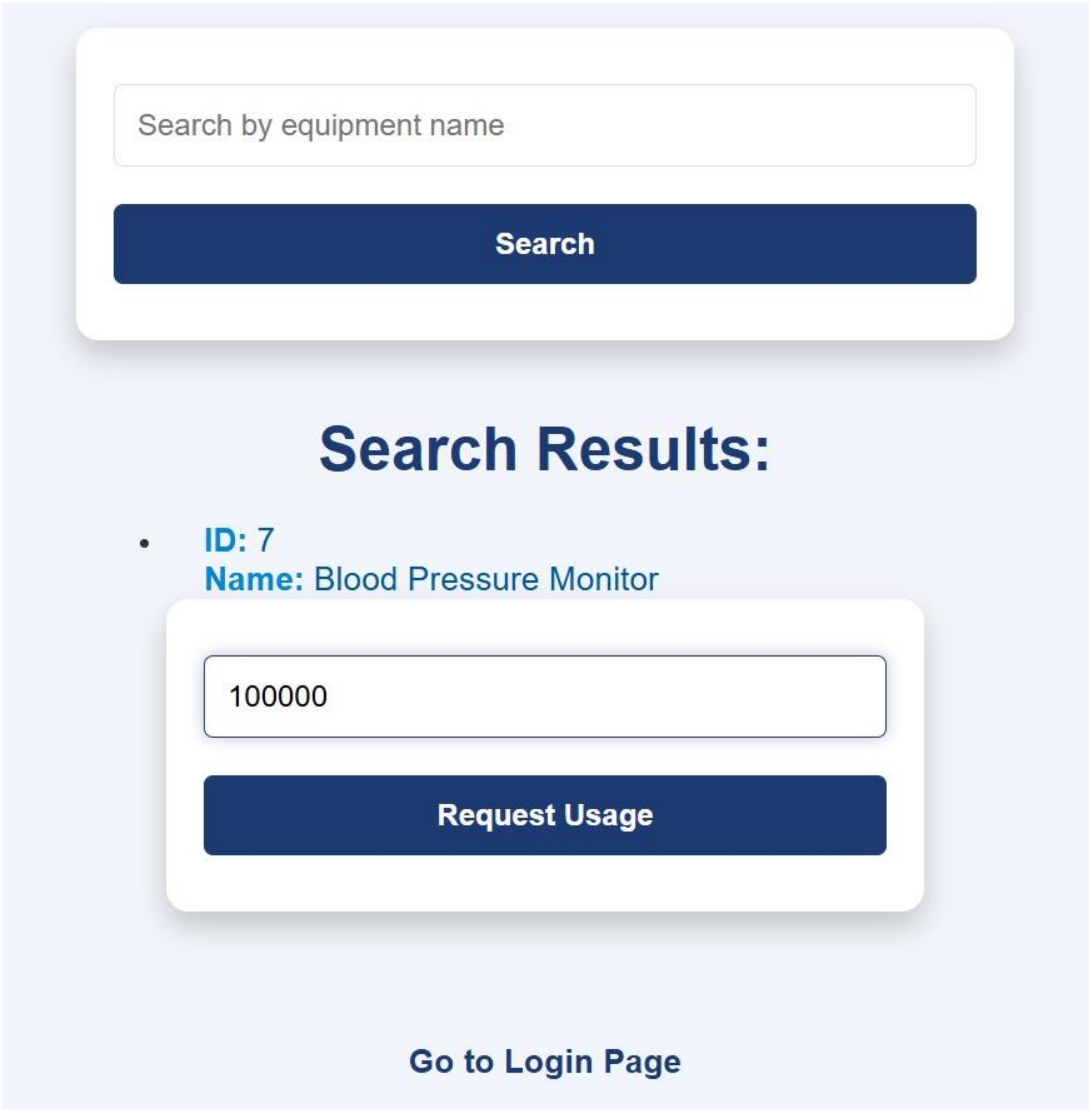
Login Page - Member



Equipment search Page



Request for Equipment usage

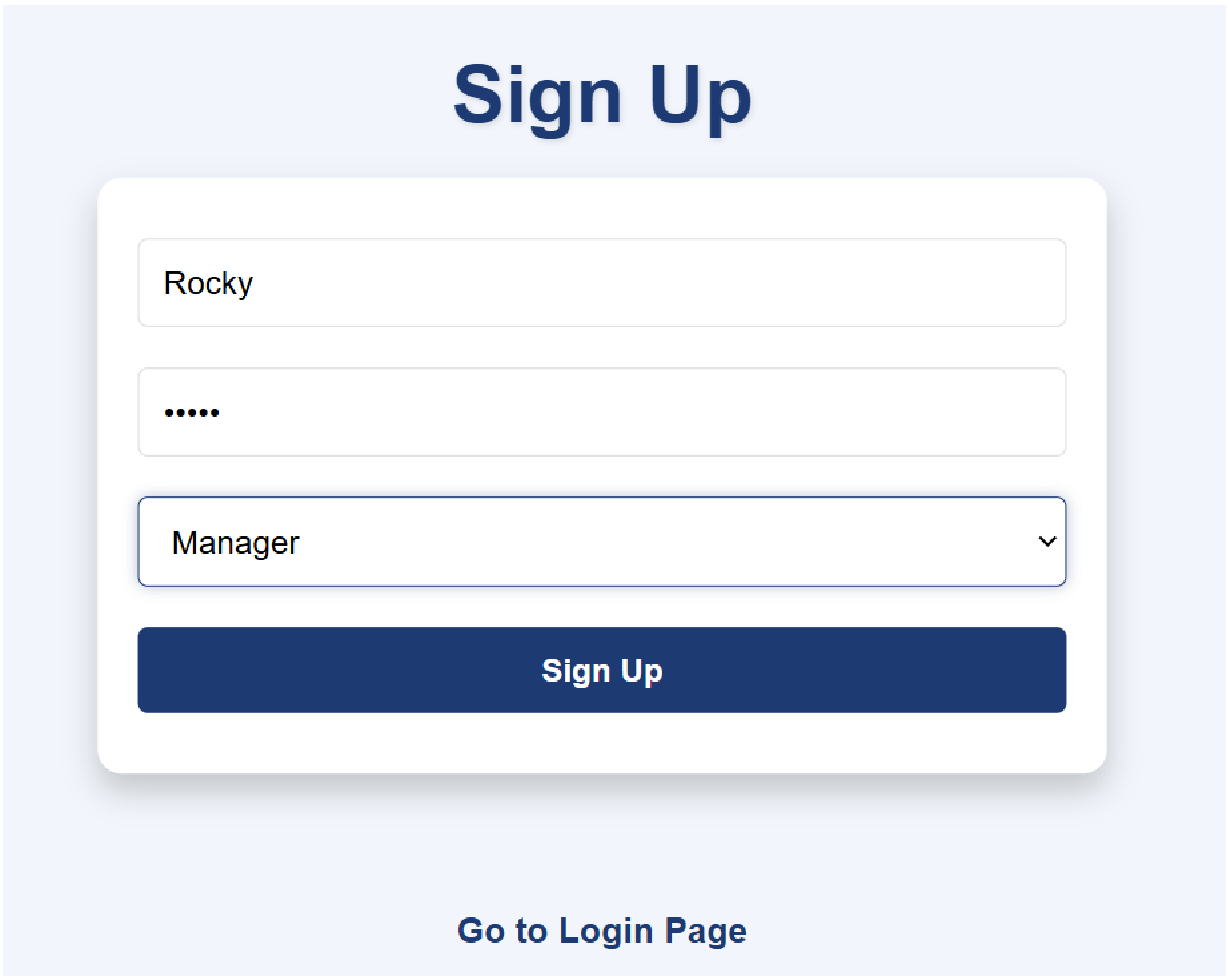


Search Results:

- **ID:** 7
  Name: Blood Pressure Monitor

100000

Sign up – Manager



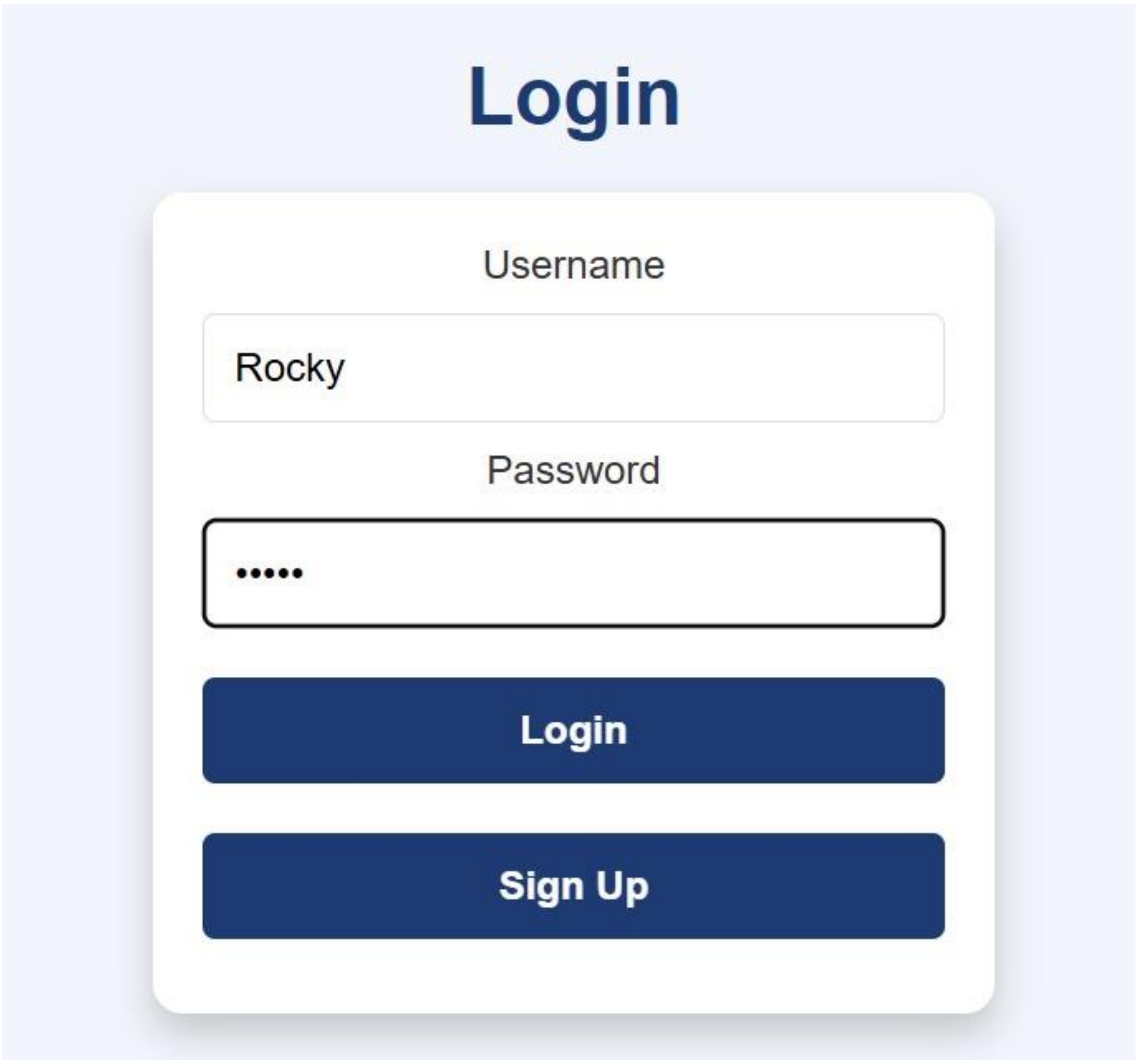
Sign Up

Rocky

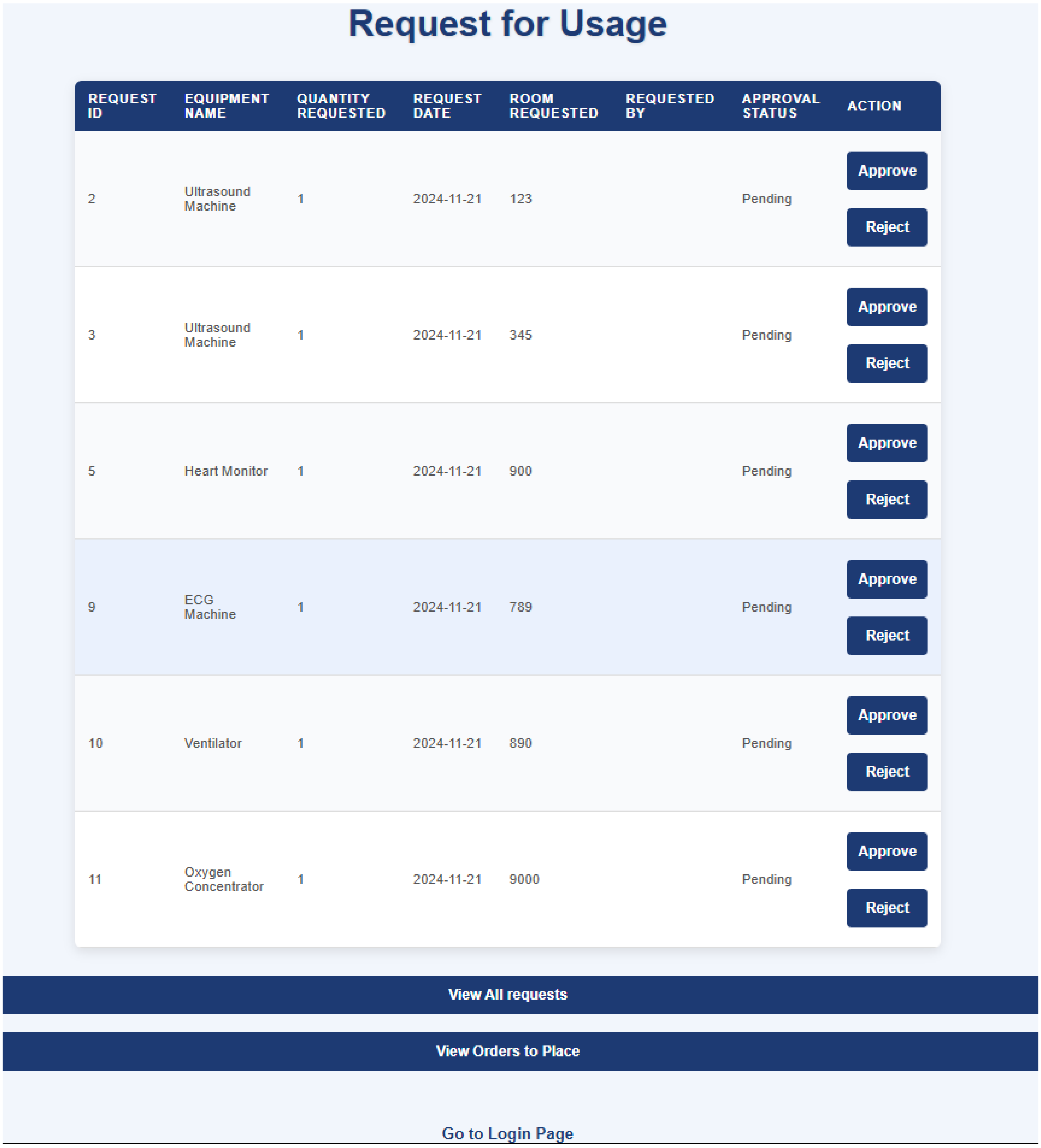
•••••

Manager

Login Page - Manager



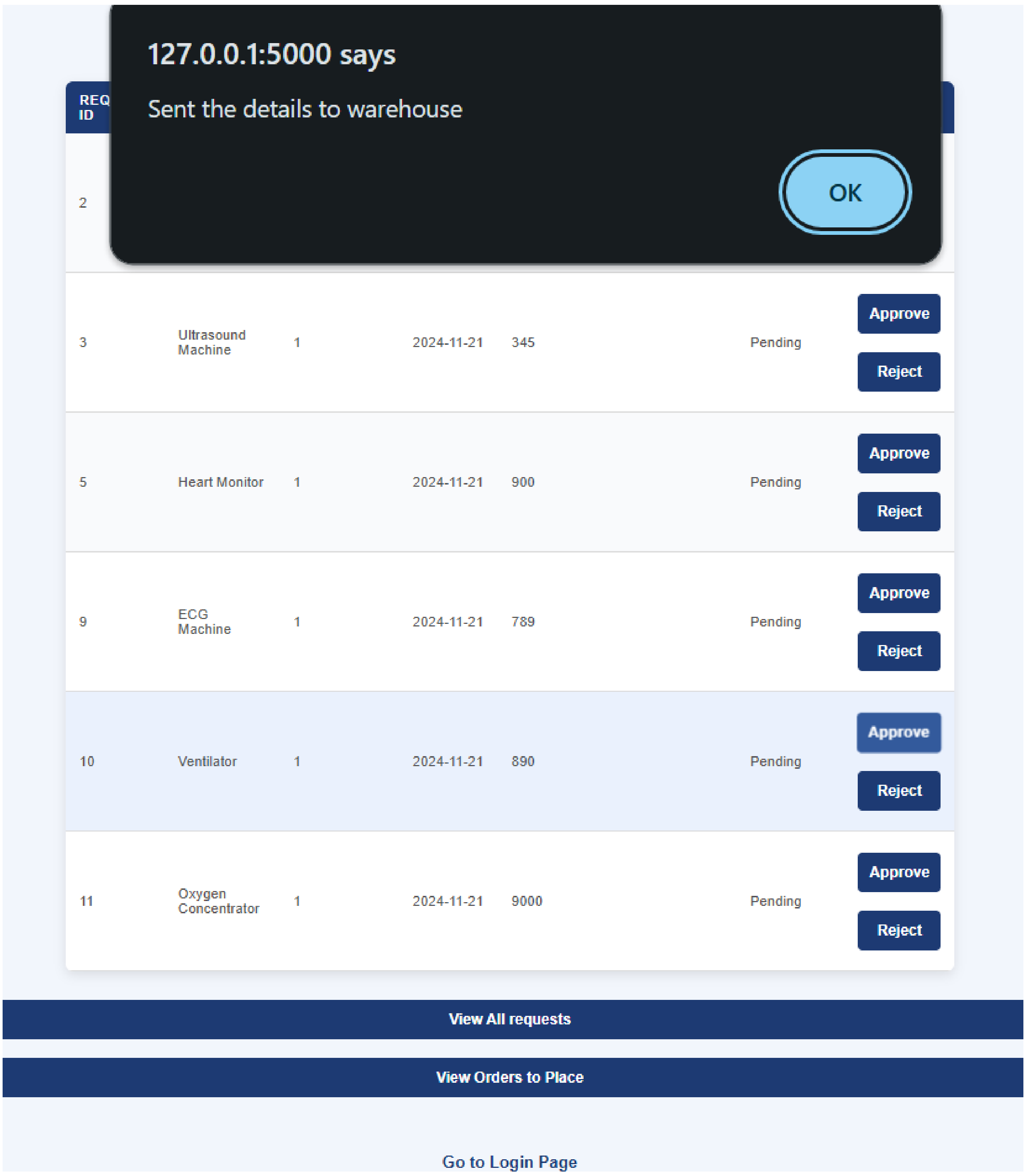
View the Requests



Request for Usage

REQUEST ID	EQUIPMENT NAME	QUANTITY REQUESTED	REQUEST DATE	ROOM REQUESTED	REQUESTED BY	APPROVAL STATUS	ACTION
2	Ultrasound Machine	1	2024-11-21	123		Pending	Approve Reject
3	Ultrasound Machine	1	2024-11-21	345		Pending	Approve Reject
5	Heart Monitor	1	2024-11-21	900		Pending	Approve Reject
9	ECG Machine	1	2024-11-21	789		Pending	Approve Reject
10	Ventilator	1	2024-11-21	890		Pending	Approve Reject
11	Oxygen Concentrator	1	2024-11-21	9000		Pending	Approve Reject

Approved the Request

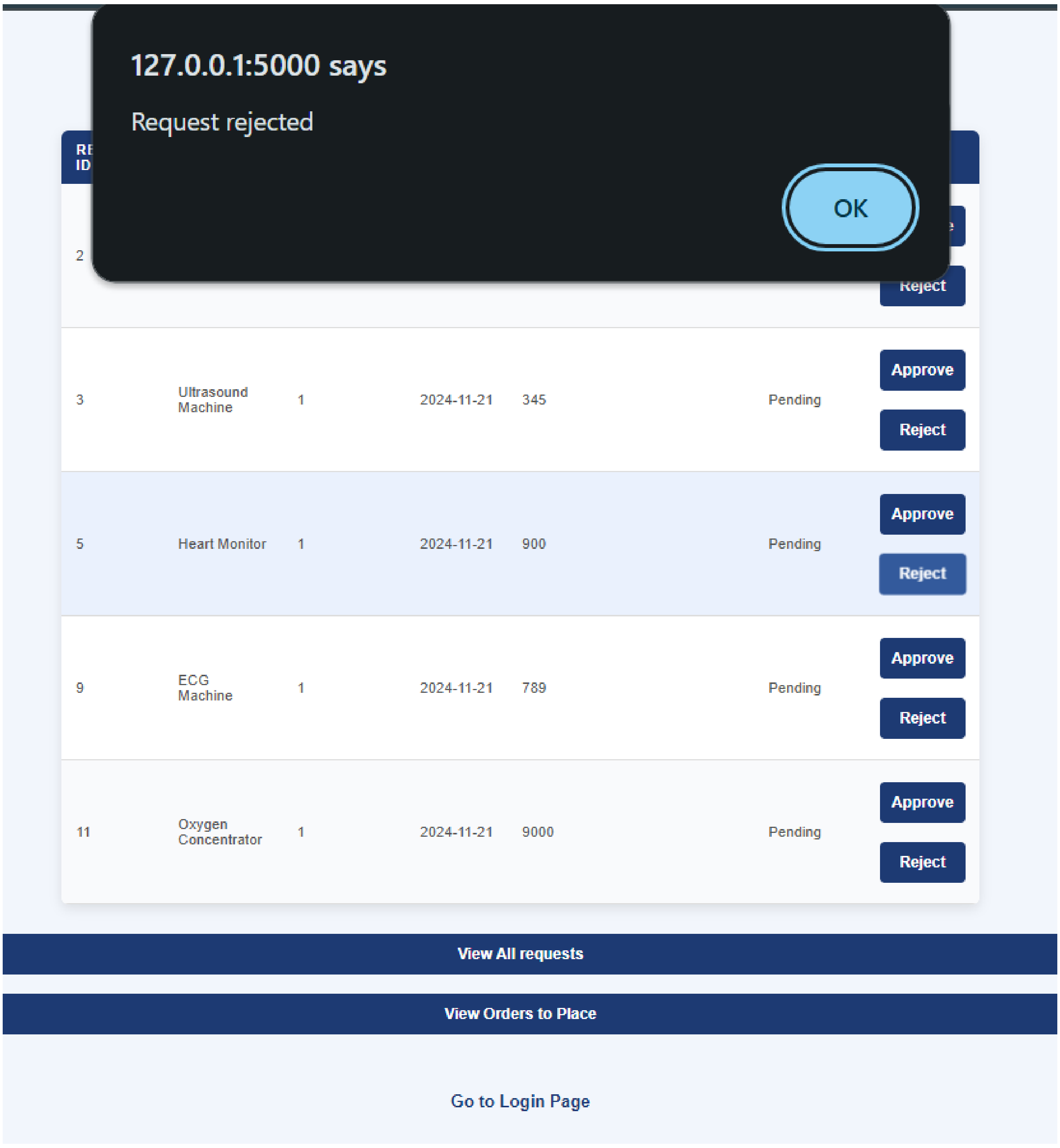


127.0.0.1:5000 says

Sent the details to warehouse

REQ ID						
2						
3	Ultrasound Machine	1	2024-11-21	345	Pending	Approve Reject
5	Heart Monitor	1	2024-11-21	900	Pending	Approve Reject
9	ECG Machine	1	2024-11-21	789	Pending	Approve Reject
10	Ventilator	1	2024-11-21	890	Pending	Approve Reject
11	Oxygen Concentrator	1	2024-11-21	9000	Pending	Approve Reject

Rejected the Request

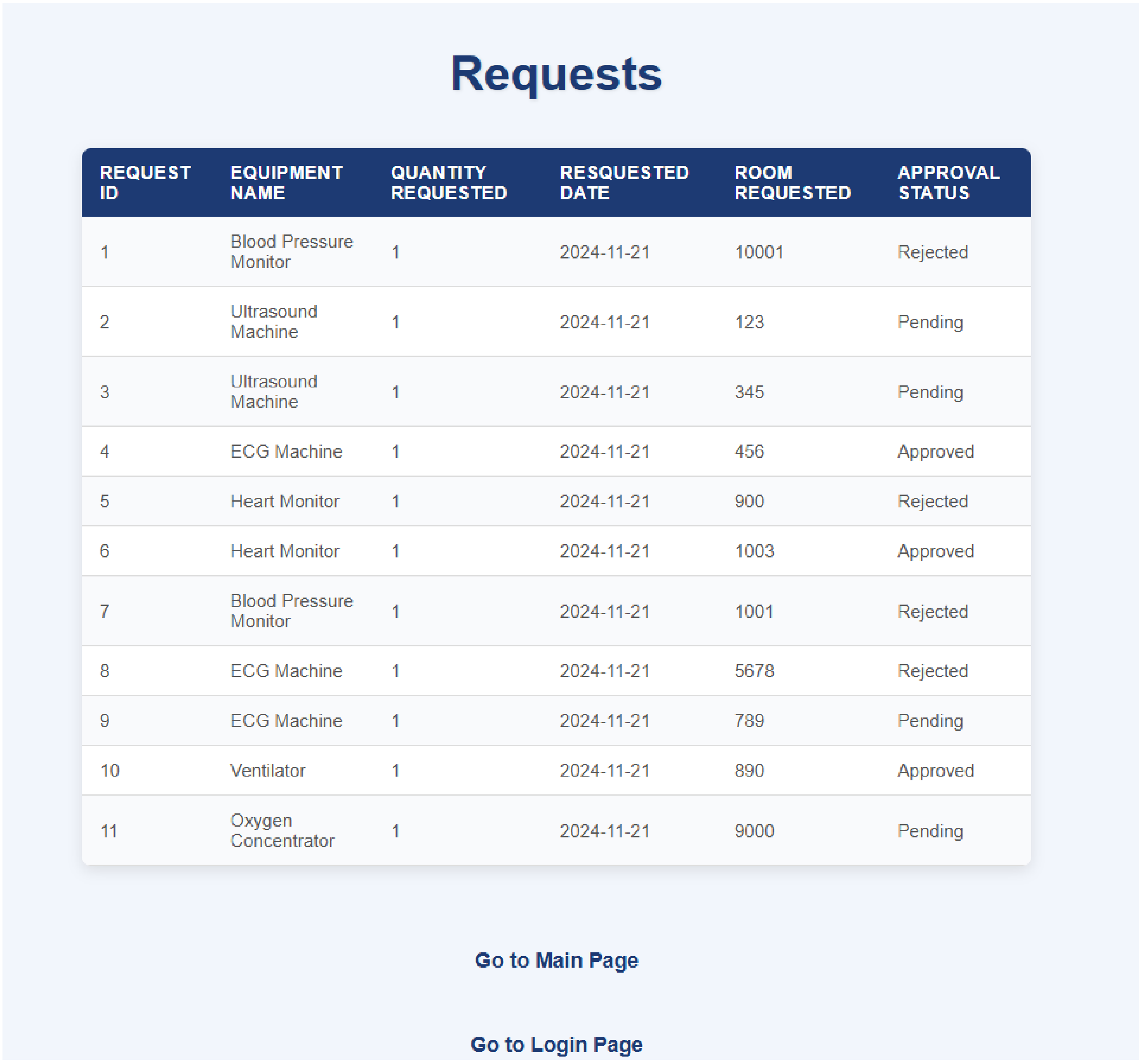


127.0.0.1:5000 says

Request rejected

RE ID						
2						Reject
3	Ultrasound Machine	1	2024-11-21	345	Pending	Approve Reject
5	Heart Monitor	1	2024-11-21	900	Pending	Approve Reject
9	ECG Machine	1	2024-11-21	789	Pending	Approve Reject
11	Oxygen Concentrator	1	2024-11-21	9000	Pending	Approve Reject

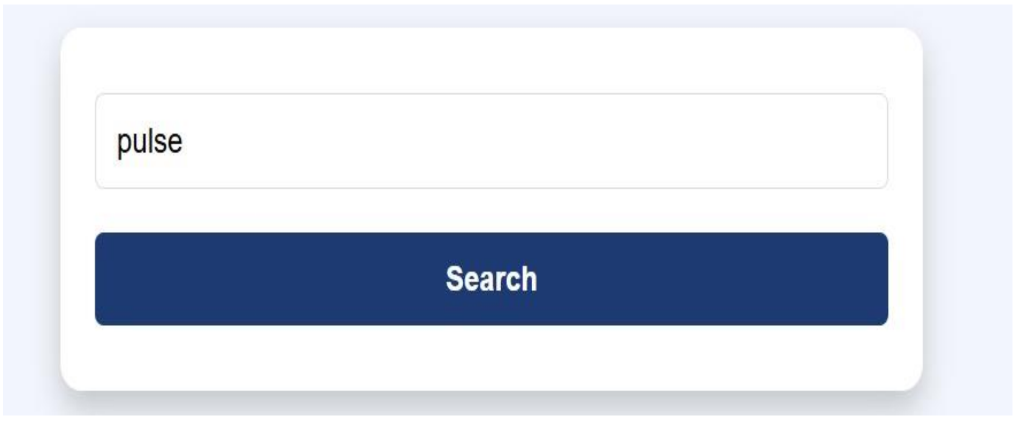
View Request History

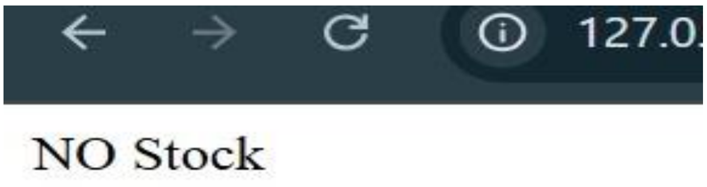


Requests

REQUEST ID	EQUIPMENT NAME	QUANTITY REQUESTED	RESQUESTED DATE	ROOM REQUESTED	APPROVAL STATUS
1	Blood Pressure Monitor	1	2024-11-21	10001	Rejected
2	Ultrasound Machine	1	2024-11-21	123	Pending
3	Ultrasound Machine	1	2024-11-21	345	Pending
4	ECG Machine	1	2024-11-21	456	Approved
5	Heart Monitor	1	2024-11-21	900	Rejected
6	Heart Monitor	1	2024-11-21	1003	Approved
7	Blood Pressure Monitor	1	2024-11-21	1001	Rejected
8	ECG Machine	1	2024-11-21	5678	Rejected
9	ECG Machine	1	2024-11-21	789	Pending
10	Ventilator	1	2024-11-21	890	Approved
11	Oxygen Concentrator	1	2024-11-21	9000	Pending

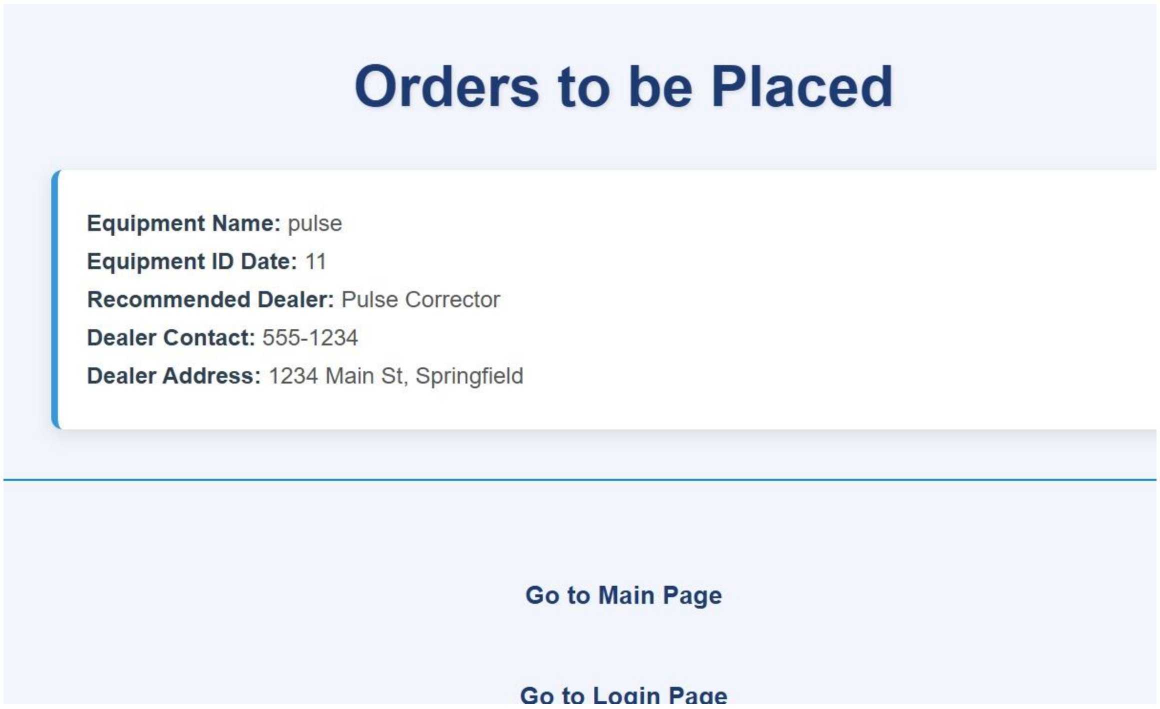
Case of having Less stock:





NO Stock

View the orders to be Placed



Orders to be Placed

Equipment Name: pulse
Equipment ID Date: 11
Recommended Dealer: Pulse Corrector
Dealer Contact: 555-1234
Dealer Address: 1234 Main St, Springfield

RESULTS AND DISCUSSION

5.1 OBSERVATIONS

- The system successfully allows staff to submit maintenance requests for medical equipment, while administrators can manage and approve these requests through an intuitive interface.
- Real-time notifications and status tracking enhance communication and transparency between staff and administrators, improving response times.
- The user-friendly GUI provides a seamless experience for both equipment users and maintenance managers.
- The application's integrated calendar feature simplifies scheduling for equipment maintenance, reducing manual errors.
- Administrators can quickly review and approve or decline requests, improving efficiency in maintenance management.

5.2 LIMITATIONS

- The application depends on stable internet connectivity for real-time notifications and smooth database functionality.
- Occasional delays in notifications may occur due to server or network issues.
- User experience can vary across devices or browsers, necessitating optimization for maximum compatibility.
- The system may face minor scalability challenges during peak periods of maintenance requests, depending on the implementation.

5.3 FUTURE IMPROVEMENTS

- Mobile App Integration: Develop a mobile app for on-the-go access, allowing medical staff and administrators to manage maintenance requests, approvals, and notifications conveniently.
- Cloud Database: Implement a cloud-based solution for data storage to improve scalability, security, and accessibility across different locations and platforms.
- Advanced Notification System: Integrate more advanced notification features, such as email alerts, SMS, and push notifications, ensuring timely communication and responsiveness.
- Additional Request Categories: Expand the management system to accommodate various types of requests beyond maintenance, such as equipment relocation and sanitization.
- AI-Powered Analytics: Implement analytics tools to help administrators track equipment usage and identify patterns, supporting better decision-making on equipment maintenance schedules and replacement policies.
- QR Code and RFID Integration: Enable quick scanning capabilities for equipment tracking and

identification, improving accuracy in equipment management and location tracking.

ADDITIONAL ANALYTICS

- Feedback from initial users indicates high satisfaction with the ease of use and reliability of the system. The streamlined workflow reduces administrative burden, allowing staff to focus more on patient care and less on maintenance requests. Staff appreciate the transparency and the ability to track request statuses in real time, which reduces anxiety and uncertainty about equipment availability.
- The project addresses many challenges associated with traditional equipment management processes. With future improvements, the system could become an indispensable tool for healthcare institutions, providing a seamless, efficient, and user-friendly solution for managing equipment and maintenance requests.

TESTING

6.1 UNIT TESTING

Unit testing in this project focuses on verifying the functionality of individual components, such as the maintenance request form, the administrator approval panel, the notification system, and database interactions. Each module is tested independently to confirm that it performs its specific function correctly. For instance, testing includes form validation, date selection logic, and input validation to ensure proper functionality.

6.2 INTEGRATION TESTING

Integration testing ensures that different modules, already validated through unit testing, work seamlessly together. This includes verifying that the submission of a maintenance request correctly triggers the approval process, and that the notification system sends real-time updates. Testing also confirms smooth data flow between the front end, back end, and notification system, as well as interactions between users and administrators.

6.3 SYSTEM TESTING

System testing evaluates the entire application to confirm that it meets both functional and non-functional requirements, such as user experience, security, and scalability. This phase includes installing the application on different operating systems and environments to ensure compatibility. Any potential errors, bugs, or crashes are identified and resolved, ensuring reliable and stable platform performance under various conditions.

6.4 ACCEPTANCE TESTING

User Acceptance Testing (UAT) involves healthcare institution representatives testing the application to verify that it meets agreed-upon requirements. The UAT process ensures that features such as maintenance request submission, approval processes, notifications, and record management function as specified. Feedback collected during this phase ensures alignment with user needs before deployment in a real-world healthcare environment.

OVERALL PERFORMANCE

To enhance the robustness and reliability of the application, performance and security testing are recommended. Performance testing ensures that the application can handle high loads during periods of increased demand, while security testing identifies potential vulnerabilities and ensures protection against threats like data breaches and unauthorized access. These additional testing measures contribute to a secure and resilient platform, ultimately increasing user trust.

CONCLUSION

The Medical Equipment Management System represents a significant improvement in automating and modernizing equipment maintenance processes within healthcare facilities. By leveraging a user-friendly interface and automated workflow, this system reduces reliance on traditional methods, streamlining the submission, review, and approval of maintenance requests.

The real-time notification and status-tracking features improve communication between staff and administrators, reducing delays and misunderstandings. With a scalable, secure backend, the system is designed to handle the demands of busy healthcare environments, ensuring data privacy and facilitating efficient equipment management.

In the future, this system could be expanded with additional features, such as mobile app compatibility, advanced analytics, and AI-driven insights into equipment usage patterns. Integrating with existing hospital management systems (HMS) and electronic medical records (EMR) systems would create a unified healthcare ecosystem. Advanced analytics could provide deeper insights into equipment usage and maintenance needs, enabling proactive management. Multi-language support and a feedback mechanism would further enhance accessibility and continuous improvement based on user needs.

REFERENCES

REFERENCES TEXTBOOKS:

- Database System Concepts (6th Edition) By Abraham Silberschatz, Henry F. Korth,S. Sudarshan.

WEBSITES:

- Www.Geeksforgeeks.Com
- Www.W3schools.Com

REFERENCE VIDEOS:

- **HTML AND CSS TIPS AND TRICKS:**

  Https://Www.Youtube.Com/Watch?V=Aactxswxsro&List=PL4-

- **HTML AND CSS FOR BEGINNERS:**

  https://www.youtube.com/watch?v=FYErehuSuuw

- **JAVASCRIPT :**

  https://www.youtube.com/watch?v=poo0BXryffI&t=4s

GITHUB PAGE:

https://github.com/Nandhini1008/dbms_output

https://github.com/MINISH4905/DBMS_mini

https://github.com/Mounesh-Kumaran/DBMS-Project