VISHWAKARMA UNIVERSITY
Maximising Human Potential

Project Report on

Name of the Project

Submitted to

VISHWAKARMA UNIVERSITY, PUNE

Under the Initiative of

Contemporary Curriculum, Pedagogy, and Practice

(C2P2)

ADVANCE WEB TECHNOLOGY LAB

By

Name of the Student :Atharva shevate

SRN. No. :202201727

Class/div. : SY / E1

Roll no. : 02

Faculty In-charge: Prof. Jayendra Jadhav

Department of Computer Engineering

Faculty of Science & Technology

Academic Year 2023-2024 Term-II

1. **Project Title**

   MediTrack: A Node.js and MySQL-based Medical Records Management System

2. **Problem Statement**

   "Create a Medical Records Management System using Node.js and MySQL."

3. **Introduction**

   The Medical Records Management System built with Node.js and MySQL offers a modern solution for organizing and accessing medicine data . Leveraging the efficiency of Node.js and the reliability of MySQL, this system ensures secure storage, streamlined workflows, and compliance with healthcare regulations.

4. **Software/Hardware Requirement**

4.1. **Software Requirement**

Node.js: Version 12.x or above

MySQL Server: Version 5.7 or above

IDE (Integrated Development Environment) for coding (e.g., Visual Studio Code, Sublime Text)

Web browser (e.g., Google Chrome, Mozilla Firefox) for testing and accessing the application

MySQL Workbench or any MySQL client for database management

4.2. **Hardware Requirement:**

Network: Internet connection for downloading dependencies and accessing external resources
Processor: Dual-core 2.0 GHz or higher.

index.html: This HTML code sets up a webpage for a Medical Records Management System. It includes a header with the system title, a main section where product information will be displayed, and a footer with copyright information. External CSS and JavaScript files are linked for styling and functionality, respectively.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Medical records Management System </title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1> Medical Records Management System</h1>
```

```
  </header>

  <main>
    <div id="product-list"></div>
  </main>

  <footer>
    <p>&copy; 2024 Medical Records Management System
    </p>
  </footer>

  <script src="script.js"></script>
</body>
</html>
```

index.css: This CSS code provides basic styling for a webpage

```
/* Reset CSS */
 { box-sizing: border-
box; margin: 0; padding:
0; }

body {
  font-family: Arial, sans-serif;
}

/* Global Styles */
.container { max-
width: 1200px;
margin: 0 auto;
padding: 0 20px; }

header { background-color:
rgb(49, 5, 67); color: #fff;
padding: 20px 0; }

main {
  padding: 20px 0;
}

footer {
  background-color:
  #30612b; color: #fff;
  padding: 20px 0;
  text-align: center;
}
```

```
/* Product Card Styles */
.product-card { background-color:
  #f9f9f9; border-radius: 8px; box-shadow:
  0 2px 4px rgba(0, 0, 0, 0.1); padding:
  20px; margin-bottom: 20px;
}

.product-card h3 { font-
  size:  18px;  margin-
  bottom: 10px;
}

.product-card p {
  font-size: 16px;
  color: #666;
}
```

Node.js: The code fetches product data from a server using the Fetch API in Node.js. Upon DOM content loading, it sends a request to '/products' endpoint, handles the response, and dynamically displays product details on the webpage. Each product's name, ID, and price are inserted into separate elements and appended to the 'product-list' container. document.addEventListener('DOMContentLoaded', () => {

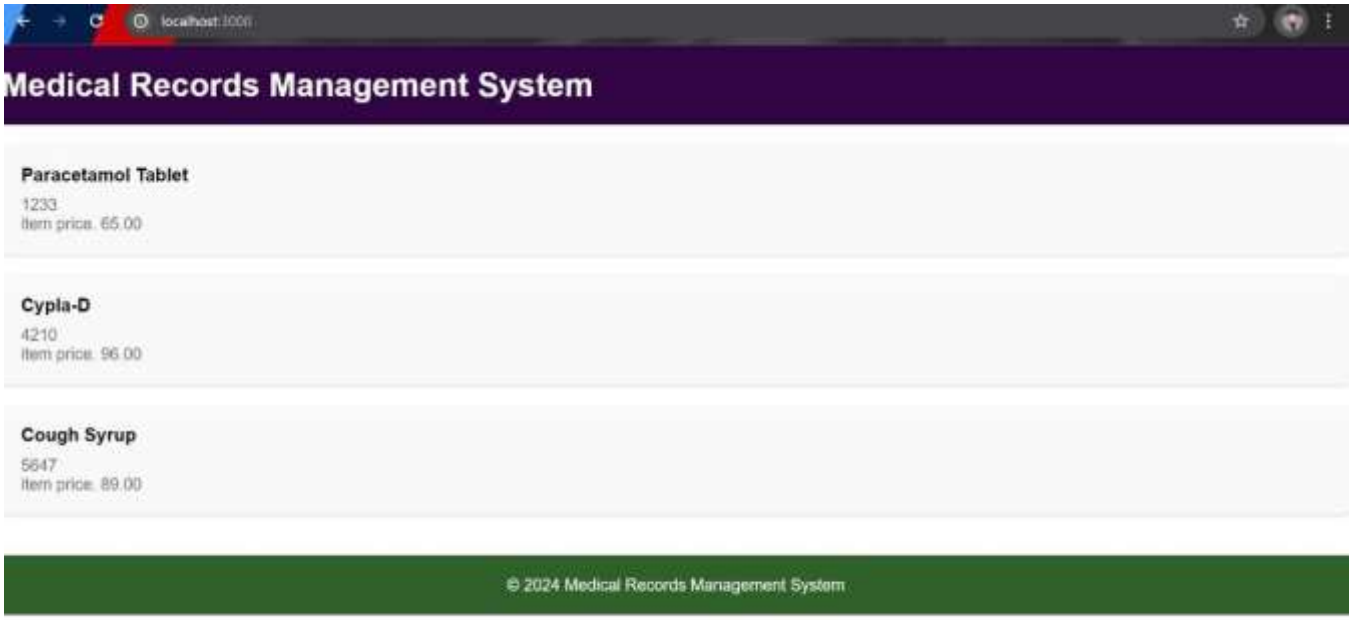
```
  fetch('/products')
    .then(response => {
      if (!response.ok) {
        throw new Error('Network response was not ok');
      } return
      response.json();
    })
```

```
    .then(products => {
      displayProducts(products);
    })
    .catch(error => console.error('Error fetching products:', error));
});

function displayProducts(products) {
  const productList = document.getElementById('product-list');
  productList.innerHTML = '';

  products.forEach(product => {
    const productCard = document.createElement('div');
    productCard.classList.add('product-card');
    productCard.innerHTML = `
      <h3>${product.name}</h3>
      <p>${product.id}</p>
      <p>item price. ${product.price}</p>
    `;
    productList.appendChild(productCard);
  });
}
```

5. **Output**




```
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);

INSERT INTO products (id, name, price) VALUES
(1233, 'Paracetamol Tablet', 65.00),
(4210, 'Cypla-D', 96.00),
(5647, 'Cough Syrup', 89.0);
```

6. **Conclusion:** The integration of Node.js and SQL in the Medical Records Management System project facilitates streamlined connectivity between server-side logic and the database, specifically for managing medical records. Harnessing Node.js' asynchronous functionality and event-driven architecture alongside SQL's robust relational database capabilities ensures seamless retrieval, storage, and management of medical data. This cohesive approach not only optimizes performance,

scalability, and security but also empowers healthcare professionals to efficiently organize and access crucial medical information. By adhering to industry standards and regulations, this system enhances the quality of patient care while maintaining data integrity and confidentiality in medical records management.