Drug inventory and management system

A PROJECT REPORT
for
Mini Project (KCA353) Session (2024-25)

Submitted by

Arjun Srivastava
University Roll No 2300290140039
Aryan Sain
University Roll No 2300290140041
Ashraf Khan
University Roll No 2300290140042
Abhishek Yadav
University Roll No 2300290140009

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Ms. Komal Salgotra
(Assistant Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS

KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

FEBRUARY 2025

CERTIFICATE

Certified that **Arjun Srivastava (2300290140039), Aryan Sain (2300290140041), Ashraf Khan (2300290140042), Abhishek Yadav (2300290140009)** have carried out the project work having "**Drug Inventory and Management System**" (**Mini Project KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself, and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Arjun Srivastava (230029014039)
Aryan Sain (2300290140041)
Ashraf Khan (2300290140042)
Abhishek Yadav (2300290140009)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Ms. Komal Salgotra
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

The Drug Inventory and Management System, designed to revolutionize medication management in healthcare, is a comprehensive solution that combines cutting-edge technology with a patient-centered approach. This abstract highlight the core functionality and purpose of the system.

The Drug Inventory and Management System is an innovative platform developed to address the critical challenges of medication management in healthcare settings. By leveraging advanced tools such as barcode scanning, real-time tracking, and automated workflows, the system ensures seamless and efficient operations. From accurate inventory tracking to streamlined order management, healthcare providers can rely on this solution to maintain optimal stock levels and prevent medication errors.

Through user-friendly interfaces and robust reporting capabilities, healthcare facilities can gain actionable insights into their inventory processes while ensuring compliance with regulatory standards. The system promotes operational efficiency, reduces costs, and enhances patient safety by making medications readily available and properly managed.

With a focus on improving healthcare outcomes, the Drug Inventory Management System empowers providers to deliver better care by optimizing their inventory practices. Join us in transforming healthcare through smarter inventory solutions that prioritize accuracy, efficiency, and patient well-being.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Komal Salgotra** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Arjun Srivastava (230029014039)

Aryan Sain (2300290140041)

Ashraf Khan (2300290140042)

Abhishek Yadav (2300290140009)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

The Drug Inventory and Management System serves as a comprehensive platform addressing the diverse challenges faced by healthcare providers in medication management. It simplifies inventory control with tools for real-time tracking, stock monitoring, and automated reordering, ensuring a smooth and efficient workflow in healthcare facilities.

This system prioritizes patient safety and regulatory compliance, providing healthcare staff with accurate records and tools to meet industry standards. By offering a user-friendly interface and advanced reporting features, it empowers staff to manage inventory efficiently and make data-driven decisions.

With automation, robust tracking capabilities, and seamless integration into existing workflows, the Drug Inventory and Management System aims to enhance healthcare operations. It ensures that medications are always available, properly stored, and accurately dispensed, ultimately contributing to improved patient care and operational efficiency in healthcare settings.

1.2 Project Overview

The Drug Inventory and Management System is a technology-driven solution transforming medication management in healthcare facilities. It features seamless user authentication, real-time inventory tracking, automated reordering, intuitive stock management tools, compliance monitoring, detailed reporting analytics, and a user-friendly interface for effortless navigation.

With its innovative design and healthcare-focused approach, the system ensures accurate medication dispensing, optimizes inventory processes, and enhances operational efficiency while prioritizing patient safety and regulatory adherence.

1.3 Objective

The primary objectives of the Drug Inventory and Management System include:

Operational Simplicity: Designing an intuitive interface that enables staff to quickly manage orders and check inventory.

Regulatory Compliance: Ensuring accurate record-keeping to meet healthcare regulations and standards.

Automated Reordering: Establishing automatic order systems to prevent stockouts and maintain consistent medication availability.

Actionable Reporting: Offering advanced tools to analyze medication usage, empowering staff to make informed decisions for better resource management.

1.4 Key Features

Drug Inventory and Management System include a number of features

Low Stock Alerts: Continuously monitor drug inventory levels and trigger alerts when stock falls below predefined thresholds. This ensures essential medications are always available and helps prevent stockouts through timely procurement.

Drug Expiration Warnings: Track expiration dates of medications and send warnings as they approach expiry. This feature prevents the use of expired drugs, ensures patient safety, and minimizes waste.

Vendor Performance Reports: Evaluate vendor efficiency and reliability by monitoring metrics such as delivery time, order accuracy, and product quality. These reports assist hospitals in selecting dependable vendors and improving supply chain efficiency.

1.5 Scope of the project

The scope of Drug Inventory and Management System extends to the following:

User Interface and Experience: Designing an intuitive and user-friendly interface for healthcare staff to efficiently navigate, manage, and track drug inventory, ensuring seamless usability and minimal training requirements.

Inventory Monitoring and Alerts: Implementing a system to continuously monitor inventory levels, generate low stock alerts, and track drug expiration dates to maintain optimal stock availability and ensure patient safety.

CHAPTER 2

PROBLEM IDENTIFICATION & FEASIBILITY STUDY

2.1 Problem Identification

Identifying potential problems in a Drug Inventory and Management System is crucial for ensuring efficient operations and patient safety. Common issues include medication errors due to manual processes, lack of real-time inventory tracking, inefficiencies in predicting medication demand, inadequate regulatory compliance, difficulty in managing vendor performance, and incomplete documentation. Addressing these challenges through automation, predictive analytics, advanced tracking systems, and robust reporting tools can significantly enhance accuracy, compliance, and overall efficiency in healthcare settings.

2.2 Feasibility Study

Before initiating the development of the Drug Inventory and Management System, a comprehensive feasibility study was conducted to evaluate the practicality and viability of the proposed solution. This study analyzed technical, operational, and economic aspects to ensure the project's success.

2.2.1 Technical Feasibility

Assess the technical requirements and capabilities needed to develop and maintain the system. Consider factors such as database management, system scalability, security protocols

2.2.2 Operational Feasibility

Evaluate the practicality of implementing and managing the system on a daily basis. Consider factors such as training requirements, user adoption, workflow integration, system maintenance, and strategic partnerships with pharmaceutical suppliers and vendors.

2.2.3 Economic Feasibility

The economic study of the Drug Inventory and Management System involves assessing its financial viability and sustainability. This includes evaluating costs for development, implementation, and ongoing operations, as well as identifying potential savings through error reduction, inventory optimization, and efficient vendor management. Market research examines demand for such systems, competitive solutions, and potential ROI, while risk analysis identifies potential challenges, ensuring the system's resilience and long-term sustainability.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Introduction

The Drug Inventory and Management System is an innovative web-based platform designed to transform how healthcare facilities manage medications and inventory. Built on a secure and scalable technology stack, it integrates essential features like inventory tracking, reporting, and data analytics, providing healthcare providers with a seamless solution to monitor and optimize medication management. The system offers user-friendly interfaces, robust security, and real-time insights to ensure accuracy, compliance, and patient safety.

3.2 Functional Requirements

3.2.1 User Authentication

Allow users (admins, hospital staff, and vendors) to create accounts by providing necessary details such as name, email, and password. Implement role-based authentication mechanisms to ensure secure access to user accounts.

3.2.2 Inventory Management

Enable hospital staff to track and manage medications by monitoring stock levels, expiration dates, and batch numbers. Ensure accurate and up-to-date records of all inventory.

3.2.3 Order Management

Allow hospital staff to place orders with vendors, track order statuses, and manage shipments to ensure timely delivery and stock replenishment.

3.3 Non-Functional Requirements

3.3.1 Performance

Ensure quick page load times for a seamless user experience, even during peak usage. The system should scale to handle increased traffic without compromising performance.

3.3.2 Security

Implement secure communication protocols (e.g., HTTPS) to encrypt data transmission. Ensure role-based access control for authentication, protecting sensitive data and adhering to data protection regulations like GDPR.

3.3.3 Scalability

Design the system to scale horizontally by adding more resources to handle traffic and ensure vertical scalability through database optimization and server configuration.

3.3.4 Usability

Ensure the interface is intuitive for users with easy navigation, search functionalities, and accessibility features compliant with standards like WCAG. Provide multilingual support for a global user base.

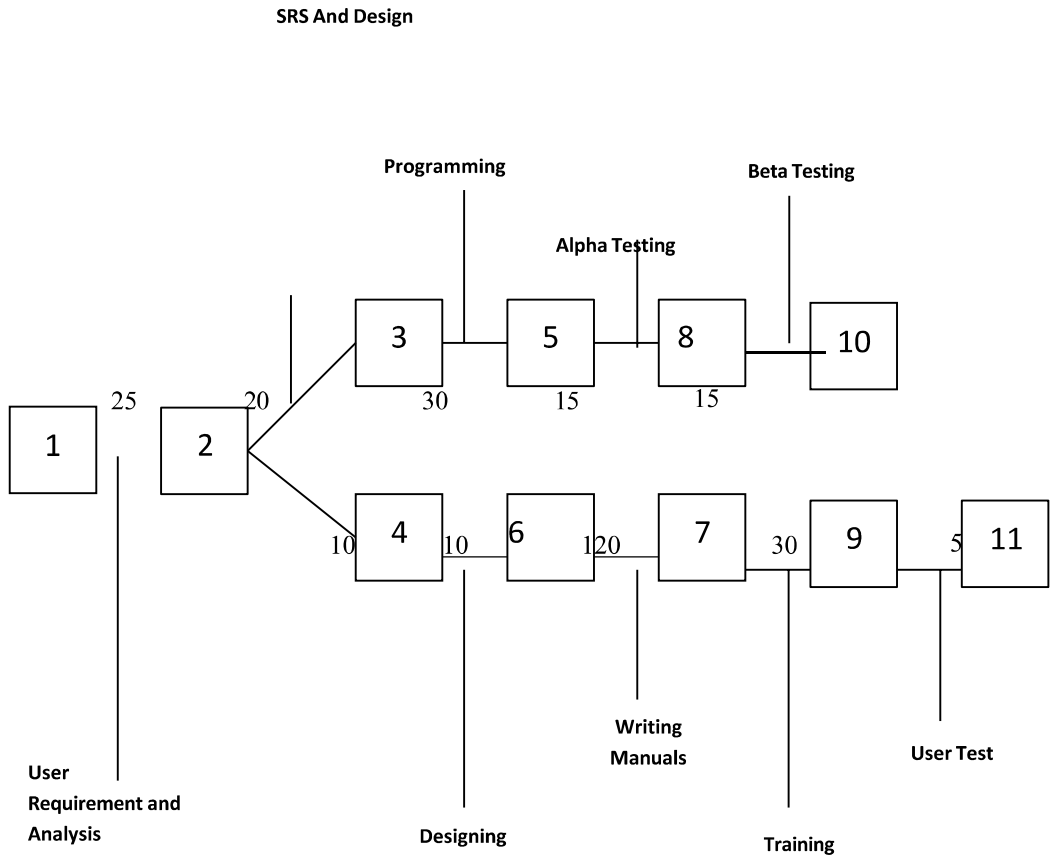
CHAPTER 4

PROJECT PLANNING AND SCHEDULING

4.1 Pert Chart:

A PERT chart is a project management tools used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project as network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project.

The direction of the arrows on the lines indicates the sequence of tasks.



CHAPTER 5

HARDWARE & SOFTWARE SPECIFICATION

5.1 Hardware Specification

The Drug Inventory and Management System will be developed and deployed on a hardware infrastructure that ensures high performance and reliability. The recommended hardware specifications are as follows:

Server:

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB or higher
- Storage: 256 GB SSD or higher

Database Server:

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB or higher
- Storage: 256 GB SSD or higher
- Network Interface: Gigabit Ethernet

Machines:

- Processor: Intel Core i3 or equivalent
- RAM: 4 GB or higher
- Storage: 128 GB SSD or higher
- Network Interface: 100 Mbps Ethernet or Wi-Fi

5.2 Software Specification

The Drug Inventory and Management System will be developed using a combination of server-side and client-side technologies to ensure optimal performance and security. The software specifications for the system are as follows:

Server-Side Technologies:

Operating System: Windows Server 2016 or later

Web Server: Node.js

Database Management System: FireBase

Server-Side Scripting Language: JavaScript (Node.js)

Client-Side Technologies:

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge

Client-Side Scripting: JavaScript

Development Tool

Integrated Development Environment (IDE): Visual Studio Code

Version Control:

Git: Version control for collaborative development

Security:

SSL/TLS: Ensure secure data transmission over the network

Firewall: Implement firewall rules to restrict unauthorized access

Anti-malware Software: Regularly updated anti-malware software on server and client machines

CHAPTER 6

CHOICE OF TOOLS & TECHNOLOGY

6.1 React

React is a JavaScript library used for building user interfaces. React allows developers to describe the UI's appearance at any point in time, and it automatically updates and renders the right components when the data changes.

React organizes the UI into reusable components, making it easier to manage and maintain complex applications. React uses a virtual DOM to optimize and update the actual DOM efficiently, improving performance by minimizing unnecessary rerendering.

React uses JSX, a syntax extension that looks similar to XML or HTML, to describe the structure of UI components in a more concise and readable way**.** React components can manage their internal state, and data can be passed to components through props, allowing for dynamic and interactive UIs

6.2 Data Flow Diagram

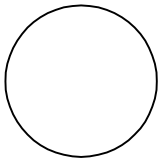
The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It's also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD: -

In the DFD, four symbols are used and they are as follows.

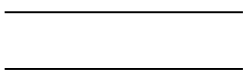
1. A square defines a source (originator) or destination of system data.

2. An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.

3. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.

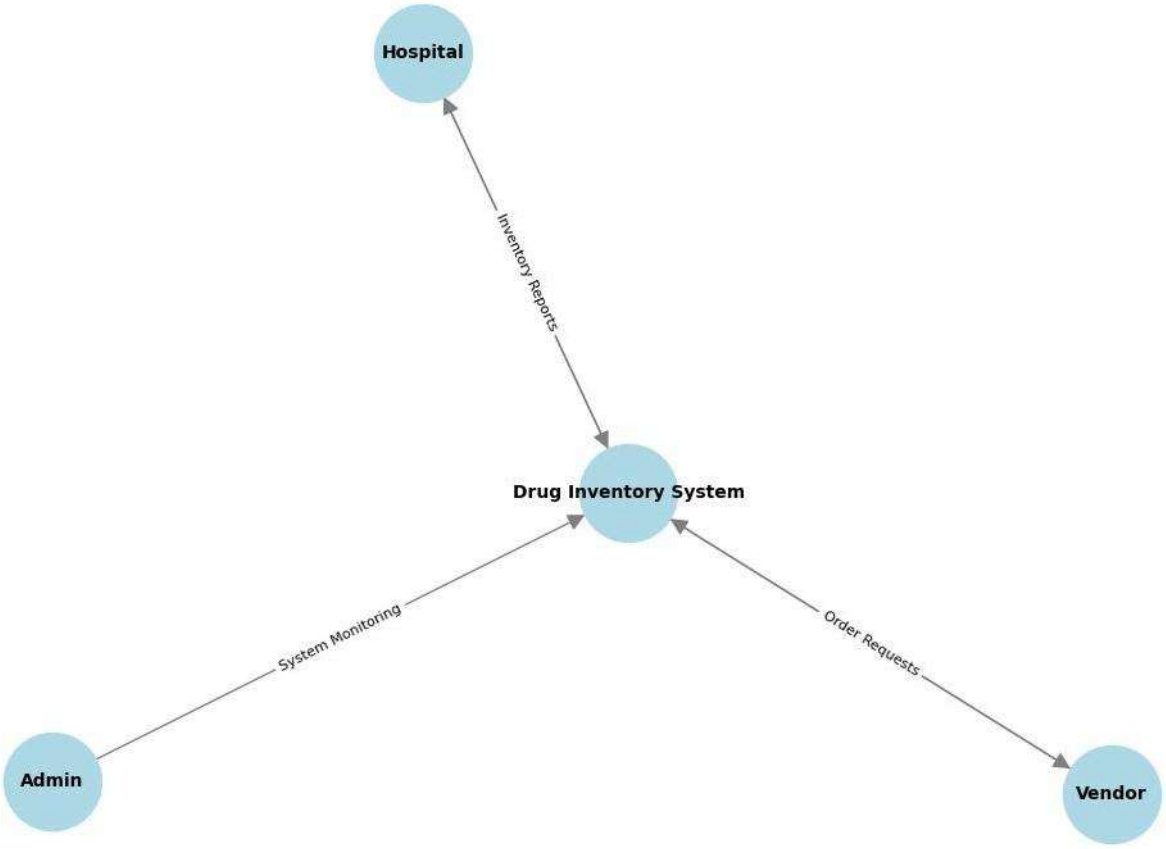


4. An open rectangle is a data store-data at rest, or a temporary repository of data.



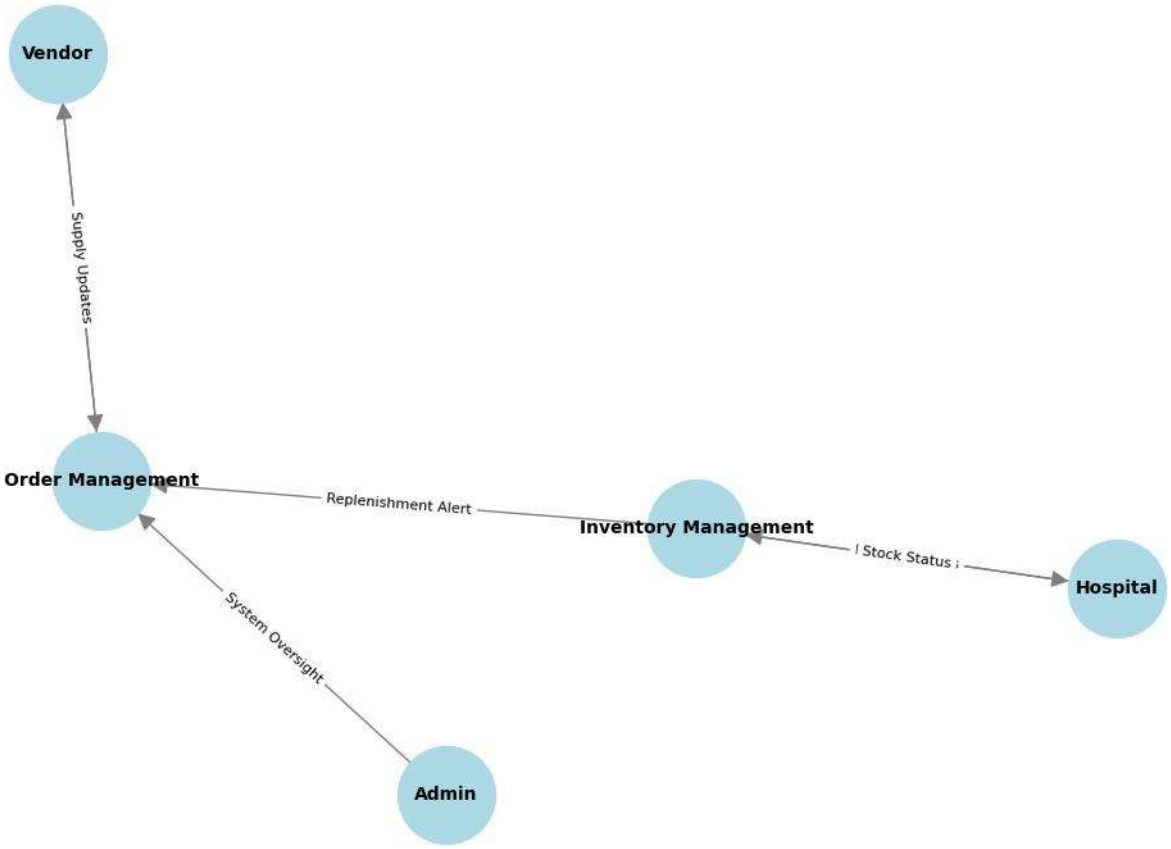
6.3 Context Level Diagram

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Travels and Tales is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.



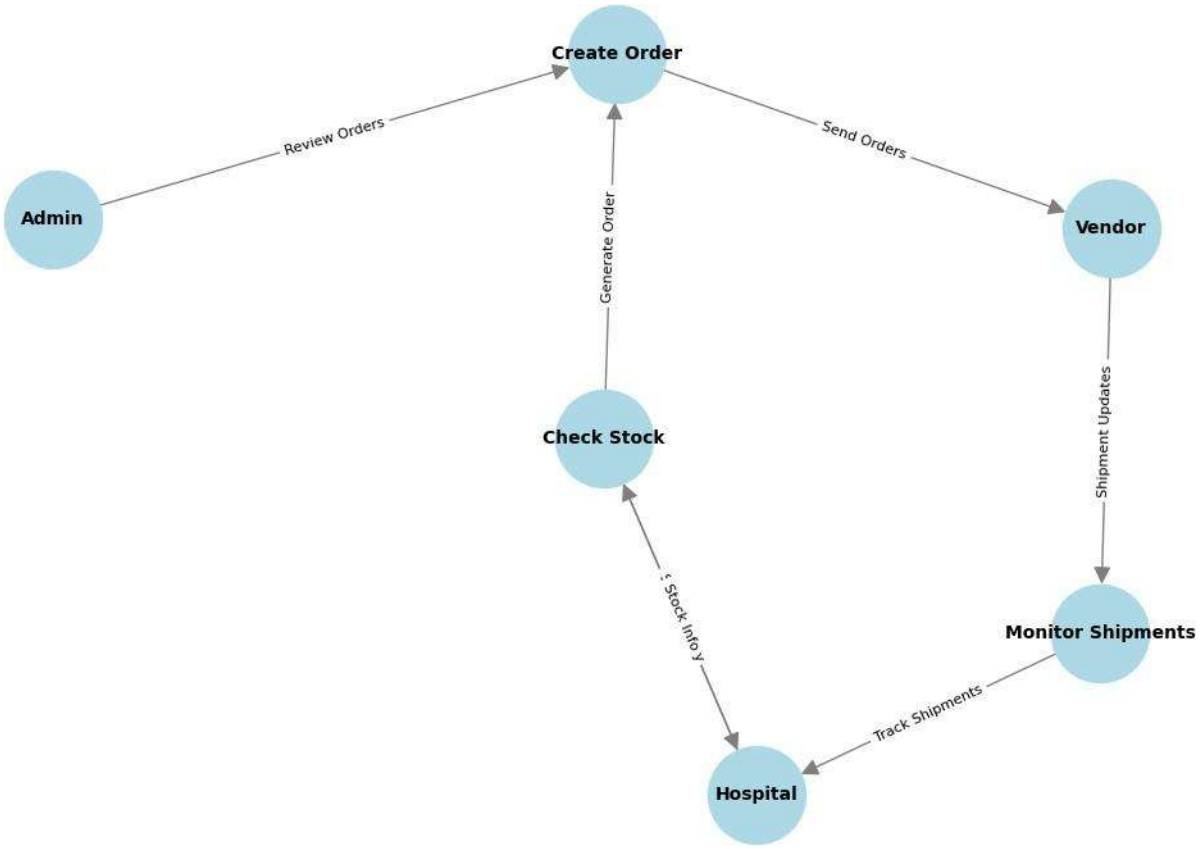
0 Level DFD

Fig 6.1



1st Level DFD

Fig 6.2



2nd Level DFD

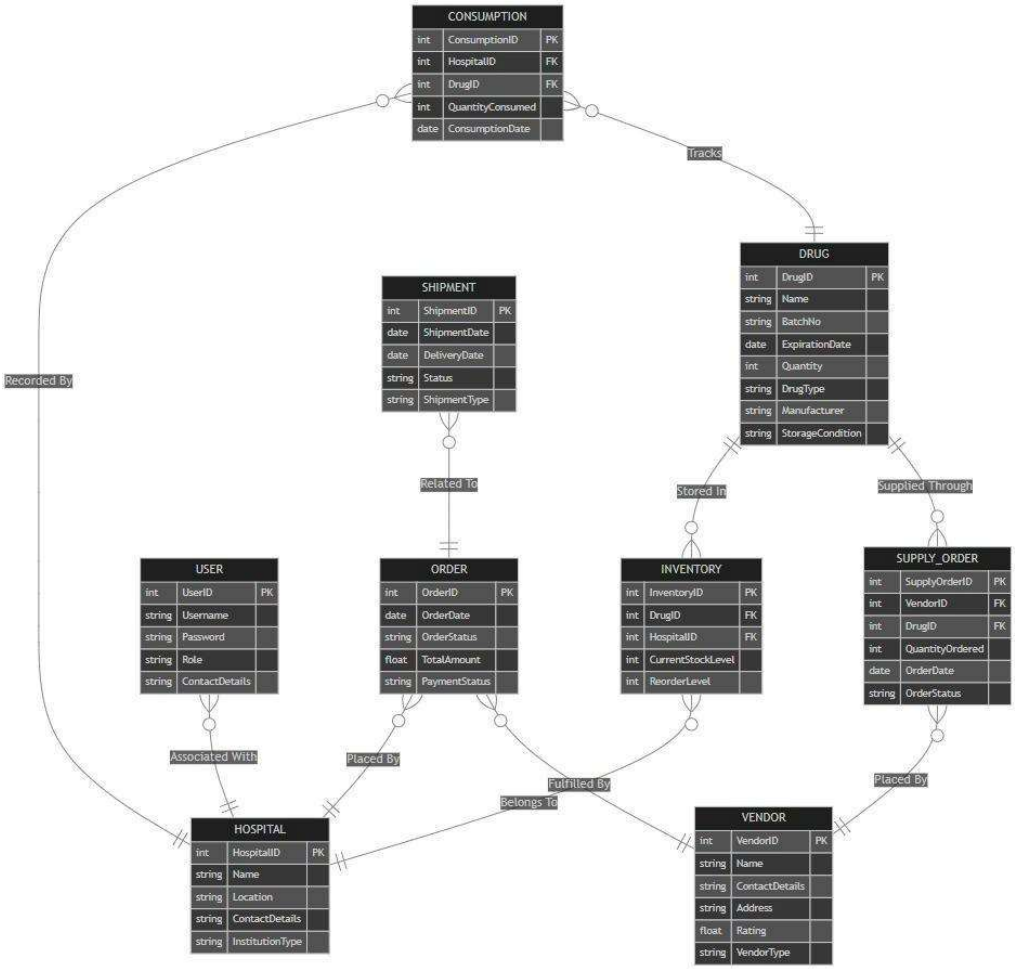
Fig 6.3

CHAPTER 7

ER-DIAGRAM

7.1 Entity-relationship model: -

The entity-relationship model or entity-relationship diagram (ERD) is a data model or diagram for high-level descriptions of conceptual data model, and it provides a graphical notation for representing such data models in the form of entityrelationship diagrams.



7.2 Class Diagram: -

Authentication:

- Classification: Weak Class
- Description: Represents user authentication details, including username and password. This class is responsible for user login functionality.

User:

- Classification: Strong Class
- Description: Represents the users of the system, including administrators and employees.

Package:

- Classification: Strong Class
- Description: Represents the package source and destinations including details such as Package ID ,source and destination price and description.

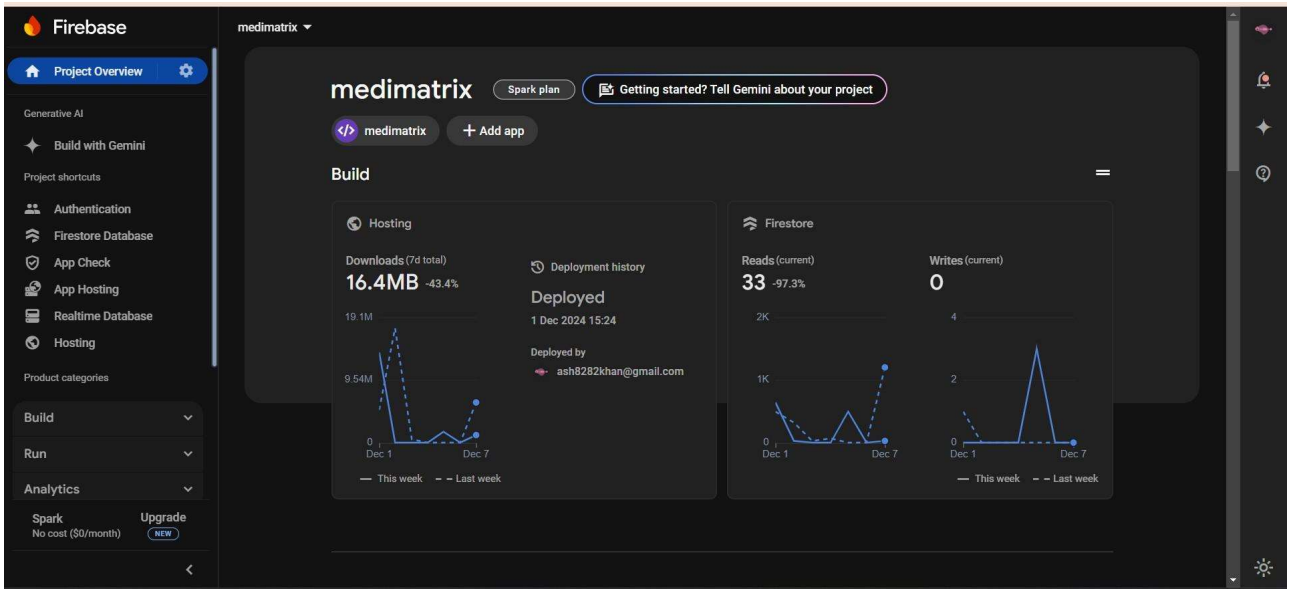
Packages:

- Classification: Strong Class
- Description: Represents the packages from source to destination in the package cart, including details such as package ID, Description and price.

CHAPTER 8

DATABASE

8.1 Admin



Email:

This attribute stores the email address of an individual. It is a unique identifier and is commonly used for user authentication and communication.

Name:

The "name" attribute typically stores the full name of an individual. It might be divided into first name, middle name, and last name for more detailed records.

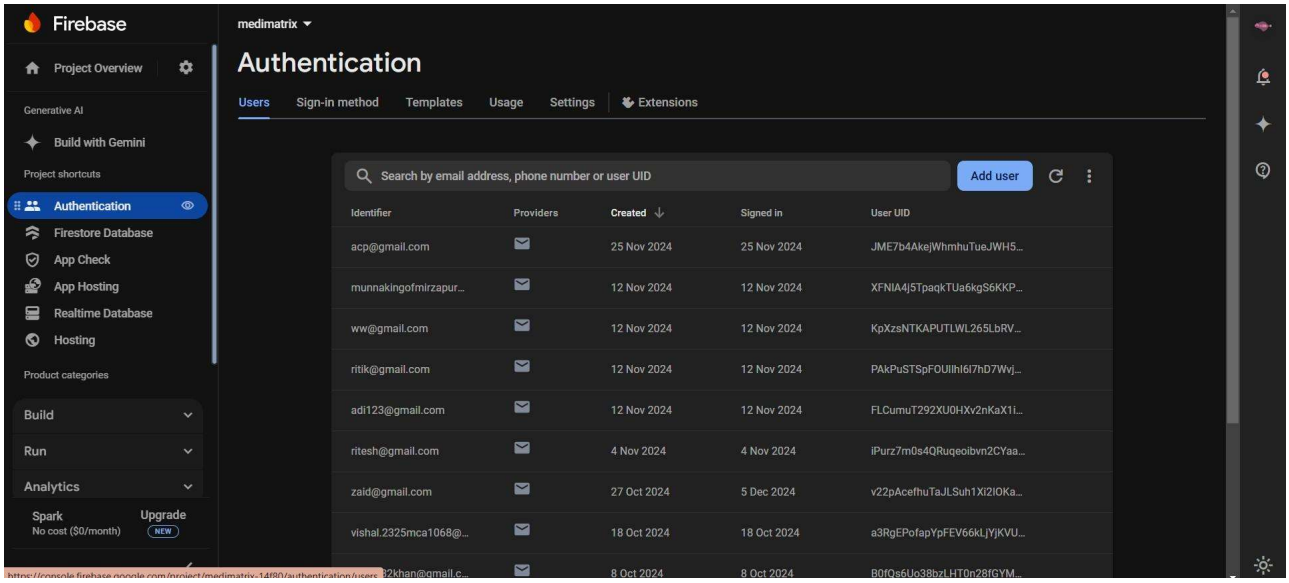
Address:

The "Address" attribute stores the physical address or location details of an individual. It could include components such as street address, city, state, and postal code..

Password:

The "password" attribute stores a securely hashed or encrypted version of the user's password. It is a critical attribute for user authentication, ensuring secure access to the system.

8.2 User



Users Credentials Id:

The "Id" attribute is typically a unique identifier assigned to each individual in the database. It serves as a primary key, ensuring that each record can be uniquely identified and referenced.

Name:

The "Name" attribute stores the full name of an individual. It is a fundamental piece of personal information and is often used for identification and communication purposes.

Email:

The "Email" attribute stores the email address of an individual. It serves as a unique identifier for user accounts and is commonly used for communication and login credentials.

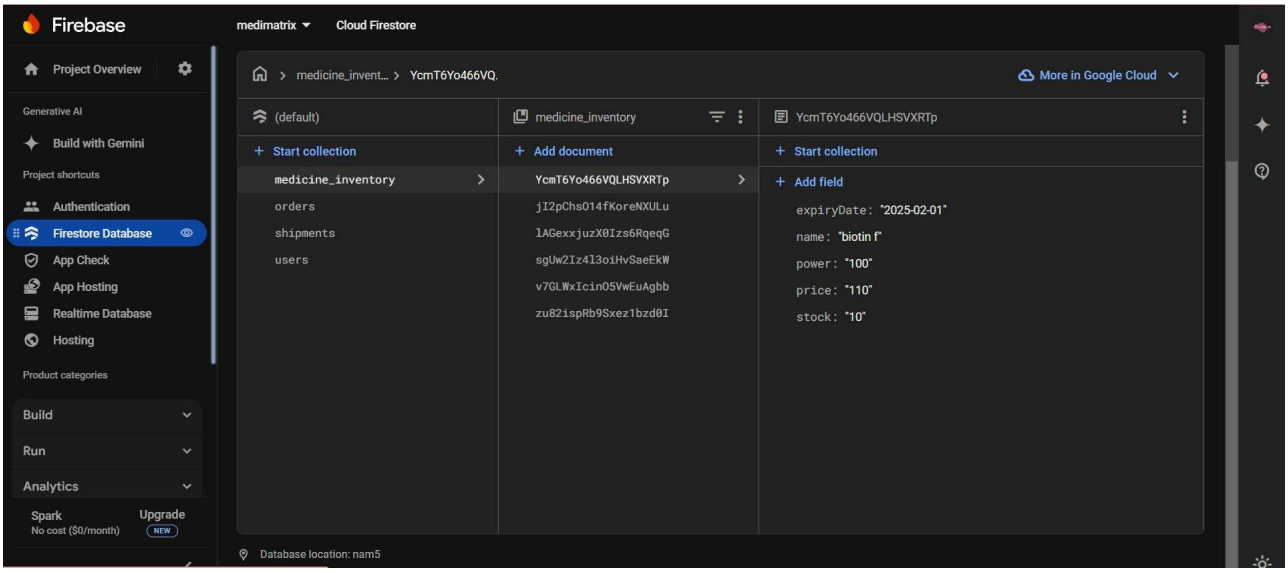
Address:

The "Address" attribute captures the physical address or location details of an individual. It might include components such as street address, city, and state, providing a comprehensive overview of an individual's residence.

Password:

The "Password" attribute stores a securely hashed or encrypted version of the user's password. It is a critical attribute for user authentication, ensuring the security of user accounts by protecting access to sensitive informatio

8.3 Packages



Packages

.

Packages_Id:

The “Id” attribute typically serves as a unique identifier for each record in the database table. It is a primary key that ensures each entry can be uniquely identified and referenced..

Description:

The “Description” attribute provides additional details or a brief description of the product. This field allows for a more comprehensive understanding of the product’s characteristics or features.

Src(Source):

The source refers to the origin or starting point of a project, including the initial problem statement, objectives, and the resources utilized for its implementation. It encapsulates the background information, research findings, and any existing frameworks or methodologies that have contributed to the project's inception.

.

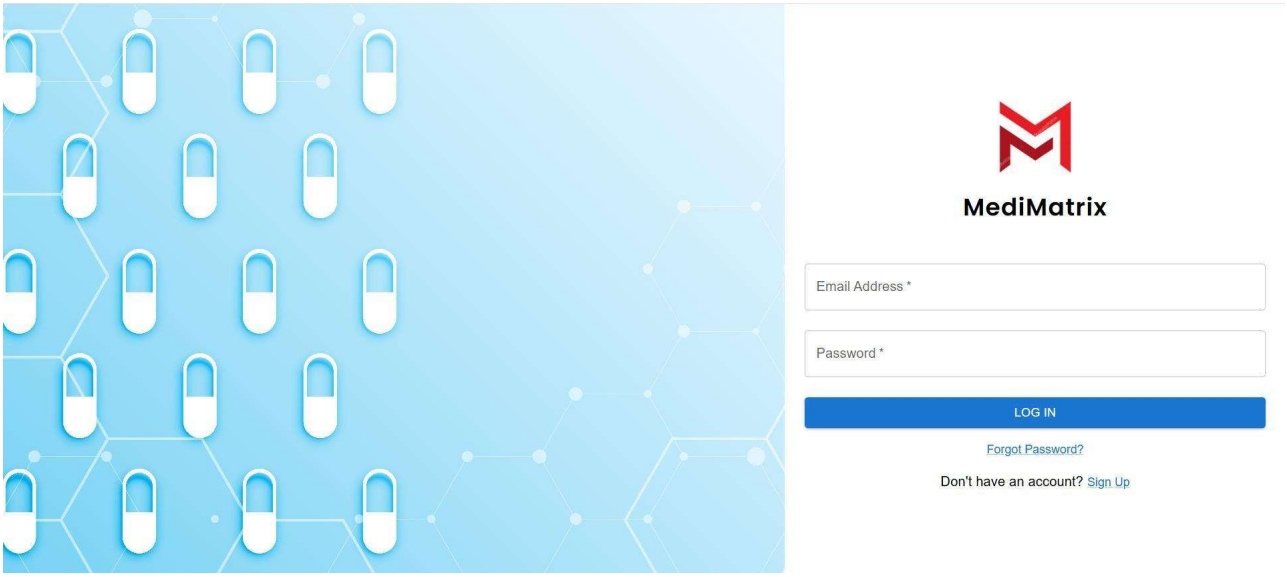
Desc (Destination):

The source refers to the origin or starting point of a project, including the initial problem statement, objectives, and the resources utilized for its implementation. It encapsulates the background information, research findings, and any existing frameworks or methodologies that have contributed to the project's inception.

CHAPTER 9

FORM DESIGN

9.1 Login

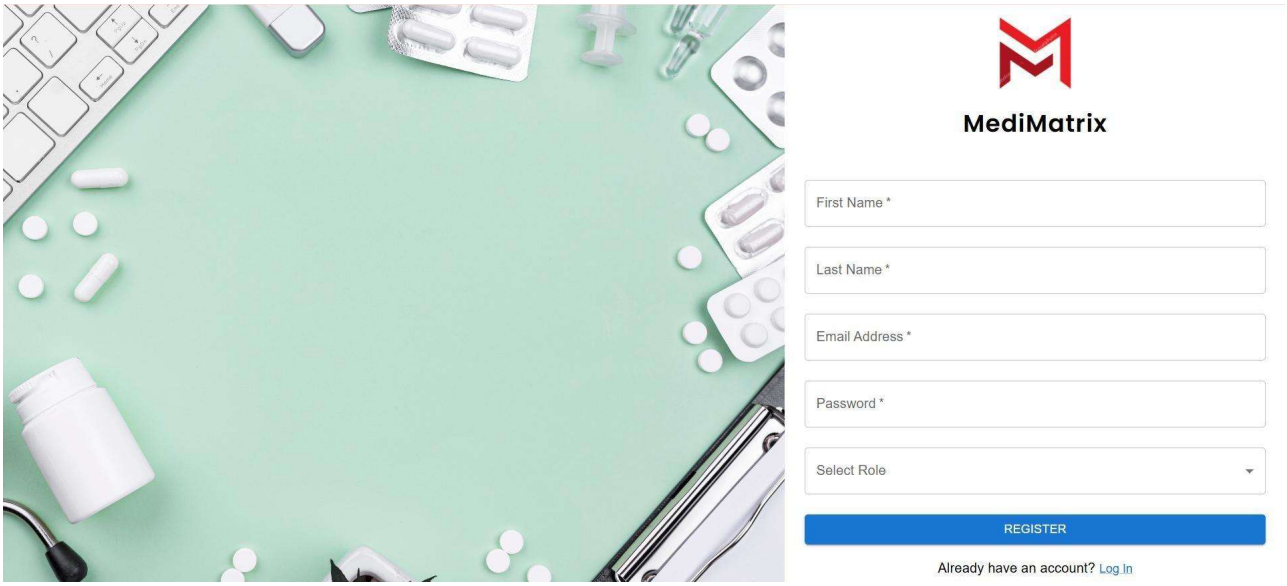


login page

The authentication module in the Travels and Tales website is a fundamental component tasked with securely verifying user identities and controlling access to the platform. This module validates user credentials, typically comprising a username and password, to authenticate users effectively. Upon successful authentication, the module assigns appropriate roles to users, distinguishing between administrators and regular users. Session management functionalities are employed to maintain user sessions throughout their interactions on the website.

The authentication module in the Travels and Tales website is a fundamental component tasked with securely verifying user identities and controlling access to the platform. This module validates user credentials, typically comprising a username and password, to authenticate users effectively. Upon successful authentication, the module assigns appropriate roles to users, distinguishing between administrators and regular users. Session management functionalities are employed to maintain user sessions throughout their interactions on the website

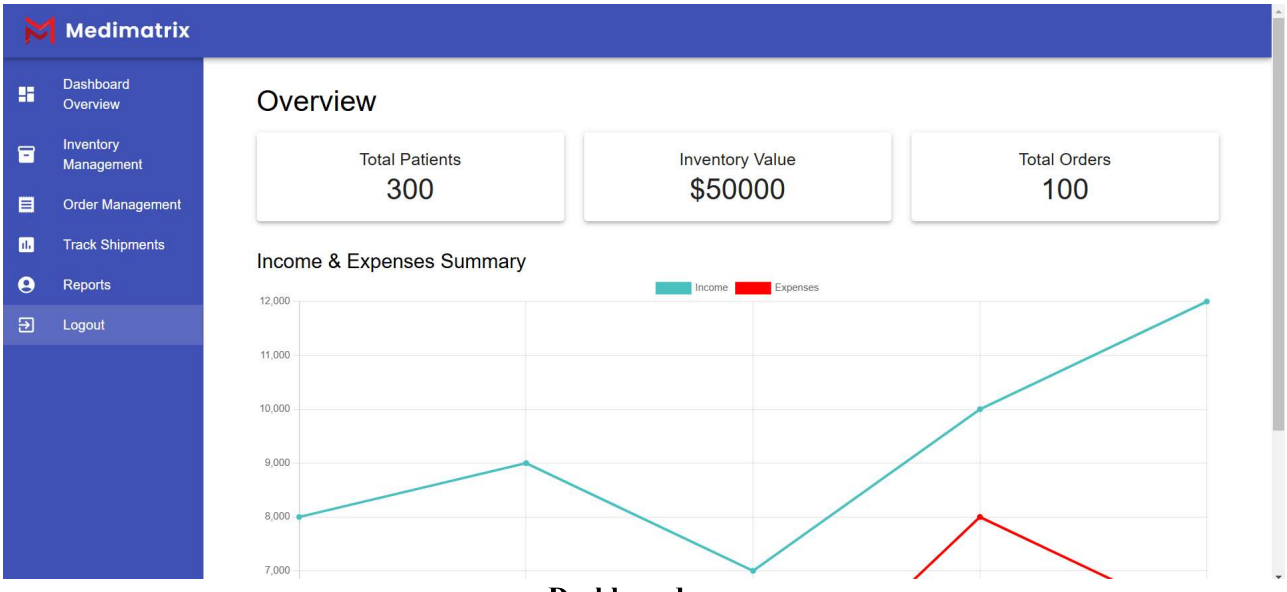
9.2 Signup



Signup Page

The signup page on Travels and Tales facilitates easy registration for users eager to join our community. By providing basic details like name and email, visitors can swiftly create personalized accounts. Optional fields may capture interests, enriching user profiles. Our secure process includes email verification to ensure authenticity. Join us to share stories, connect with travellers, and embark on new adventures!

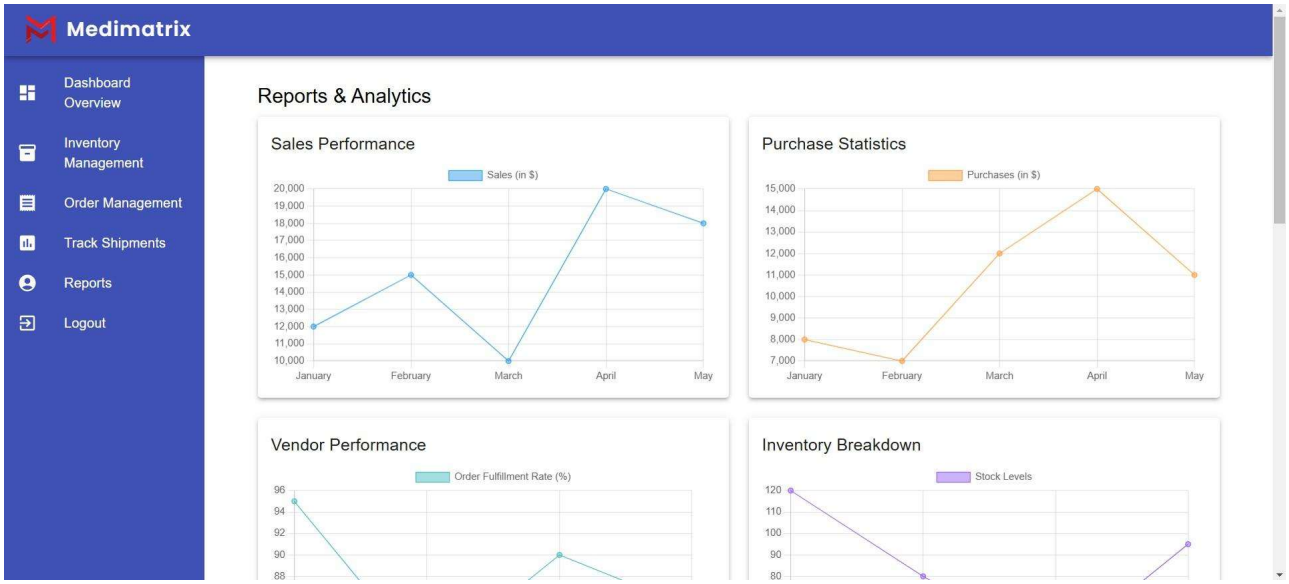
9.3 Admin Dashboard



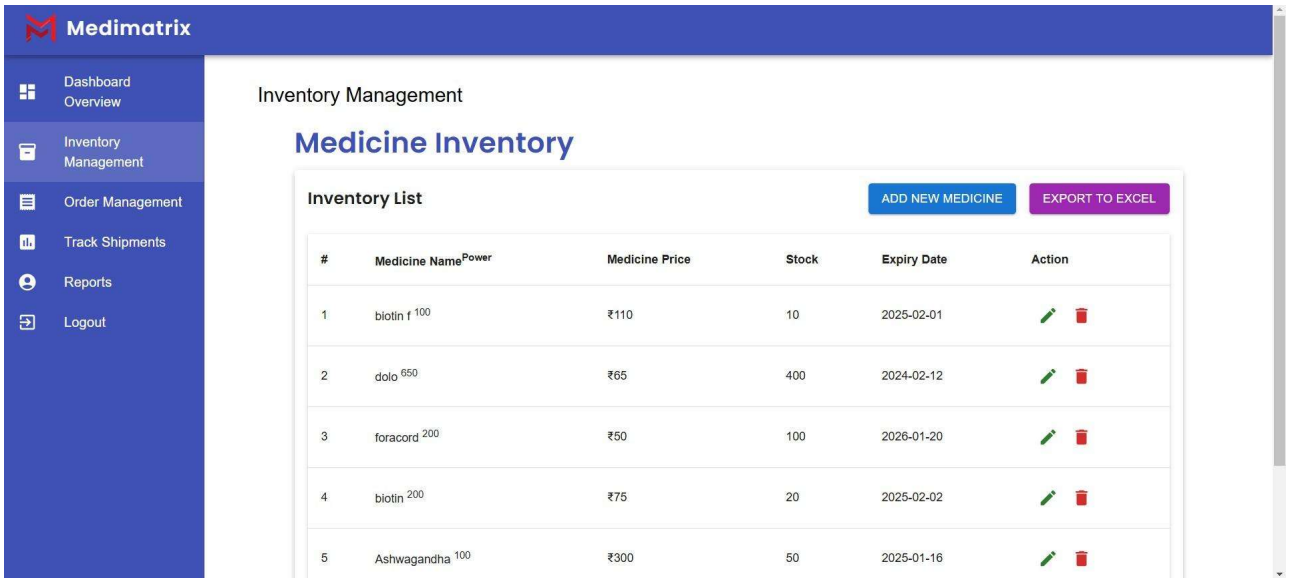
Dashboard page

The user module of the **Drug Inventory and Supply Chain Tracking System** offers a robust and interactive platform for administrators, hospitals, and vendors to manage and track the pharmaceutical supply chain effectively. With a user-friendly interface, members can easily access and manage essential tasks such as

inventory updates, order placements, shipment tracking, and vendor management. The system allows hospitals to monitor drug stocks, view expiring items, and generate notifications for low-stock alerts, ensuring timely procurement. Vendors can manage supply orders, track shipments, and update delivery statuses, enhancing communication with hospitals. Admin users have comprehensive control, including managing users, generating reports, and overseeing the entire system for seamless operation. Join us in transforming the way healthcare institutions manage their pharmaceutical supply chains, optimizing efficiency and improving patient care.



Reports page



Inventory management page

CHAPTER 10

CODING

10.1 App

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes, Navigate } from 'react-router-dom';

// Context Providers
import { AuthProvider, useAuth } from './contexts/AuthContext';
import { InventoryProvider } from './contexts/InventoryContext';
import { OrderProvider } from './contexts/OrderContext';
import { ShipmentProvider } from './contexts/ShipmentContext';
import { NotificationProvider } from './contexts/NotificationContext';
import { AuditLogProvider } from './contexts/AuditLogContext';
import { VendorProvider } from './contexts/VendorContext';

// Page Components
import LoginPage from './pages/auth/LoginPage';
import RegisterPage from './pages/auth/RegisterPage';
import ForgotPasswordPage from './pages/auth/ForgotPasswordPage';
import AdminDashboard from './pages/dashboard/AdminDashboard';
import HospitalDashboard from './pages/dashboard/HospitalDashboard';
import VendorDashboard from './pages/dashboard/VendorDashboard';
import InventoryOverview from './pages/inventory/InventoryOverview';
import AddDrugPage from './pages/inventory/AddDrugPage';
import EditDrugPage from './pages/inventory/EditDrugPage';
import DeleteDrugPage from './pages/inventory/DeleteDrugPage';
import OrderListPage  from  './pages/orders/OrderListPage';
import NewOrderPage from './pages/orders/NewOrderPage';
import OrderTrackingPage from './pages/orders/OrderTrackingPage';
import ShipmentListPage from './pages/shipments/ShipmentListPage';
import TrackShipmentPage from './pages/shipments/TrackShipmentPage';
import UserListPage from './pages/users/UserListPage';
import AddUserPage from './pages/users/AddUserPage';
import VendorListPage from './pages/vendors/VendorListPage';
import ReportsOverviewPage from './pages/reports/ReportsOverviewPage';
import NotificationsListPage from './pages/notifications/NotificationsListPage';
import UserProfilePage from './pages/settings/UserProfilePage';
import ChangePasswordPage from './pages/settings/ChangePasswordPage';

// Private Route Component
const PrivateRoute = ({ children }) => {
  const { currentUser } = useAuth();

  // Debugging user state
```

```
  console.log('Current User:', currentUser);

  // Redirect to login page if not authenticated
  return currentUser ? children : <Navigate to="/" />;
};

// Main Application Component
function App() {
  return (
    <AuthProvider>
      <InventoryProvider>
        <OrderProvider>
          <ShipmentProvider>
            <NotificationProvider>
              <AuditLogProvider>
                <VendorProvider>
                  <Router>
                    <Routes>
                      {/* Public Routes */}
                      <Route path="/" element={<LoginPage />} />
                      <Route path="/register" element={<RegisterPage />} />
                      <Route path="/forgot-password" element={<ForgotPasswordPage />} />

                      {/* Private Routes */}
                      <Route
                        path="/dashboard/admin"
                        element={
                          <PrivateRoute>
                            <AdminDashboard />
                          </PrivateRoute>
                        }
                      />
                      <Route
                        path="/dashboard/hospital"
                        element={
                          <PrivateRoute>
                            <HospitalDashboard />
                          </PrivateRoute>
                        }
                      />
                      <Route
                        path="/dashboard/vendor"
                        element={
                          <PrivateRoute>
                            <VendorDashboard />
                          </PrivateRoute>
                        }
                      />
                      <Route
                        path="/inventory"
                        element={
                          <PrivateRoute>
                            <InventoryOverview />
                          </PrivateRoute>
```

```
  }
/>
<Route
  path="/inventory/add"
  element={
    <PrivateRoute>
      <AddDrugPage />
    </PrivateRoute>
  }
/>
<Route
  path="/inventory/edit"
  element={
    <PrivateRoute>
      <EditDrugPage />
    </PrivateRoute>
  }
/>
<Route
  path="/inventory/delete/:id"
  element={
    <PrivateRoute>
      <DeleteDrugPage />
    </PrivateRoute>
  }
/>
<Route
  path="/orders"
  element={
    <PrivateRoute>
      <OrderListPage />
    </PrivateRoute>
  }
/>
<Route
  path="/orders/new"
  element={
    <PrivateRoute>
      <NewOrderPage />
    </PrivateRoute>
  }
/>
<Route
  path="/orders/track/:id"
  element={
    <PrivateRoute>
      <OrderTrackingPage />
    </PrivateRoute>
  }
/>
<Route
  path="/shipments"
  element={
    <PrivateRoute>
```

```
      <ShipmentListPage />
    </PrivateRoute>
  }
/>
<Route
  path="/shipments/track/:id"
  element={
    <PrivateRoute>
      <TrackShipmentPage />
    </PrivateRoute>
  }
/>
<Route
  path="/users"
  element={
    <PrivateRoute>
      <UserListPage />
    </PrivateRoute>
  }
/>
<Route
  path="/users/add"
  element={
    <PrivateRoute>
      <AddUserPage />
    </PrivateRoute>
  }
/>
<Route
  path="/vendors"
  element={
    <PrivateRoute>
      <VendorListPage />
    </PrivateRoute>
  }
/>
<Route
  path="/reports"
  element={
    <PrivateRoute>
      <ReportsOverviewPage />
    </PrivateRoute>
  }
/>
<Route
  path="/notifications"
  element={
    <PrivateRoute>
      <NotificationsListPage />
    </PrivateRoute>
  }
/>
<Route
  path="/settings/profile"
```

```
                element={
                  <PrivateRoute>
                    <UserProfilePage />
                  </PrivateRoute>
                }
              />
              <Route
                path="/settings/change-password"
                element={
                  <PrivateRoute>
                    <ChangePasswordPage />
                  </PrivateRoute>
                }
              />
            </Routes>
          </Router>
        </VendorProvider>
      </AuditLogProvider>
    </NotificationProvider>
  </ShipmentProvider>
 </OrderProvider>
</InventoryProvider>
</AuthProvider>
);
}

export default App;
```

Pages

```
import React, { useState } from 'react';
import {
  AppBar,
  Toolbar,
  Typography,
  Container,
  Grid,
  Paper,
  Drawer,
  List,
  ListItem,
  ListItemIcon,
  ListItemText,
  Box,
  TextField,
  Button
} from '@mui/material';
import {
  Dashboard,
  Assessment,
  AccountCircle,
```

```
  ExitToApp,
} from '@mui/icons-material';
import { Line, Pie, Bar } from 'react-chartjs-2';
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  BarElement,
  Tooltip,
  Legend,
  ArcElement,
} from 'chart.js';
import { useNavigate } from 'react-router-dom';
import logo from '../../assets/logo.png';
import '@fontsource/poppins';

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, BarElement, ArcElement, Tooltip,
Legend);

export default function AdminDashboard() {
  const [selectedSection, setSelectedSection] = useState('overview');
  const navigate = useNavigate();

  // Data for reports
  const salesData = {
    labels: ['January', 'February', 'March', 'April', 'May'],
    datasets: [
      {
        label: 'Sales',
        data: [5000, 7000, 8000, 6000, 9000],
        backgroundColor: 'rgba(75,192,192,0.4)',
        borderColor: 'rgba(75,192,192,1)',
        borderWidth: 1,
      },
      {
        label: 'Purchases',
        data: [3000, 4000, 5000, 3500, 4500],
        backgroundColor: 'rgba(255,159,64,0.4)',
        borderColor: 'rgba(255,159,64,1)',
        borderWidth: 1,
      },
    ],
  };

  const userData = {
    labels: ['Admin', 'Vendors', 'Hospitals'],
    datasets: [
      {
        label: 'User Distribution',
        data: [5, 10, 15],
        backgroundColor: ['#3f51b5', '#4caf50', '#ff9800'],
        hoverOffset: 4,
```

```
    },
  ],
};

const vendorPerformanceData = {
  labels: ['Vendor A', 'Vendor B', 'Vendor C', 'Vendor D'],
  datasets: [
    {
      label: 'Performance (Orders Delivered)',
      data: [120, 150, 100, 170],
      backgroundColor: ['#2196f3', '#4caf50', '#ff5722', '#9c27b0'],
    },
  ],
};

const handleLogout = () => {
  localStorage.removeItem('authToken'); // Example: remove token
  navigate('/'); // Redirect to the login page
};

const renderContent = () => {
  switch (selectedSection) {
  case 'reports':
      return (
        <Box>
          <Typography variant="h4" sx={{ marginBottom: 2 }}>Reports</Typography>
          <Grid container spacing={3}>
            {/* Sales and Purchases */}
            <Grid item xs={12} md={6}>
              <Paper elevation={3} sx={{ padding: 2 }}>
                <Typography variant="h6">Sales and Purchases</Typography>
                <Line data={salesData} />
              </Paper>
            </Grid>

            {/* User Distribution */}
            <Grid item xs={12} md={6}>
              <Paper elevation={3} sx={{ padding: 2 }}>
                <Typography variant="h6">User Distribution</Typography>
                <Pie data={userData} />
              </Paper>
            </Grid>

            {/* Vendor Performance */}
            <Grid item xs={12}>
              <Paper elevation={3} sx={{ padding: 2 }}>
                <Typography variant="h6">Vendor Performance</Typography>
                <Bar data={vendorPerformanceData} />
              </Paper>
            </Grid>
          </Grid>
        </Box>
      );
      case 'profile':
```

```
return (
  <Box>
    <Typography variant="h5" sx={{ marginBottom: 2 }}>Profile Management</Typography>
    <Grid container spacing={3}>
      <Grid item xs={12} md={6}>
        <Paper elevation={3} sx={{ padding: 3 }}>
          <Typography variant="h6" sx={{ marginBottom: 2 }}>Profile Details</Typography>
          <Box component="form">
            <Grid container spacing={2}>
              <Grid item xs={12} sm={6}>
                <TextField
                  fullWidth
                  label="Name"
                  variant="outlined"
                  value="Admin Name" // Replace with dynamic value
                  disabled
                />
              </Grid>
              <Grid item xs={12} sm={6}>
                <TextField
                  fullWidth
                  label="Email"
                  variant="outlined"
                  value="admin@example.com" // Replace with dynamic value
                  disabled
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                  fullWidth
                  label="Phone"
                  variant="outlined"
                  value="+1234567890" // Replace with dynamic value
                  disabled
                />
              </Grid>
            </Grid>
            <Box sx={{ marginTop: 2, display: 'flex', justifyContent: 'flex-end' }}>
              <Button variant="contained" color="primary">
                Save Changes
              </Button>
            </Box>
          </Box>
        </Paper>
      </Grid>
      <Grid item xs={12} md={6}>
        <Paper elevation={3} sx={{ padding: 3 }}>
          <Typography variant="h6" sx={{ marginBottom: 2 }}>Change Password</Typography>
          <Box component="form">
            <Grid container spacing={2}>
              <Grid item xs={12}>
                <TextField
                  fullWidth
                  label="Current Password"
```

```
                type="password"
                variant="outlined"
                value=""
              />
            </Grid>
            <Grid item xs={12}>
              <TextField
                fullWidth
                label="New Password"
                type="password"
                variant="outlined"
                value=""
              />
            </Grid>
            <Grid item xs={12}>
              <TextField
                fullWidth
                label="Confirm New Password"
                type="password"
                variant="outlined"
                value=""
              />
            </Grid>
          </Grid>
          <Box sx={{ marginTop: 2, display: 'flex', justifyContent: 'flex-end' }}>
            <Button variant="contained" color="secondary">
              Change Password
            </Button>
          </Box>
        </Box>
      </Paper>
    </Grid>
  </Box>
);

default:
  return (
    <Box>
      <Typography variant="h4" sx={{ marginBottom: 2 }}>Overview</Typography>
      <Grid container spacing={3}>
        <Grid item xs={12} sm={6} md={3}>
          <Paper elevation={3} sx={{ padding: 2, textAlign: 'center' }}>
            <Typography variant="h6">Total Sales</Typography>
            <Typography variant="h4">$50,000</Typography>
          </Paper>
        </Grid>
        <Grid item xs={12} sm={6} md={3}>
          <Paper elevation={3} sx={{ padding: 2, textAlign: 'center' }}>
            <Typography variant="h6">Total Orders</Typography>
            <Typography variant="h4">150</Typography>
          </Paper>
        </Grid>
      </Grid>
```

```
        </Box>
      );
    }
  };

  return (
    <div style={{ display: 'flex' }}>
      <Drawer
        variant="permanent"
        anchor="left"
        sx={{
          width: 240,
          flexShrink: 0,
          '& .MuiDrawer-paper': {
            width: 240,
            boxSizing: 'border-box',
            backgroundColor: '#3f51b5',
            color: '#ffffff',
          },
        }}
      >
        <Toolbar />
        <List>
          {[
            { text: 'Dashboard Overview', icon: <Dashboard />, section: 'overview' },
            { text: 'Reports', icon: <Assessment />, section: 'reports' },
            { text: 'Profile Management', icon: <AccountCircle />, section: 'profile' },
            { text: 'Logout', icon: <ExitToApp />, action: handleLogout },
          ].map((item, index) => (
            <ListItem
              button
              key={item.text}
              onClick={item.action || (() => setSelectedSection(item.section))}
              sx={{
                transition: 'all 0.3s ease-in-out',
                '&:hover': {
                  backgroundColor: 'rgba(255, 255, 255, 0.2)',
                },
              }}
            >
              <ListItemIcon sx={{ color: '#ffffff' }}>{item.icon}</ListItemIcon>
              <ListItemText primary={item.text} />
            </ListItem>
          ))}
        </List>
      </Drawer>

      <Container sx={{ flexGrow: 1, padding: 4 }}>
        <AppBar position="fixed" sx={{ zIndex: (theme) => theme.zIndex.drawer + 1, backgroundColor:
'#3f51b5' }}>
          <Toolbar>
            <Box sx={{ display: 'flex', alignItems: 'center' }}>
              <img
                src={logo}
```

```
            alt="Medimatrix Logo"
            style={{ height: 40, marginRight: 10 }}
           />
           <Typography variant="h5" noWrap sx={{ fontFamily: 'Poppins, sans-serif', fontWeight: 700,
letterSpacing: '1px' }}>
            Medimatrix
           </Typography>
          </Box>
         </Toolbar>
       </AppBar>

       <Toolbar />
       {renderContent()}
     </Container>
   </div>
  );
}


import React, { useState } from 'react';
import {
  AppBar,
  Toolbar,
  Typography,
  Container,
  Grid,
  Paper,
  Drawer,
  List,
  ListItem,
  ListItemIcon,
  ListItemText,
  Box,
} from '@mui/material';
import {
  Dashboard,
  Inventory,
  Receipt,
  Assessment,
  AccountCircle,
  ExitToApp,
} from '@mui/icons-material';
import { Line ,Bar ,Pie} from 'react-chartjs-2';
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Tooltip,
  Legend,
} from 'chart.js';
import { useNavigate } from 'react-router-dom'; // Import useNavigate for routing
import InventoryOverview from '../inventory/InventoryOverview';
```

```
import OrderListPage from '../orders/OrderListPage';
import NewOrderPage from '../orders/NewOrderPage';
import TrackShipmentPage from '../shipments/TrackShipmentPage';
import logo from '../../assets/logo.png';

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, Tooltip, Legend);

export default function HospitalDashboard() {
  const [selectedSection, setSelectedSection] = useState('overview');
  const navigate = useNavigate(); // Initialize navigate

  const handleLogout = () => {
    // Perform any additional logout actions here, like clearing session data
    navigate('/'); // Redirect to the login page
  };

  const statistics = {
    totalPatients: 300,
    totalInventoryValue: 50000,
    totalOrders: 100,
  };

  const profitLossData = {
    labels: ['January', 'February', 'March', 'April', 'May'],
    datasets: [
      {
        label: 'Income',
        data: [8000, 9000, 7000, 10000, 12000],
        fill: false,
        backgroundColor: 'rgba(75,192,192,1)',
        borderColor: 'rgba(75,192,192,1)',
      },
      {
        label: 'Expenses',
        data: [5000, 6000, 4000, 8000, 6000],
        fill: false,
        backgroundColor: 'rgba(255,0,0,1)',
        borderColor: 'rgba(255,0,0,1)',
      },
    ],
  };

  const renderContent = () => {
    switch (selectedSection) {
    case 'inventory':
        return (
          <Box>
            <Typography variant="h5">Inventory Management</Typography>
            <InventoryOverview />
          </Box>
        );
      case 'orders':
        return (
          <Box>
```

```
    <Typography variant="h5">Order Management</Typography>
    <NewOrderPage />
    <OrderListPage />
   </Box>
  );
case 'shipments':
  return (
   <Box>
    <Typography variant="h5">Track Shipments</Typography>
    <TrackShipmentPage />
   </Box>
  );
  case 'reports':
   return (
    <Box>
     <Typography variant="h5" gutterBottom>
      Reports & Analytics
     </Typography>
     <Grid container spacing={3}>
      {/* Sales Report */}
      <Grid item xs={12} md={6}>
       <Paper elevation={3} sx={{ padding: 2 }}>
        <Typography variant="h6" gutterBottom>
         Sales Performance
        </Typography>
        <Line
         data={{
          labels: ['January', 'February', 'March', 'April', 'May'],
          datasets: [
           {
            label: 'Sales (in $)',
            data: [12000, 15000, 10000, 20000, 18000],
            backgroundColor: 'rgba(54, 162, 235, 0.5)',
            borderColor: 'rgba(54, 162, 235, 1)',
            borderWidth: 1,
           },
          ],
         }}
        />
       </Paper>
      </Grid>

      {/* Purchase Report */}
      <Grid item xs={12} md={6}>
       <Paper elevation={3} sx={{ padding: 2 }}>
        <Typography variant="h6" gutterBottom>
         Purchase Statistics
        </Typography>
        <Line
         data={{
          labels: ['January', 'February', 'March', 'April', 'May'],
          datasets: [
           {
            label: 'Purchases (in $)',
```

```
            data: [8000, 7000, 12000, 15000, 11000],
            backgroundColor: 'rgba(255, 159, 64, 0.5)',
            borderColor: 'rgba(255, 159, 64, 1)',
            borderWidth: 1,
          },
        ],
      }}
    />
  </Paper>
</Grid>

{/* Vendor Performance */}
<Grid item xs={12} md={6}>
  <Paper elevation={3} sx={{ padding: 2 }}>
    <Typography variant="h6" gutterBottom>
      Vendor Performance
    </Typography>
    <Line
      data={{
        labels: ['Vendor A', 'Vendor B', 'Vendor C', 'Vendor D'],
        datasets: [
          {
            label: 'Order Fulfillment Rate (%)',
            data: [95, 80, 90, 85],
            backgroundColor: 'rgba(75, 192, 192, 0.5)',
            borderColor: 'rgba(75, 192, 192, 1)',
            borderWidth: 1,
          },
        ],
      }}
    />
  </Paper>
</Grid>

{/* Inventory Overview */}
<Grid item xs={12} md={6}>
  <Paper elevation={3} sx={{ padding: 2 }}>
    <Typography variant="h6" gutterBottom>
      Inventory Breakdown
    </Typography>
    <Line
      data={{
        labels: ['Drug A', 'Drug B', 'Drug C', 'Drug D'],
        datasets: [
          {
            label: 'Stock Levels',
            data: [120, 80, 45, 95],
            backgroundColor: 'rgba(153, 102, 255, 0.5)',
            borderColor: 'rgba(153, 102, 255, 1)',
            borderWidth: 1,
          },
        ],
      }}
    />
```

```
  </Paper>
</Grid>

{/* Pie Chart for Expense Breakdown */}
<Grid item xs={12} md={6}>
  <Paper elevation={3} sx={{ padding: 2 }}>
    <Typography variant="h6" gutterBottom>
      Expense Breakdown
    </Typography>
    <Pie
      data={{
        labels: ['Inventory', 'Staff Salaries', 'Utilities', 'Miscellaneous'],
        datasets: [
          {
            label: 'Expenses',
            data: [5000, 2000, 1500, 500],
            backgroundColor: [
              'rgba(255, 99, 132, 0.5)',
              'rgba(54, 162, 235, 0.5)',
              'rgba(255, 206, 86, 0.5)',
              'rgba(75, 192, 192, 0.5)',
            ],
          },
        ],
      }}
    />
  </Paper>
</Grid>

{/* Shipment Reports */}
<Grid item xs={12} md={6}>
  <Paper elevation={3} sx={{ padding: 2 }}>
    <Typography variant="h6" gutterBottom>
      Shipment Performance
    </Typography>
    <Bar
      data={{
        labels: ['Delivered', 'In Transit', 'Delayed', 'Cancelled'],
        datasets: [
          {
            label: 'Shipment Status Count',
            data: [200, 50, 30, 20],
            backgroundColor: [
              'rgba(75, 192, 192, 0.7)',
              'rgba(54, 162, 235, 0.7)',
              'rgba(255, 206, 86, 0.7)',
              'rgba(255, 99, 132, 0.7)',
            ],
          },
        ],
      }}
      options={{
        plugins: {
          legend: { display: true },
```

```
                },
              }}
            />
          </Paper>
        </Grid>

        {/* Average Delivery Time */}
        <Grid item xs={12} md={6}>
          <Paper elevation={3} sx={{ padding: 2 }}>
            <Typography variant="h6" gutterBottom>
              Average Delivery Time (Days)
            </Typography>
            <Bar
              data={{
                labels: ['Vendor A', 'Vendor B', 'Vendor C', 'Vendor D'],
                datasets: [
                  {
                    label: 'Avg Delivery Time',
                    data: [3, 5, 4, 6],
                    backgroundColor: 'rgba(153, 102, 255, 0.5)',
                    borderColor: 'rgba(153, 102, 255, 1)',
                    borderWidth: 1,
                  },
                ],
              }}
            />
          </Paper>
        </Grid>
      </Grid>
    </Box>
  );

default:
  return (
    <Box>
      <Typography variant="h4" sx={{ marginBottom: 2 }}>Overview</Typography>
      <Grid container spacing={3}>
        <Grid item xs={12} sm={6} md={4}>
          <Paper elevation={3} sx={{ padding: 2, textAlign: 'center', transition: '0.3s', '&:hover': {
boxShadow: '0 8px 16px rgba(0,0,0,0.3)' } }}>
            <Typography variant="h6">Total Patients</Typography>
            <Typography variant="h4">{statistics.totalPatients}</Typography>
          </Paper>
        </Grid>
        <Grid item xs={12} sm={6} md={4}>
          <Paper elevation={3} sx={{ padding: 2, textAlign: 'center', transition: '0.3s', '&:hover': {
boxShadow: '0 8px 16px rgba(0,0,0,0.3)' } }}>
            <Typography variant="h6">Inventory Value</Typography>
            <Typography variant="h4">${statistics.totalInventoryValue}</Typography>
          </Paper>
        </Grid>
        <Grid item xs={12} sm={6} md={4}>
          <Paper elevation={3} sx={{ padding: 2, textAlign: 'center', transition: '0.3s', '&:hover': {
boxShadow: '0 8px 16px rgba(0,0,0,0.3)' } }}>
```

```
              <Typography variant="h6">Total Orders</Typography>
              <Typography variant="h4">{statistics.totalOrders}</Typography>
            </Paper>
          </Grid>
        </Grid>
        <Box sx={{ marginTop: 4 }}>
          <Typography variant="h5">Income & Expenses Summary</Typography>
          <Line data={profitLossData} />
        </Box>
      </Box>
    );
  }
};

 return (
  <div style={{ display: 'flex' }}>
    {/* Sidebar */}
    <Drawer
      variant="permanent"
      anchor="left"
      sx={{
        width: 240,
        flexShrink: 0,
        '& .MuiDrawer-paper': {
          width: 240,
          boxSizing: 'border-box',
          backgroundColor: '#3f51b5',
          color: '#ffffff',
        },
      }}
    >
      <Toolbar />
      <List>
        <ListItem button onClick={() => setSelectedSection('overview')} sx={{ transition: '0.3s', '&:hover': {
backgroundColor: '#5c6bc0' } }}>
          <ListItemIcon sx={{ color: '#ffffff' }}><Dashboard /></ListItemIcon>
          <ListItemText primary="Dashboard Overview" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('inventory')} sx={{ transition: '0.3s', '&:hover': {
backgroundColor: '#5c6bc0' } }}>
          <ListItemIcon sx={{ color: '#ffffff' }}><Inventory /></ListItemIcon>
          <ListItemText primary="Inventory Management" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('orders')} sx={{ transition: '0.3s', '&:hover': {
backgroundColor: '#5c6bc0' } }}>
          <ListItemIcon sx={{ color: '#ffffff' }}><Receipt /></ListItemIcon>
          <ListItemText primary="Order Management" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('shipments')} sx={{ transition: '0.3s', '&:hover': {
backgroundColor: '#5c6bc0' } }}>
          <ListItemIcon sx={{ color: '#ffffff' }}><Assessment /></ListItemIcon>
          <ListItemText primary="Track Shipments" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('reports')} sx={{ transition: '0.3s', '&:hover': {
```

```
backgroundColor: '#5c6bc0' } }}>
          <ListItemIcon sx={{ color: '#ffffff' }}><AccountCircle /></ListItemIcon>
          <ListItemText primary="Reports" />
        </ListItem>
        <ListItem button onClick={handleLogout} sx={{ transition: '0.3s', '&:hover': { backgroundColor:
'#5c6bc0' } }}>
          <ListItemIcon sx={{ color: '#ffffff' }}><ExitToApp /></ListItemIcon>
          <ListItemText primary="Logout" />
        </ListItem>
      </List>
    </Drawer>

    {/* Main Content Area */}
    <Container sx={{ flexGrow: 1, padding: 4 }}>
      {/* Top Bar */}
      <AppBar position="fixed" sx={{ zIndex: (theme) => theme.zIndex.drawer + 1, backgroundColor:
'#3f51b5' }}>
        <Toolbar>
          <Box sx={{ display: 'flex', alignItems: 'center' }}>
            <img
              src={logo}
              alt="Medimatrix Logo"
              style={{ height: 40, marginRight: 10 }}
            />
            <Typography variant="h5" noWrap sx={{ fontFamily: 'Poppins, sans-serif', fontWeight: 700,
letterSpacing: '1px' }}>
              Medimatrix
            </Typography>
          </Box>
        </Toolbar>
      </AppBar>

      {/* Dashboard Content */}
      <Toolbar />
      {renderContent()}
    </Container>
  </div>
 );
}


import React, { useState } from 'react';
import {
 AppBar,
 Toolbar,
 Typography,
 Container,
 Grid,
 Paper,
 Drawer,
 List,
 ListItem,
 ListItemIcon,
 ListItemText,
```

```
  Box,
} from '@mui/material';
import {
  Dashboard,
  Receipt,
  Assessment,
  ExitToApp,
} from '@mui/icons-material';
import { Line, Pie, Bar } from 'react-chartjs-2';
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Tooltip,
  Legend,
  ArcElement,
} from 'chart.js';
import { useNavigate } from 'react-router-dom';
import VendorOrdersPage from '../orders/VendorOrdersPage';
import ShipmentListPage from '../shipments/ShipmentListPage';
import logo from '../../assets/logo.png';

ChartJS.register(
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Tooltip,
  Legend,
  ArcElement // Register ArcElement for Pie chart
);

export default function VendorDashboard() {
  const [selectedSection, setSelectedSection] = useState('overview');
  const navigate = useNavigate();

  const handleLogout = () => {
    console.log('Logout');
    localStorage.removeItem('authToken');
    navigate('/');
  };

  const statistics = {
    totalOrders: 150,
    totalSales: 50000,
  };

  const profitLossData = {
    labels: ['January', 'February', 'March', 'April', 'May'],
    datasets: [
      {
        label: 'Profit',
```

```
      data: [4000, 5000, 3000, 7000, 6000],
      fill: false,
      backgroundColor: 'rgba(75,192,192,1)',
      borderColor: 'rgba(75,192,192,1)',
    },
    {
      label: 'Loss',
      data: [2000, 1000, 3000, 2000, 1500],
      fill: false,
      backgroundColor: 'rgba(255,0,0,1)',
      borderColor: 'rgba(255,0,0,1)',
    },
  ],
};

const orderStatusData = {
  labels: ['Pending', 'Shipped', 'Delivered'],
  datasets: [
    {
      data: [60, 30, 10], // Example data for order statuses
      backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56'],
      hoverOffset: 4,
    },
  ],
};

const salesByMonthData = {
  labels: ['January', 'February', 'March', 'April', 'May'],
  datasets: [
    {
      label: 'Monthly Sales ($)',
      data: [5000, 10000, 8000, 12000, 15000],
      backgroundColor: 'rgba(54, 162, 235, 0.6)',
      borderColor: 'rgba(54, 162, 235, 1)',
      borderWidth: 1,
    },
  ],
};

const renderContent = () => {
  switch (selectedSection) {
  case 'supplyOrders':
      return (
        <Box sx={{ padding: 3 }}>
          <Typography variant="h5" gutterBottom>Supply Orders</Typography>
          <VendorOrdersPage />
          <ShipmentListPage />
        </Box>
      );
    case 'performance':
      return (
        <Box sx={{ padding: 3 }}>
          <Typography variant="h5" gutterBottom>Performance Metrics</Typography>
          <Paper elevation={3} sx={{ padding: 2, marginTop: 2 }}>
```

```
            <Line data={profitLossData} />
          </Paper>
        </Box>
      );
    case 'reports':
      return (
        <Box sx={{ padding: 3 }}>
          <Typography variant="h5" gutterBottom>Reports</Typography>
          <Grid container spacing={3}>
            <Grid item xs={12} sm={6}>
              <Paper elevation={3} sx={{ padding: 2, textAlign: 'center' }}>
                <Typography variant="h6">Order Status Distribution</Typography>
                <Pie data={orderStatusData} />
              </Paper>
            </Grid>
            <Grid item xs={12} sm={6}>
              <Paper elevation={3} sx={{ padding: 2, textAlign: 'center' }}>
                <Typography variant="h6">Sales by Month</Typography>
                <Bar data={salesByMonthData} />
              </Paper>
            </Grid>
          </Grid>
        </Box>
      );
    default:
      return (
        <Box sx={{ padding: 3 }}>
          <Typography variant="h4" gutterBottom>Vendor Dashboard Overview</Typography>
          <Grid container spacing={3}>
            <Grid item xs={12} sm={6}>
              <Paper elevation={3} sx={{ padding: 2, textAlign: 'center' }}>
                <Typography variant="h6">Total Orders</Typography>
                <Typography variant="h4">{statistics.totalOrders}</Typography>
              </Paper>
            </Grid>
            <Grid item xs={12} sm={6}>
              <Paper elevation={3} sx={{ padding: 2, textAlign: 'center' }}>
                <Typography variant="h6">Total Sales</Typography>
                <Typography variant="h4">${statistics.totalSales}</Typography>
              </Paper>
            </Grid>
          </Grid>
        </Box>
      );
  }
};

return (
  <div style={{ display: 'flex' }}>
    {/* Sidebar */}
    <Drawer
      variant="permanent"
      anchor="left"
      sx={{
```

```
        width: 240,
        flexShrink: 0,
        '& .MuiDrawer-paper': {
          width: 240,
          boxSizing: 'border-box',
        },
      }}
    >
      <Toolbar />
      <List>
        <ListItem button onClick={() => setSelectedSection('overview')} selected={selectedSection ===
'overview'}>
          <ListItemIcon><Dashboard /></ListItemIcon>
          <ListItemText primary="Dashboard Overview" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('supplyOrders')} selected={selectedSection ===
'supplyOrders'}>
          <ListItemIcon><Receipt /></ListItemIcon>
          <ListItemText primary="Supply Orders" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('performance')} selected={selectedSection ===
'performance'}>
          <ListItemIcon><Assessment /></ListItemIcon>
          <ListItemText primary="Performance" />
        </ListItem>
        <ListItem button onClick={() => setSelectedSection('reports')} selected={selectedSection ===
'reports'}>
          <ListItemIcon><Assessment /></ListItemIcon>
          <ListItemText primary="Reports" />
        </ListItem>
        <ListItem button onClick={handleLogout}>
          <ListItemIcon><ExitToApp /></ListItemIcon>
          <ListItemText primary="Logout" />
        </ListItem>
      </List>
    </Drawer>

    {/* Main Content Area */}
    <Container sx={{ flexGrow: 1, padding: 4 }}>
      <AppBar position="fixed" sx={{ zIndex: (theme) => theme.zIndex.drawer + 1, backgroundColor:
'#3f51b5' }}>
        <Toolbar sx={{ display: 'flex', justifyContent: 'center' }}>
          <Box sx={{ display: 'flex', alignItems: 'center' }}>
            <img src={logo} alt="Medimatrix Logo" style={{ height: 40, marginRight: 10 }} />
            <Typography variant="h5" noWrap sx={{ fontFamily: 'Poppins, sans-serif', fontWeight: 700,
letterSpacing: '1px' }}>
              Medimatrix
            </Typography>
          </Box>
        </Toolbar>
      </AppBar>

      <Toolbar />
      {renderContent()}
```

```
      </Container>
    </div>
  );
}


import React, { useRef, useState } from 'react';
import { useAuth } from '../../contexts/AuthContext';
import { Button, TextField, Container, Typography, Link, Box } from '@mui/material';
import { useNavigate } from 'react-router-dom';
import { doc, getDoc } from 'firebase/firestore'; // To fetch user roles
import { firestore } from '../../firebase'; // Firebase Firestore
import logo from '../../assets/logo.png'; // Assuming your logo image is saved in assets folder
import backgroundImg from '../../assets/login-background.jpg'; // Replace with your chosen image

export default function LoginPage() {
  const emailRef = useRef();
  const passwordRef = useRef();
  const { login } = useAuth();
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  async function handleSubmit(e) {
    e.preventDefault();
    try {
      setError('');
      setLoading(true);

      // Log the user in
      const userCredential = await login(emailRef.current.value, passwordRef.current.value);
      const user = userCredential.user;

      // Fetch the user role from Firestore
      const userDoc = await getDoc(doc(firestore, 'users', user.uid));
      const userData = userDoc.data();

      // Role-based redirection
      if (userData.role === 'admin') {
        navigate('/dashboard/admin');
      } else if (userData.role === 'vendor') {
        navigate('/dashboard/vendor');
      } else if (userData.role === 'hospitalStaff') {
        navigate('/dashboard/hospital');
      } else {
        setError('Unknown role, please contact admin.');
      }
    } catch (err) {
      setError('Failed to sign in');
    }
    setLoading(false);
  }

  return (
```

```
   <Box sx={{ display: 'flex', height: '100vh' }}>
     {/* Image Section */}
     <Box
       sx={{
       flex: 1,
         backgroundImage: `url(${backgroundImg})`,
         backgroundSize:  'cover',
         backgroundPosition: 'center',
         display: { xs: 'none', md: 'block' }, // Hides on small screens
       }}
     />

     {/* Login Form Section */}
     <Container maxWidth="sm" sx={{ flex: 1, display: 'flex', flexDirection: 'column', justifyContent: 'center',
padding: 4 }}>
       <Box sx={{ display: 'flex', flexDirection: 'column', alignItems: 'center', mb: 4 }}>
         <img src={logo} alt="App Logo" style={{ width: '100px', marginBottom: '16px' }} />
         <Typography
  variant="h5"
  noWrap
  sx={{
    fontFamily: 'Poppins, sans-serif', // Updated to use the Google Fonts CDN
    fontWeight: 600,
    fontSize: '1.8rem',
    letterSpacing: 1,
    color: 'black',
  }}
>
  MediMatrix
</Typography>

         {error && <Typography color="error" align="center">{error}</Typography>}
       </Box>

       <form onSubmit={handleSubmit} style={{ width: '100%' }}>
         <TextField
           variant="outlined"
           margin="normal"
           required
           fullWidth
           label="Email Address"
           inputRef={emailRef}
         />
         <TextField
           variant="outlined"
           margin="normal"
           required
           fullWidth
           label="Password"
           type="password"
           inputRef={passwordRef}
         />
         <Button
           type="submit"
```

```
          fullWidth
          variant="contained"
          color="primary"
          disabled={loading}
          sx={{ marginTop: 2 }}
        >
          Log In
        </Button>
      </form>

      <Typography align="center" variant="body2" marginTop={2}>
        <Link href="/forgot-password">Forgot Password?</Link>
      </Typography>

      <Typography align="center" style={{ marginTop: '16px' }}>
        Don't have an account?{' '}
        <Link href="/register" variant="body2">Sign Up</Link>
      </Typography>
    </Container>
  </Box>
 );
}


import React, { useRef, useState } from 'react';
import { useAuth } from '../../contexts/AuthContext';
import { Button, TextField, Container, Typography, MenuItem, Select, FormControl, InputLabel, Link, Box }
from '@mui/material';
import { useNavigate } from 'react-router-dom';
import { firestore } from '../../firebase'; // Firebase Firestore for saving user role
import { doc, setDoc } from 'firebase/firestore'; // For adding user role data to Firestore
import logo from '../../assets/logo.png'; // Assuming your logo image is saved in assets folder
import backgroundImg from '../../assets/register-background.jpg'; // Replace with your chosen image

export default function RegisterPage() {
  const firstNameRef = useRef();
  const lastNameRef = useRef();
  const emailRef = useRef();
  const passwordRef = useRef();
  const roleRef = useRef();
  const { signup } = useAuth();
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  async function handleSubmit(e) {
    e.preventDefault();
    try {
      setError('');
      setLoading(true);

      // Sign up the user
      const userCredential = await signup(emailRef.current.value, passwordRef.current.value);
      const user = userCredential.user;
```

```
      // Store the user information and role in Firestore
      await setDoc(doc(firestore, 'users', user.uid), {
      firstName: firstNameRef.current.value,
      lastName: lastNameRef.current.value,
        email: user.email,
        role: roleRef.current.value,
      });

      // Navigate to corresponding dashboard based on selected role
      if (roleRef.current.value === 'admin') {
      navigate('/dashboard/admin');
      } else if (roleRef.current.value === 'vendor') {
        navigate('/dashboard/vendor');
      } else if (roleRef.current.value === 'hospitalStaff') {
        navigate('/dashboard/hospital');
      } else {
        setError('Unknown role, please contact support.');
      }
    } catch (e) {
      setError('Failed to create account: ' + e.message);
    }
    setLoading(false);
  }

  return (
    <Box sx={{ display: 'flex', height: '100vh' }}>
      {/* Image Section */}
      <Box
        sx={{
        flex: 1,
          backgroundImage: `url(${backgroundImg})`,
          backgroundSize:  'cover',
          backgroundPosition: 'center',
          display: { xs: 'none', md: 'block' }, // Hides on small screens
        }}
      />

      {/* Register Form Section */}
      <Container maxWidth="sm" sx={{ flex: 1, display: 'flex', flexDirection: 'column', justifyContent: 'center',
padding: 4 }}>
        <Box sx={{ display: 'flex', flexDirection: 'column', alignItems: 'center', mb: 4 }}>
          <img src={logo} alt="App Logo" style={{ width: '100px', marginBottom: '16px' }} />
          <Typography
  variant="h5"
  noWrap
  sx={{
    fontFamily: 'Poppins, sans-serif', // Updated to use the Google Fonts CDN
    fontWeight: 600,
    fontSize: '1.8rem',
    letterSpacing: 1,
    color: 'black',
  }}
>
```

```
  MediMatrix
</Typography>

        {error && <Typography color="error" align="center">{error}</Typography>}
      </Box>

      <form onSubmit={handleSubmit} style={{ width: '100%' }}>
        <TextField
          variant="outlined"
          margin="normal"
          required
          fullWidth
          label="First Name"
          inputRef={firstNameRef}
        />
        <TextField
          variant="outlined"
          margin="normal"
          required
          fullWidth
          label="Last Name"
          inputRef={lastNameRef}
        />
        <TextField
          variant="outlined"
          margin="normal"
          required
          fullWidth
          label="Email Address"
          inputRef={emailRef}
        />
        <TextField
          variant="outlined"
          margin="normal"
          required
          fullWidth
          label="Password"
          type="password"
          inputRef={passwordRef}
        />
        <FormControl fullWidth margin="normal">
          <InputLabel id="role-label">Select Role</InputLabel>
          <Select
            labelId="role-label"
            inputRef={roleRef}
            defaultValue=""
            required
          >
            <MenuItem value="admin">Admin</MenuItem>
            <MenuItem value="vendor">Vendor</MenuItem>
            <MenuItem value="hospitalStaff">Hospital Staff</MenuItem>
          </Select>
        </FormControl>
```

```
          <Button
            type="submit"
            fullWidth
            variant="contained"
            color="primary"
            disabled={loading}
            sx={{ marginTop: 2 }}
          >
            Register
          </Button>
        </form>

        <Typography align="center" style={{ marginTop: '16px' }}>
          Already have an account?{' '}
          <Link href="/" variant="body2">Log In</Link>
        </Typography>
      </Container>
    </Box>
  );
}


import React, { useRef, useState } from 'react';
import { useAuth } from '../../contexts/AuthContext';
import { Button, TextField, Container, Typography, Box, Link } from '@mui/material';
import logo from '../../assets/logo.png'; // Logo for branding
import backgroundImg from '../../assets/forgot-password-background.jpg'; // Background image

export default function ForgotPasswordPage() {
  const emailRef = useRef();
  const { resetPassword } = useAuth();
  const [message, setMessage] = useState('');
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);

  async function handleSubmit(e) {
    e.preventDefault();
    try {
      setError('');
      setMessage('');
      setLoading(true);
      await resetPassword(emailRef.current.value);
      setMessage('Check your inbox for further instructions');
    } catch {
      setError('Failed to reset password');
    }
    setLoading(false);
  }

  return (
    <Box sx={{ display: 'flex', height: '100vh' }}>
      {/* Image Section */}
      <Box
        sx={{
```

```
        flex: 1,
        backgroundImage: `url(${backgroundImg})`,
        backgroundSize:  'cover',
        backgroundPosition: 'center',
        display: { xs: 'none', md: 'block' },
       }}
      />

      {/* Forgot Password Form Section */}
      <Container maxWidth="sm" sx={{ flex: 1, display: 'flex', flexDirection: 'column', justifyContent: 'center',
padding: 4 }}>
       <Box sx={{ display: 'flex', flexDirection: 'column', alignItems: 'center', mb: 4 }}>
        <img src={logo} alt="App Logo" style={{ width: '100px', marginBottom: '16px' }} />
        <Typography
  variant="h5"
  noWrap
  sx={{
   fontFamily: 'Poppins, sans-serif', // Updated to use the Google Fonts CDN
   fontWeight: 600,
   fontSize: '1.8rem',
   letterSpacing: 1,
   color: 'red',
  }}
>
  MediMatrix
</Typography>

        {error && <Typography color="error" align="center">{error}</Typography>}
        {message && <Typography color="primary" align="center">{message}</Typography>}
       </Box>

       <form onSubmit={handleSubmit} style={{ width: '100%' }}>
        <TextField
         variant="outlined"
         margin="normal"
         required
         fullWidth
         label="Email Address"
         inputRef={emailRef}
        />
        <Button
         type="submit"
         fullWidth
         variant="contained"
         color="primary"
         disabled={loading}
         sx={{ marginTop: 2 }}
        >
         Reset Password
        </Button>
       </form>

       <Typography align="center" style={{ marginTop: '16px' }}>
        <Link href="/" variant="body2">Back to Login</Link>
```

```
      </Typography>
    </Container>
  </Box>
 );
}


import React, { useState } from "react";
import { useNavigate, Link } from "react-router-dom";
import { firestore } from '../../firebase';
import { collection, addDoc } from "firebase/firestore";
import { Box, Container, Grid, TextField, Button, Typography, Card, CardHeader, CardContent } from
"@mui/material";

export default function AddDrugPage() {
 const navigate = useNavigate();
 const [errorMsg, setErrorMsg] = useState("");
 const [successMsg, setSuccessMsg] = useState("");
 const [medicine, setMedicine] = useState({
  name: "",
  power: "",
  price: "",
  stock: "",
  expiryDate: "",
 });

 const medicinesCollectionRef = collection(firestore, "medicine_inventory");

 // Function to add new medicine to Firestore
 const handleAddMedicine = async () => {
  if (
   medicine.name &&
   medicine.power &&
   medicine.price &&
   medicine.stock &&
   medicine.expiryDate
  ) {
   setErrorMsg("");
   await addDoc(medicinesCollectionRef, {
    name: medicine.name,
    power: medicine.power,
    price: medicine.price,
    stock: medicine.stock,
    expiryDate: medicine.expiryDate,
   });
   setSuccessMsg("Medicine added successfully!");
   setTimeout(() => {
    setSuccessMsg("");
    navigate("/inventory");
   }, 1000);
  } else {
   setErrorMsg("Please fill out all the required fields!");
  }
 };
```

```
return (
  <Box display="flex" flexDirection="column" sx={{ padding: 2 }}>
    <Container>
      <Typography variant="h4" sx={{ marginBottom: 2 }}>Create Medicine</Typography>
      <Grid container spacing={2}>
        <Grid item xs={12}>
          <Card>
            <CardHeader
              title="New Medicine Details"
              action={
                <Link to="/inventory">
                  <Button variant="contained" color="error" size="small">
                    Go BACK
                  </Button>
                </Link>
              }
            />
            <CardContent>
              {/* Medicine Name */}
              <TextField
                fullWidth
                label="Medicine Name"
                variant="outlined"
                value={medicine.name}
                onChange={(event) =>
                  setMedicine((prev) => ({ ...prev, name: event.target.value }))
                }
                sx={{ marginBottom: 2 }}
              />

              {/* Medicine Power */}
              <TextField
                fullWidth
                label="Medicine Power"
                variant="outlined"
                value={medicine.power}
                onChange={(event) =>
                  setMedicine((prev) => ({ ...prev, power: event.target.value }))
                }
                sx={{ marginBottom: 2 }}
              />

              {/* Medicine Price */}
              <TextField
                fullWidth
                label="Medicine Price (in ₹.)"
                variant="outlined"
                value={medicine.price}
                onChange={(event) =>
                  setMedicine((prev) => ({ ...prev, price: event.target.value }))
                }
                sx={{ marginBottom: 2 }}
              />
```

```
            {/* Medicine Stock */}
            <TextField
              fullWidth
              label="Medicine Stock"
              variant="outlined"
              value={medicine.stock}
              onChange={(event) =>
                setMedicine((prev) => ({ ...prev, stock: event.target.value }))
              }
              sx={{ marginBottom: 2 }}
            />

            {/* Expiry Date */}
            <TextField
              fullWidth
              type="date"
              label="Expiry Date"
              variant="outlined"
              value={medicine.expiryDate}
              onChange={(event) =>
                setMedicine((prev) => ({ ...prev, expiryDate: event.target.value }))
              }
              sx={{ marginBottom: 2 }}
            />
          </CardContent>

          {/* Success/Error Messages */}
          <Box textAlign="center" sx={{ padding: 2 }}>
            {errorMsg && <Typography color="error">{errorMsg}</Typography>}
            {successMsg && <Typography color="success.main">{successMsg}</Typography>}
            <Button variant="contained" color="primary" onClick={handleAddMedicine}>
              Add Medicine
            </Button>
          </Box>
        </Card>
      </Grid>
    </Grid>
  </Container>
 </Box>
);
}


import React, { useState, useEffect, useCallback } from "react";
import { Link, useNavigate } from "react-router-dom";
import { collection, getDocs, doc, deleteDoc } from "firebase/firestore";
import { firestore } from '../../firebase';
import * as XLSX from 'xlsx';
import {
 Container,
 Card,
 CardContent,
 Typography,
```

```
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  Button,
  IconButton,
  Box,
} from "@mui/material";
import { Edit, Delete } from "@mui/icons-material";

export default function InventoryOverview() {
  const [medicines, setMedicines] = useState([]);
  const medicinesCollectionRef = collection(firestore, "medicine_inventory");
  let counter = 1;

  const navigate = useNavigate();

  // Fetch medicine data from Firestore
  const getTypes = useCallback(async () => {
    const data = await getDocs(medicinesCollectionRef);
    setMedicines(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })));
  }, [medicinesCollectionRef]);

  // Handle deletion of medicine
  const handleDeleteButton = async (id) => {
    const medDoc = doc(firestore, "medicine_inventory", id);
    await deleteDoc(medDoc);
    getTypes(); // Refresh the data after deletion
  };

  // Fetch medicines on component mount
  useEffect(() => {
    getTypes();
  }, [getTypes]);

  // Handle edit action by storing the selected medicine in local storage
  const handleEditButton = (medicine) => {
  localStorage.setItem("medicine_obj", JSON.stringify(medicine));
  navigate("/inventory/edit"); // Navigate to the update page
  };

  // Export inventory data to Excel
  const handleExportToExcel = () => {
    const worksheet = XLSX.utils.json_to_sheet(
      medicines.map((medicine) => ({
        Name: medicine.name,
        Power: medicine.power,
        Price: medicine.price,
        Stock: medicine.stock,
        "Expiry Date": medicine.expiryDate,
      }))
    );
```

```
    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheet, "Medicine Inventory");

    // Generate an Excel file and download it
    XLSX.writeFile(workbook, "Medicine_Inventory.xlsx");
  };

  return (
    <div style={{ display: 'flex', flexDirection: 'column', padding: '20px' }}>
      <Container maxWidth="xl">
        <Typography variant="h4" gutterBottom sx={{ fontFamily: 'Poppins, sans-serif', fontWeight: 700, color:
'#3f51b5' }}>
          Medicine Inventory
        </Typography>
        <Card sx={{ boxShadow: 3 }}>
          <CardContent>
            <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', marginBottom: 3 }}>
              <Typography variant="h6" component="div" sx={{ fontFamily: 'Poppins, sans-serif', fontWeight: 600
}}>
                Inventory List
              </Typography>
              <Box>
                <Button
                  component={Link}
                  to="/inventory/add"
                  variant="contained"
                  color="primary"
                  sx={{ marginLeft: 2 }}
                >
                  Add New Medicine
                </Button>
                <Button
                  variant="contained"
                  color="secondary"
                  sx={{ marginLeft: 2 }}
                  onClick={handleExportToExcel}
                >
                  Export to Excel
                </Button>
              </Box>
            </Box>
            <TableContainer sx={{ boxShadow: 2 }}>
              <Table>
                <TableHead>
                  <TableRow>
                    <TableCell sx={{ fontWeight: 'bold' }}>#</TableCell>
                    <TableCell sx={{ fontWeight: 'bold' }}>Medicine Name<sup>Power</sup></TableCell>
                    <TableCell sx={{ fontWeight: 'bold' }}>Medicine Price</TableCell>
                    <TableCell sx={{ fontWeight: 'bold' }}>Stock</TableCell>
                    <TableCell sx={{ fontWeight: 'bold' }}>Expiry Date</TableCell>
                    <TableCell sx={{ fontWeight: 'bold' }}>Action</TableCell>
                  </TableRow>
                </TableHead>
                <TableBody>
```

```
              {medicines.map((medicine) => (
                <TableRow key={medicine.id}>
                  <TableCell>{counter++}</TableCell>
                  <TableCell>{medicine.name} <sup>{medicine.power}</sup></TableCell>
                  <TableCell>₹{medicine.price}</TableCell>
                  <TableCell>{medicine.stock}</TableCell>
                  <TableCell>{medicine.expiryDate}</TableCell>
                  <TableCell>
                    <IconButton
                      color="success"
                      onClick={() => handleEditButton(medicine)}
                    >
                      <Edit />
                    </IconButton>
                    <IconButton
                      color="error"
                      onClick={() => handleDeleteButton(medicine.id)}
                    >
                      <Delete />
                    </IconButton>
                  </TableCell>
                </TableRow>
              ))}
            </TableBody>
          </Table>
        </TableContainer>
      </CardContent>
    </Card>
  </Container>
 </div>
 );
}


import React, { useEffect, useState } from 'react';
import { Container, Typography, Paper, Box, Button } from '@mui/material';
import { collection, getDocs } from 'firebase/firestore';
import { firestore } from '../../firebase';
import * as XLSX from 'xlsx'; // Import xlsx library

const OrderListPage = () => {
  const [orders, setOrders] = useState([]);

  useEffect(() => {
    const fetchOrders = async () => {
      try {
        const ordersCollection = collection(firestore, 'orders');
        const snapshot = await getDocs(ordersCollection);
        const ordersData = snapshot.docs.map(doc => {
          const data = doc.data();
          return {
            id: doc.id,
            ...data,
            createdAt: data.createdAt ? data.createdAt.toDate() : null, // Convert Firestore Timestamp to Date
```

```
        };
      });
      setOrders(ordersData);
    } catch (error) {
      console.error("Error fetching orders: ", error);
    }
  };

  fetchOrders();
}, []);

const exportToExcel = () => {
  // Create a worksheet from the orders data
  const ws = XLSX.utils.json_to_sheet(orders);

  // Create a new workbook and add the worksheet
  const wb = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(wb, ws, 'Orders');

  // Export the workbook to an Excel file
  XLSX.writeFile(wb, 'orders.xlsx');
};

return (
  <Container maxWidth="md" sx={{ padding: 4, fontFamily: '"Poppins", sans-serif' }}>
    <Paper elevation={3} sx={{ padding: 3, borderRadius: 3, boxShadow: '0 4px 8px rgba(0, 0, 0, 0.1)',
backgroundColor: '#f8f8f8' }}>
      <Typography variant="h4" sx={{ marginBottom: 3, fontWeight: 'bold', textAlign: 'center', color: '#333'
}}>
        Order List
      </Typography>

      {/* Export Button */}
      <Button
        variant="contained"
        color="primary"
        onClick={exportToExcel}
        sx={{ marginBottom: 3 }}
      >
        Export to Excel
      </Button>

      {orders.length > 0 ? (
        orders.map(order => (
          <Paper key={order.id} sx={{ marginBottom: 3, padding: 3, borderRadius: 2, backgroundColor: '#fff',
boxShadow: '0 2px 5px rgba(0, 0, 0, 0.1)' }}>
            <Typography variant="h6" sx={{ fontWeight: '600', color: '#2c6bed' }}>Order ID:
{order.id}</Typography>
            <Box sx={{ marginTop: 1 }}>
              <Typography variant="body1">Drug Name: <span style={{ fontWeight: '500', color: '#555'
}}>{order.drugName}</span></Typography>
              <Typography variant="body1">Quantity: <span style={{ fontWeight: '500', color: '#555'
}}>{order.quantity}</span></Typography>
              <Typography variant="body1">Vendor Name: <span style={{ fontWeight: '500', color: '#555'
```

```
}}>{order.vendorName}</span></Typography>
            <Typography variant="body1">Status: <span style={{ fontWeight: '500', color: '#555'
}}>{order.status}</span></Typography>
            <Typography variant="body1">
              Created At: <span style={{ fontWeight: '500', color: '#555' }}>
                {order.createdAt ? order.createdAt.toLocaleString() : "N/A"}
              </span>
            </Typography>
          </Box>
        </Paper>
      ))
    ) : (
      <Typography variant="body1" sx={{ textAlign: 'center', fontStyle: 'italic' }}>No orders
found.</Typography>
    )}
  </Paper>
 </Container>
);
};

export default OrderListPage;


import React from 'react';
import { useParams } from 'react-router-dom';
import { Container, Typography } from '@mui/material';
import { useOrders } from '../../contexts/OrderContext';

export default function OrderTrackingPage() {
 const { id } = useParams();
 const { getOrderById } = useOrders();
 const order = getOrderById(id);

 if (!order) return <Typography>Loading...</Typography>;

 return (
  <Container>
   <Typography variant="h4">Order Tracking</Typography>
   <Typography variant="h6">Order ID: {order.id}</Typography>
   <Typography>Status: {order.status}</Typography>
   {/* Additional tracking details */}
  </Container>
 );
}


import React, { useState, useEffect } from 'react';
import { Container, Typography, Paper, Button, CircularProgress, Snackbar } from '@mui/material';
import { firestore } from '../../firebase';
import { collection, query, where, getDocs } from 'firebase/firestore';
import { useAuth } from '../../contexts/AuthContext';
import { useShipments } from '../../contexts/ShipmentContext';
import { format } from 'date-fns';
import * as XLSX from 'xlsx'; // Import xlsx library
```

```
const VendorOrdersPage = () => {
  const { currentUser } = useAuth();
  const vendorId = currentUser?.uid;
  const [vendorOrders, setVendorOrders] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [successMessage, setSuccessMessage] = useState('');
  const { handleCreateShipment } = useShipments();

  useEffect(() => {
    const fetchVendorOrders = async () => {
      if (!vendorId) return;
      try {
        const ordersCollection = collection(firestore, 'orders');
        const ordersQuery = query(ordersCollection, where('vendorId', '==', vendorId));
        const ordersSnapshot = await getDocs(ordersQuery);
        const filteredOrders = ordersSnapshot.docs
          .map(doc => ({ id: doc.id, ...doc.data() }));
        setVendorOrders(filteredOrders);
      } catch (error) {
        console.error("Error fetching vendor orders: ", error);
        setError("Failed to fetch orders");
      } finally {
        setLoading(false);
      }
    };

    fetchVendorOrders();
  }, [vendorId]);

  const handleCreateShipmentClick = async (order) => {
    try {
      await handleCreateShipment(order);
      setSuccessMessage('Shipment successfully created!');
    } catch (error) {
      setError("Failed to create shipment");
      console.error("Error creating shipment: ", error);
    }
  };

  const exportToExcel = () => {
    const formattedOrders = vendorOrders.map(order => ({
      OrderID: order.id,
      DrugName: order.drugName || "N/A",
      Quantity: order.quantity || "N/A",
      Status: order.status || "Pending",
      OrderedAt: order.createdAt ? format(order.createdAt.toDate(), 'MM/dd/yyyy HH:mm') : "N/A"
    }));

    const ws = XLSX.utils.json_to_sheet(formattedOrders);
    const wb = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, 'Vendor Orders');
```

```
  // Generate Excel file and trigger download
  XLSX.writeFile(wb, 'Vendor_Orders.xlsx');
 };

 if (loading) {
  return (
   <Container sx={{ display: 'flex', justifyContent: 'center', alignItems: 'center', height: '80vh' }}>
    <CircularProgress />
    <Typography sx={{ marginLeft: 2 }}>Loading orders...</Typography>
   </Container>
  );
 }

 return (
  <Container sx={{ padding: 4 }}>
   <Paper elevation={3} sx={{ padding: 4, borderRadius: 2, backgroundColor: '#f5f5f5' }}>
    <Typography variant="h4" sx={{ fontWeight: 600, marginBottom: 3 }}>Orders Received</Typography>

    {/* Export Button */}
    <Button
     variant="contained"
     onClick={exportToExcel}
     sx={{ marginBottom: 3 }}
    >
     Export to Excel
    </Button>

    {error && <Typography color="error" sx={{ marginBottom: 2 }}>{error}</Typography>}
    {vendorOrders.length > 0 ? (
     vendorOrders.map(order => (
      <Paper
       key={order.id}
       sx={{
       marginBottom: 2,
        padding: 3,
        backgroundColor: '#fff',
        borderRadius: 2,
        boxShadow: '0px 4px 12px rgba(0, 0, 0, 0.1)',
        transition: 'all 0.3s ease',
        '&:hover': {
         boxShadow: '0px 8px 24px rgba(0, 0, 0, 0.15)',
         backgroundColor: '#f1f1f1',
        },
       }}
      >
       <Typography variant="h6" sx={{ fontWeight: 500 }}>Order ID: {order.id}</Typography>
       <Typography sx={{ fontWeight: 500, color: '#1976d2' }}>Drug Name: {order.drugName ||
"N/A"}</Typography>
       <Typography>Quantity: {order.quantity || "N/A"}</Typography>
       <Typography>Status: {order.status || 'Pending'}</Typography>
       <Typography sx={{ color: 'textSecondary' }}>
        Ordered At:{" "}
        {order.createdAt ? format(order.createdAt.toDate(), 'MM/dd/yyyy HH:mm') : "N/A"}
       </Typography>
```

```
            {order.status === 'Pending' && (
              <Button
                variant="contained"
                sx={{ marginTop: 2 }}
                onClick={() => handleCreateShipmentClick(order)}
              >
                Create Shipment
              </Button>
            )}
          </Paper>
        ))
      ) : (
        <Typography sx={{ fontStyle: 'italic', color: '#888' }}>No orders received yet.</Typography>
      )}
    </Paper>

    {successMessage && (
      <Snackbar
        open={true}
        autoHideDuration={3000}
        onClose={() => setSuccessMessage('')}
        message={successMessage}
      />
    )}
  </Container>
 );
};

export default VendorOrdersPage;


// src/pages/reports/ReportsOverviewPage.js

import React from 'react';
import { Container, Typography, Grid, Paper } from '@mui/material';
import InventoryChart from './InventoryChart';
import VendorPerformanceChart from './VendorPerformanceChart';
import OrderChart from './OrderChart';

export default function ReportsOverviewPage() {
 return (
  <Container>
   <Typography variant="h4">Reports Overview</Typography>
   <Grid container spacing={3}>
    <Grid item xs={12} sm={6} md={4}>
     <Paper elevation={3}>
      <Typography variant="h6">Inventory Reports</Typography>
      <InventoryChart />
     </Paper>
    </Grid>
    <Grid item xs={12} sm={6} md={4}>
     <Paper elevation={3}>
      <Typography variant="h6">Vendor Performance</Typography>
      <VendorPerformanceChart />
```

```
        </Paper>
      </Grid>
      <Grid item xs={12} sm={6} md={4}>
        <Paper elevation={3}>
          <Typography variant="h6">Order Reports</Typography>
          <OrderChart />
        </Paper>
      </Grid>
    </Grid>
  </Container>
 );
}


import React, { useState } from 'react';
import { Container, Typography, Grid, Paper, Button, TextField, Dialog, DialogActions, DialogContent,
DialogTitle } from '@mui/material';
import { useShipments } from '../../contexts/ShipmentContext';
import * as XLSX from 'xlsx'; // Import the xlsx library

export default function ShipmentListPage() {
 const { shipments, updateShipmentStatus } = useShipments();
 const [open, setOpen] = useState(false);
 const [selectedShipment, setSelectedShipment] = useState(null);
 const [newStatus, setNewStatus] = useState('');

 const handleClickOpen = (shipment) => {
  setSelectedShipment(shipment);
  setNewStatus(shipment.status); // Set the current status as the default in the dialog
  setOpen(true);
 };

 const handleClose = () => {
  setOpen(false);
  setSelectedShipment(null);
  setNewStatus('');
 };

 const handleUpdateStatus = async () => {
  if (selectedShipment && newStatus) {
  try {
     // Call the function to update the status
     await updateShipmentStatus(selectedShipment.id, newStatus, selectedShipment.estimatedDelivery);
     handleClose();
    } catch (error) {
     console.error('Error updating shipment status:', error);
    }
  } else {
    console.error('Please provide a new status');
  }
 };

 // Function to export shipments to Excel
 const exportToExcel = () => {
```

```
  const ws = XLSX.utils.json_to_sheet(shipments); // Convert shipments to worksheet
  const wb = XLSX.utils.book_new(); // Create a new workbook
  XLSX.utils.book_append_sheet(wb, ws, 'Shipments'); // Append the sheet to the workbook
  XLSX.writeFile(wb, 'shipments.xlsx'); // Download the file as 'shipments.xlsx'
};

return (
  <Container sx={{ padding: 4 }}>
    <Typography variant="h4" sx={{ fontWeight: 600, marginBottom: 4 }}>Shipments</Typography>

    {/* Export Button */}
    <Button
      variant="contained"
      sx={{
      marginBottom: 3,
        backgroundColor: '#1976d2',
        '&:hover':  {
        backgroundColor: '#1565c0',
        },
      }}
      onClick={exportToExcel} // Trigger export on click
    >
      Export to Excel
    </Button>

    <Grid container spacing={3}>
      {shipments.map((shipment) => (
        <Grid item xs={12} sm={6} md={4} key={shipment.id}>
          <Paper
            elevation={3}
            sx={{
            padding: 3,
              backgroundColor: '#fff',
              borderRadius: 2,
              boxShadow: '0px 4px 12px rgba(0, 0, 0, 0.1)',
              transition: 'all 0.3s ease',
              '&:hover': {
                boxShadow: '0px 6px 18px rgba(0, 0, 0, 0.2)',
                backgroundColor: '#f9f9f9',
              },
            }}
          >
            <Typography variant="h6" sx={{ fontWeight: 500 }}>Shipment ID: {shipment.id}</Typography>
            <Typography variant="body1">Order ID: {shipment.orderId}</Typography>
            <Typography variant="body1" sx={{ fontWeight: 500 }}>Status: {shipment.status}</Typography>
            <Typography variant="body2" sx={{ color: 'textSecondary' }}>
              Estimated Delivery: {shipment.estimatedDelivery || "Not Available"}
            </Typography>
            <Button
              variant="contained"
              sx={{
                marginTop: 2,
                backgroundColor: '#1976d2',
                '&:hover': {
```

```
                backgroundColor: '#1565c0',
              },
            }}
            onClick={() => handleClickOpen(shipment)}
          >
            Update Status
          </Button>
        </Paper>
      </Grid>
    ))}
  </Grid>

  <Dialog open={open} onClose={handleClose}>
    <DialogTitle>Update Shipment Status</DialogTitle>
    <DialogContent>
      <TextField
        autoFocus
        margin="dense"
        label="New Status"
        type="text"
        fullWidth
        variant="outlined"
        value={newStatus}
        onChange={(e) => setNewStatus(e.target.value)}
        sx={{ marginBottom: 2 }}
      />
      <TextField
        margin="dense"
        label="Estimated Delivery"
        type="text"
        fullWidth
        variant="outlined"
        value={selectedShipment?.estimatedDelivery || ""}
        onChange={(e) => setSelectedShipment({
          ...selectedShipment, estimatedDelivery: e.target.value })}
      />
    </DialogContent>
    <DialogActions>
      <Button onClick={handleClose} sx={{ color: '#1976d2' }}>Cancel</Button>
      <Button
        onClick={handleUpdateStatus}
        sx={{
          backgroundColor: '#1976d2',
          '&:hover':  {
          backgroundColor: '#1565c0',
          },
        }}
      >
        Update
      </Button>
    </DialogActions>
  </Dialog>
 </Container>
);
```

```
}

import React, { useEffect, useState } from 'react';
import { Container, Typography, CircularProgress, Paper, Button } from '@mui/material';
import { Timeline, TimelineItem, TimelineSeparator, TimelineConnector, TimelineContent, TimelineDot }
from '@mui/lab';
import { useShipments } from '../../contexts/ShipmentContext';
import { useOrders } from '../../contexts/OrderContext';
import * as XLSX from 'xlsx';
import { saveAs } from 'file-saver';

export default function TrackShipmentPage() {
  const { shipments } = useShipments();
  const { getOrderById } = useOrders();
  const [loading, setLoading] = useState(true);
  const [shipmentDetails, setShipmentDetails] = useState([]);

  useEffect(() => {
    const fetchShipmentDetails = () => {
      const details = shipments.map((shipment) => {
        const orderDetails = getOrderById(shipment.orderId);
        return {
          ...shipment,
          drugName: orderDetails?.drugName || 'Unknown Drug',
          vendorName: orderDetails?.vendorName || 'Unknown Vendor',
        };
      });
      setShipmentDetails(details);
      setLoading(false);
    };

    if (shipments && shipments.length > 0) {
      fetchShipmentDetails();
    }
  }, [shipments, getOrderById]);

  if (loading) {
    return (
      <Container>
        <CircularProgress />
        <Typography>Loading shipment details...</Typography>
      </Container>
    );
  }

  const getStatusSteps = (status) => {
    const allStatuses = ["Order Placed", "Shipped", "In Transit", "Out for Delivery", "Delivered"];
    return allStatuses.slice(0, allStatuses.indexOf(status) + 1);
  };

  const exportToExcel = () => {
    const shipmentData = shipmentDetails.map((shipment) => ({
      ShipmentID: shipment.id,
      OrderID: shipment.orderId,
```

```
    DrugName: shipment.drugName,
    VendorName: shipment.vendorName,
    Status: shipment.status,
    EstimatedDelivery: shipment.estimatedDelivery || 'Not Available',
   }));

   const ws = XLSX.utils.json_to_sheet(shipmentData);
   const wb = XLSX.utils.book_new();
   XLSX.utils.book_append_sheet(wb, ws, 'Shipments');

   const excelBuffer = XLSX.write(wb, { bookType: 'xlsx', type: 'array' });
   const data = new Blob([excelBuffer], { bookType: 'xlsx', type: 'application/octet-stream' });

   saveAs(data, 'shipments.xlsx');
  };

  return (
   <Container>
    <Typography variant="h4" gutterBottom sx={{ fontWeight: 600, marginBottom: 2 }}>
     Shipment Tracking
    </Typography>
    <Button
     variant="contained"
     color="primary"
     onClick={exportToExcel}
     sx={{ marginBottom: 3 }}
    >
     Export to Excel
    </Button>

    {shipmentDetails.map((shipment) => (
     <Paper
      key={shipment.id}
      sx={{
       marginBottom: '2rem',
       padding: '1rem',
       border: '1px solid #ddd',
       borderRadius: '8px',
       transition: 'all 0.3s ease',
       '&:hover': {
        boxShadow: '0px 4px 12px rgba(0, 0, 0, 0.1)', // Hover shadow
        backgroundColor: '#f9f9f9', // Hover background color
       },
      }}
     >
      <Typography variant="h6" sx={{ fontWeight: 500 }}>Shipment ID: {shipment.id}</Typography>
      <Typography variant="body1" sx={{ marginBottom: 1 }}>Order ID:
{shipment.orderId}</Typography>
      <Typography variant="body1" sx={{ marginBottom: 1 }}>Drug Name:
{shipment.drugName}</Typography>
      <Typography variant="body1" sx={{ marginBottom: 1 }}>Vendor Name:
{shipment.vendorName}</Typography>
      <Typography variant="body2" color="textSecondary">Estimated Delivery:
{shipment.estimatedDelivery || "Not Available"}</Typography>
```

```
        <Timeline align="left" sx={{ marginTop: '1rem' }}>
          {getStatusSteps(shipment.status).map((statusStep, index) => (
            <TimelineItem key={index}>
              <TimelineSeparator>
                <TimelineDot
                  color={index === getStatusSteps(shipment.status).length - 1 ? "success" : "primary"}
                  sx={{
                    '&:hover': {
                      backgroundColor: '#1976d2', // Hover color change for timeline dot
                      transform: 'scale(1.2)', // Hover scale effect
                    },
                    transition: 'all 0.3s ease',  // Smooth transition for hover effects
                  }}
                />
                {index < getStatusSteps(shipment.status).length - 1 && <TimelineConnector />}
              </TimelineSeparator>
              <TimelineContent>
                <Typography variant="body1" sx={{ fontWeight: 400 }}>{statusStep}</Typography>
              </TimelineContent>
            </TimelineItem>
          ))}
        </Timeline>
      </Paper>
    ))}
  </Container>
 );
}
```

Package.json

```
{
 "name": "medimatrix",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
  "@emotion/react": "^11.13.3",
  "@emotion/styled": "^11.13.0",
  "@fontsource/poppins": "^5.1.0",
  "@mui/icons-material": "^6.1.2",
  "@mui/lab": "^6.0.0-beta.13",
  "@mui/material": "^6.1.1",
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "chart.js": "^4.4.4",
  "date-fns": "^4.1.0",
  "file-saver": "^2.0.5",
  "firebase": "^10.14.0",
  "react": "^18.3.1",
  "react-chartjs-2": "^5.2.0",
```

```
    "react-dom": "^18.3.1",
    "react-router-dom": "^6.26.2",
    "react-scripts": "5.0.1",
    "recharts": "^2.13.0",
    "web-vitals": "^2.1.4",
    "xlsx": "^0.18.5"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
    ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "@babel/plugin-proposal-private-property-in-object": "^7.21.11"
  }
}
```

CHAPTER 11

TESTING

11.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

11.2 Types of Testing

11.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

11.2.2 Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

11.2.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT

The future scope of the **Drug Inventory and Supply Chain Tracking System** lies in the integration of cutting-edge technologies such as Artificial Intelligence (AI) and Machine Learning (ML) to predict drug demand patterns and optimize inventory management. By leveraging AI algorithms, the system can predict potential stock shortages or expirations, enabling proactive restocking. Additionally, the incorporation of **Blockchain** technology can ensure the authenticity and traceability of pharmaceutical products across the supply chain, enhancing transparency and combating counterfeit drugs.

Furthermore, **Internet of Things (IoT)** integration could allow real-time monitoring of drug storage conditions, ensuring that drugs are maintained within optimal temperature and humidity levels. This would significantly improve the quality and safety of drugs being transported and stored.

The system can also be enhanced to include **mobile app support** for better accessibility, allowing vendors, hospitals, and administrators to monitor the supply chain, place orders, and receive notifications on-the-go.

Another potential enhancement is the **customer-facing portal**, which could allow healthcare institutions to access reports, track drug shipments, and receive real-time notifications. Additionally, the system could incorporate **predictive analytics** to help hospitals forecast future drug requirements based on historical consumption data, improving procurement strategies.

In summary, the future of this system includes the integration of smart technologies to improve efficiency, reduce waste, enhance drug safety, and provide a more personalized, user-friendly experience for healthcare providers.

CONCLUSION & REFERNCES

The successful design and development of the **Drug Inventory and Supply Chain Tracking System** represents a significant milestone in optimizing pharmaceutical supply chains. This project has provided invaluable insights into the complexities of healthcare logistics and inventory management. By utilizing modern technologies such as React.js, Firebase, and Material UI, we have built a robust system capable of tracking inventory, managing orders, and ensuring the timely delivery of pharmaceutical products. Through this project, we have refined our skills in full-stack development and learned how to integrate real-time data processing, enhancing both user experience and operational efficiency.

As we move forward, we see tremendous potential for further enhancements, such as AI-powered predictive analytics for demand forecasting, IoT-based monitoring for drug storage conditions, and blockchain for enhanced traceability. The system's adaptability will enable future updates and integrations, ensuring its relevance as the pharmaceutical industry evolves. Our commitment to continuous improvement and innovation will guide us in pushing the boundaries of what is possible in drug supply chain management, ultimately aiming to improve healthcare delivery and patient outcomes.

References

Coding Phase:

1. **React Documentation**: https://react.dev/learn
2. **W3Schools React tutorial:** https://www.w3schools.com/react/
3. **Firebase Documentation**: https://firebase.google.com/docs
4. **Material UI Documentation**: https://mui.com/
5. **NetworkX Library**: https://networkx.github.io