

# **MediMatrix**

## **A PROJECT REPORT**

**for**

**Project (KCA451)**

**Session (2024-25)**

**Submitted by**

**Abhishek Yadav**

**University Roll No 2300290140009**

**Arjun Srivastava**

**University Roll No 2300290140039**

**Aryan Sain**

**University Roll No 2300290140041**

**Ashraf Khan**

**University Roll No 2300290140042**

**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**

**(MAY 2025)**

# CERTIFICATE

Certified that **Abhishek Yadav (2300290140009), Arjun Srivastava (2300290140039), Aryan Sain (2300290140041), Ashraf Khan (2300290140042)** have carried out the project work having “**MediMatrix**” (**Project-KCA451**) 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/herself 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.

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**  
**Ghaziabad**

**Dr. Akash Rajak**  
**Dean**  
**Department of Computer Applications**  
**KIET Group of Institutions,**

## **MediMatrix ABSTRACT**

The MediMatrix, 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 MediMatrix 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 MediMatrix 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 Salgorta** 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. Akash Rajak**, Professor and Dean, 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 kind 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.

**Abhishek Yadav (2300290140009)**

**Arjun Srivastava (230029014039)**

**Aryan Sain (2300290140041)**

**Ashraf Khan (2300290140042)**

# TABLE OF CONTENTS

	Certificate	ii
	Abstract	iii
	Acknowledgements	iv
	Table of Contents	v-vi
	List of Figures	vii
<b>1</b>	<b>Introduction</b>	<b>1-2</b>
1.1	Background	1
1.2	Project overview	1
1.3	Objective	1
1.4	Key features	2
1.5	Scope of the project	2
<b>2</b>	<b>Problem identification &amp; feasibility study</b>	<b>3-4</b>
2.1	Problem Identification	3
2.2	Feasibility Study	3
	2.2.1 Technical Feasibility	3
	2.2.2 Operational Feasibility	3
	2.2.3 Economic Feasibility	4
<b>3</b>	<b>Requirement Analysis</b>	<b>5-6</b>
3.1	Introduction	5
3.2	Functional Requirements	5
	3.2.1 User Authentication	5
	3.2.2 Inventory Management	5
	3.2.3 Order Management	5
3.3	Non-Functional Requirements	5
	3.3.1 Performance	5
	3.3.2 Security	6
	3.3.3 Scalability	6
	3.3.4 Usability	6
<b>4</b>	<b>Project Planning and Scheduling</b>	<b>7</b>
4.1	Pert Chart	7
<b>5</b>	<b>Hardware and Software Specification</b>	<b>8-9</b>
5.1	Hardware Specification	8
5.2	Software Specification	9
<b>6</b>	<b>Choice of Tools &amp; Technology</b>	<b>10-14</b>
6.1	React	10
6.2	Data Flow Diagram	11

6.3	Context Level Diagram	11-13
6.4	Use Case Diagram	14
<b>7</b>	<b>ER-Diagram</b>	<b>15-16</b>
7.1	Entity-relationship model	15
7.2	Class Diagram	16
<b>8</b>	<b>Database</b>	<b>17-19</b>
8.1	Admin	17
8.2	User	18
8.3	Packages	19
<b>9</b>	<b>Form Design</b>	<b>20-22</b>
9.1	Login	20
9.2	Signup	20
9.3	Admin Dashboard	21
9.4	Reports Page	22
9.5	Inventory Management Page	22
<b>10</b>	<b>Coding</b>	<b>23-64</b>
<b>11</b>	<b>Testing</b>	<b>65-66</b>
11.1	Introduction	65
11.2	Types of Testing	66
<b>12</b>	<b>Further Scope and Further Enhancement of the Project</b>	<b>67</b>
<b>13</b>	<b>Conclusion</b>	<b>68</b>
<b>14</b>	<b>References</b>	<b>69</b>

# LIST OF FIGURES

<b>Figure No.</b>	<b>Name of Figure</b>	<b>Page No.</b>
4.1	Pert Chart	7
6.1	0 Level DFD	12
6.2	1 Level DFD	12
6.3	2 <sup>nd</sup> Level DFD	13
6.4	Use Case Diagram	14
7.1	Entity-relationship Model	15
8.1	Admin	17
8.2	User	18
8.3	Packages	19
9.1	Login Page	20
9.2	Sign Up Page	20
9.3	Dashboard page	21
9.4	Reports page	22
9.5	Inventory management page	22

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Background**

The MediMatrix 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 MediMatrix 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 MediMatrix 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 MediMatrix 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 analyse medication usage, empowering staff to make informed decisions for better resource management.

#### 1.4 Key Features

MediMatrix 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 MediMatrix 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 & FEAIBILITY STUDY**

#### **2.1 Problem Identification**

Identifying potential problems in a MediMatrix 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 MediMatrix, 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 MediMatrix 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 MediMatrix 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.

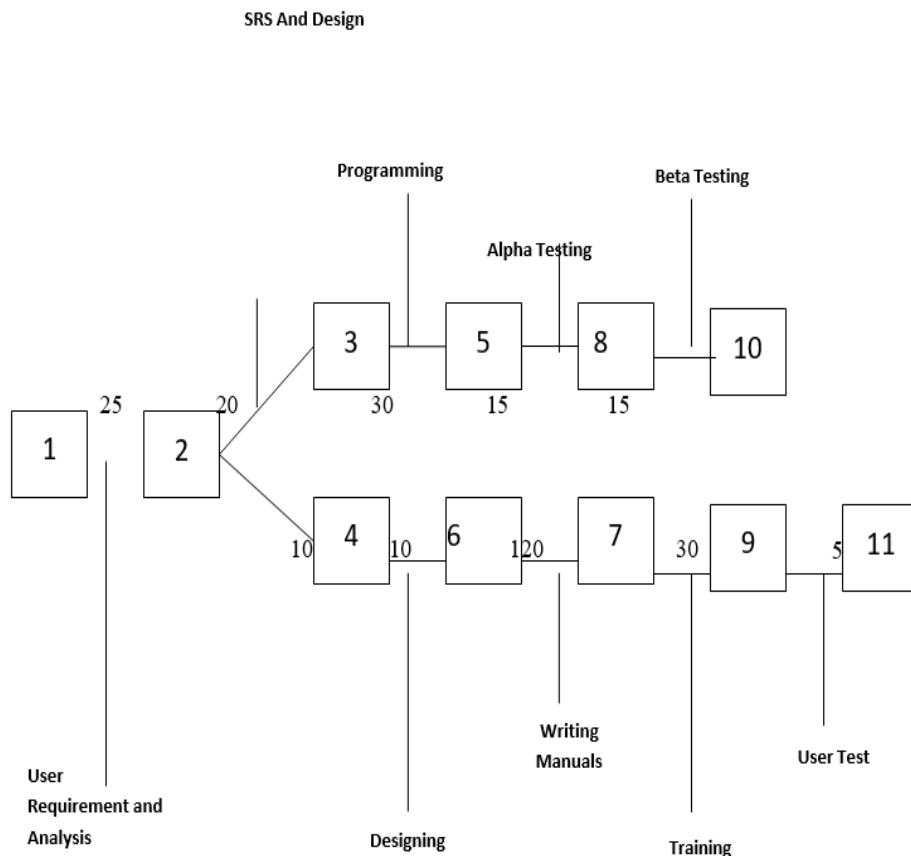
## CHAPTER 4

### PROJECT PLANNING AND SCHEDULING

#### 4.1 Pert Chart:

A PERT chart is a project management tool 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.



**Fig 4.1 Pert Chart**

## **CHAPTER 5**

### **HARDWARE & SOFTWARE SPECIFICATION**

#### **5.1 Hardware Specification**

The MediMatrix 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 MediMatrix 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 re-rendering.

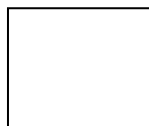
React uses JSX, a syntax extension that looks similar to XML or HTML, to describe the structure of UI components in a more concise and readable way. React components can manage their internal state, and data can be passed to components through props, allowing for dynamic and interactive UIs

#### **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.

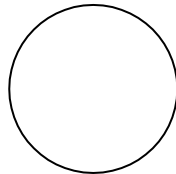
1. A square defines a source (originator) or destination of system data.



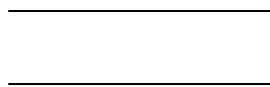
2. An arrow identifies data flow-data in motion. It is a pipeline through which information flows.



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

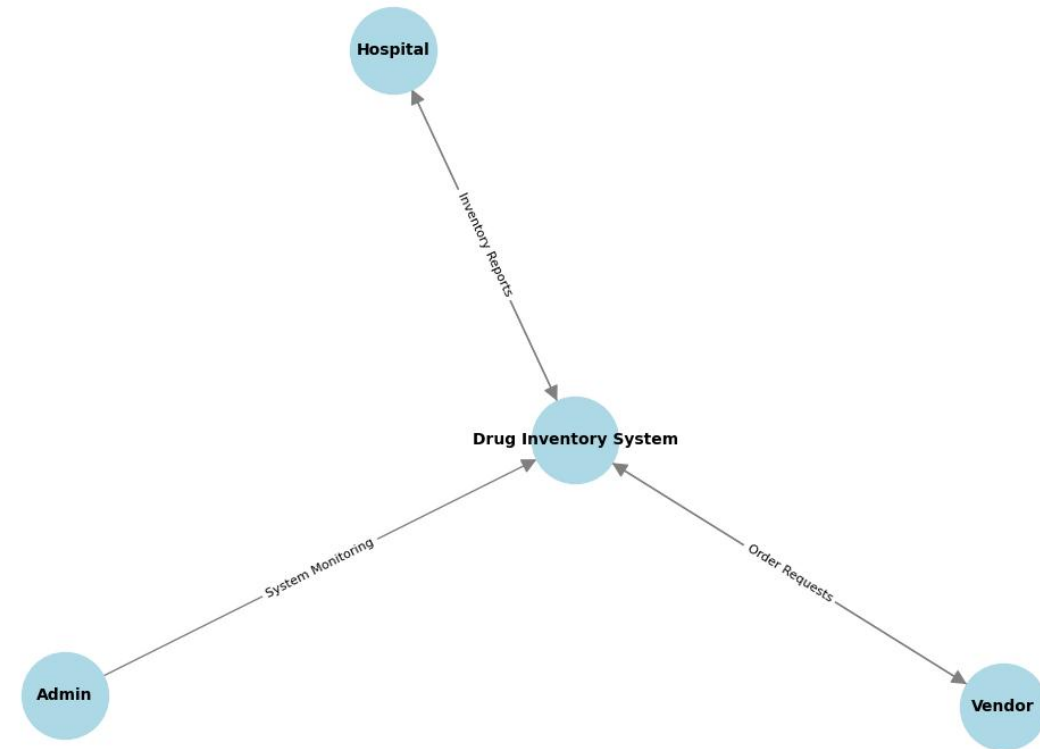


4. An open rectangle is a data store-data at rest, or a temporary repository of data.

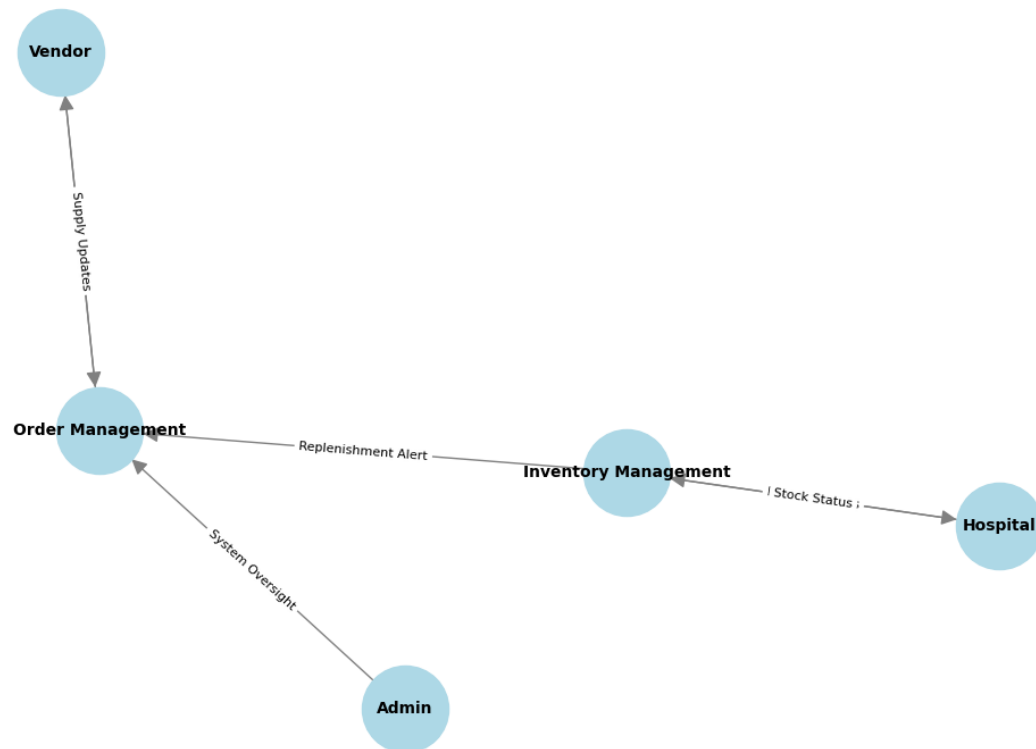


### 6.3 Context Level Diagram

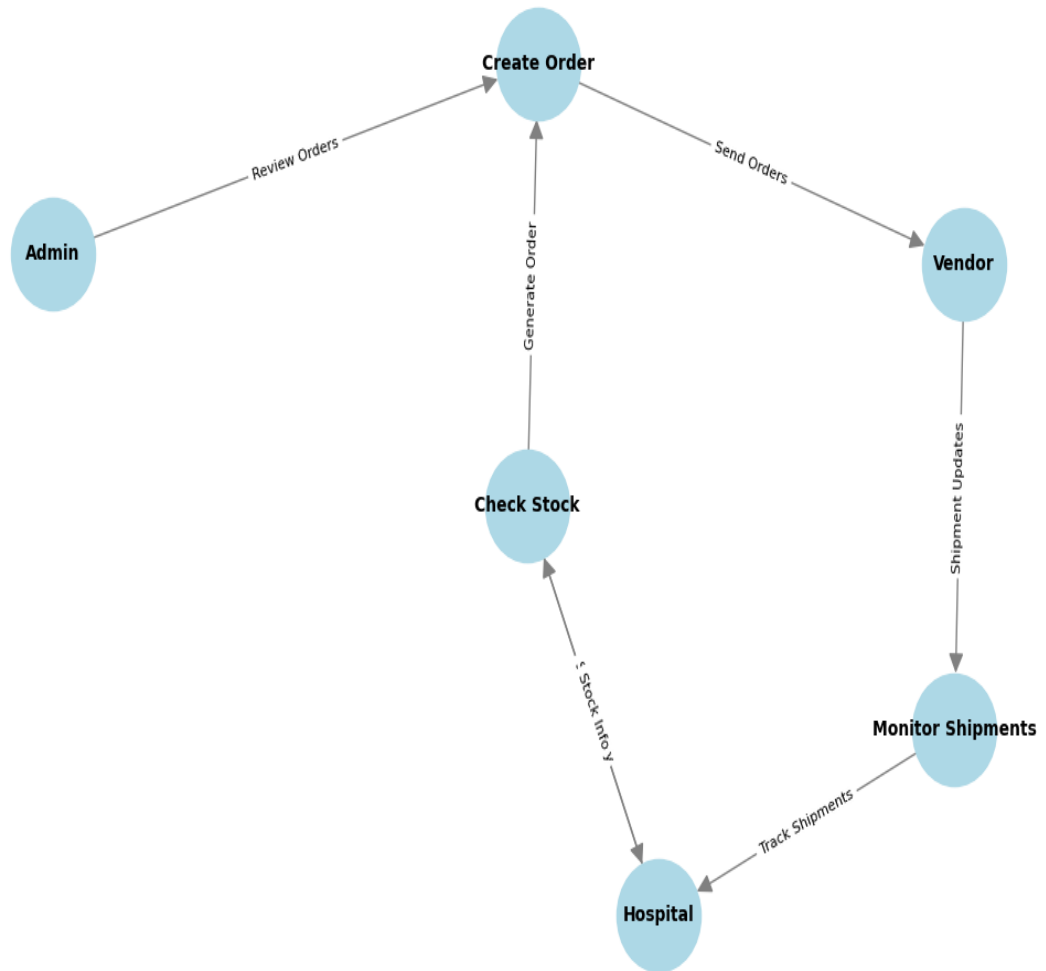
This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Travels and Tales is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays an important role in understanding 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 1<sup>st</sup> level DFD comes into play.



**Fig 6.1 0 Level DFD**



**Fig 6.2 1st Level DFD**

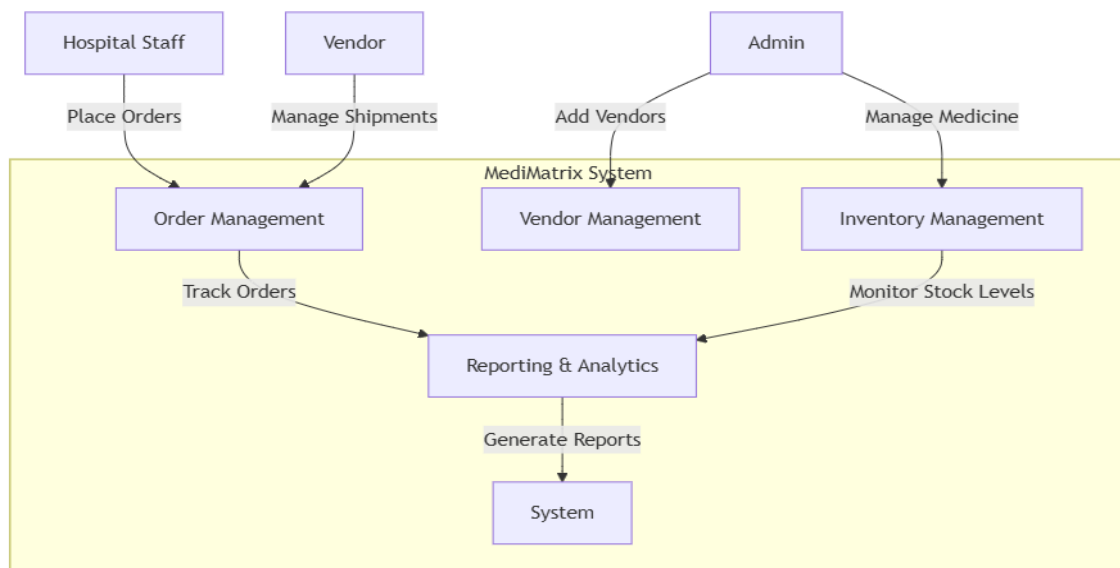


**Fig 6.3 2nd Level DFD**

## 6.4 Use Case Diagram

The MediMatrix supports hospital staff, vendors, and administrators in managing medicine-related operations. Hospital staff can place orders, while vendors handle shipments through the Order Management module.

Admins manage vendors via the Vendor Management module and oversee medicines using the Inventory Management module. These modules help in tracking orders and monitoring stock levels.



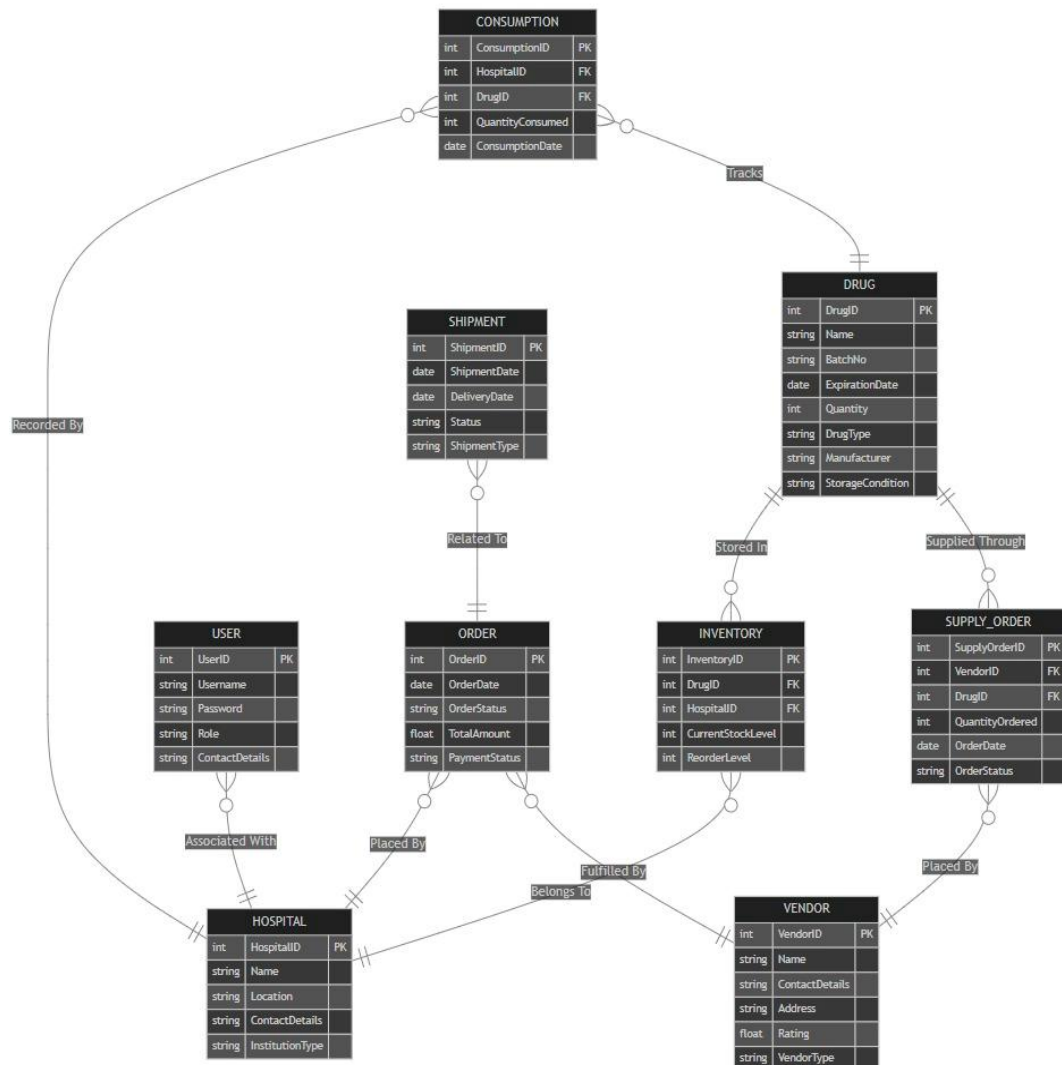
**Fig 6.4 Use Case**

## 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 entity relationship diagrams.



**Fig 7.1 ER-Diagram**

## 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

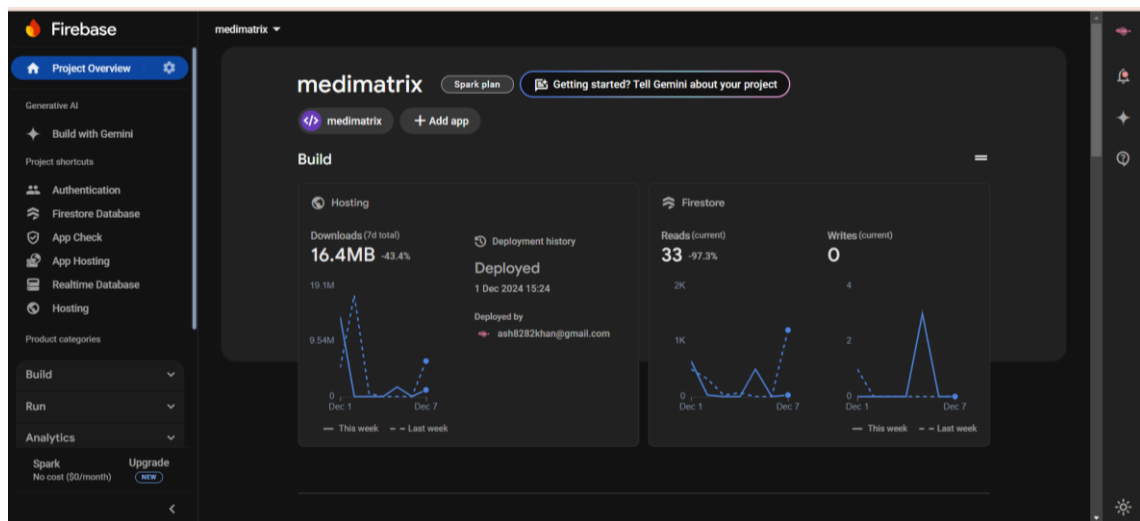


Fig 8.1 Admin

#### 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

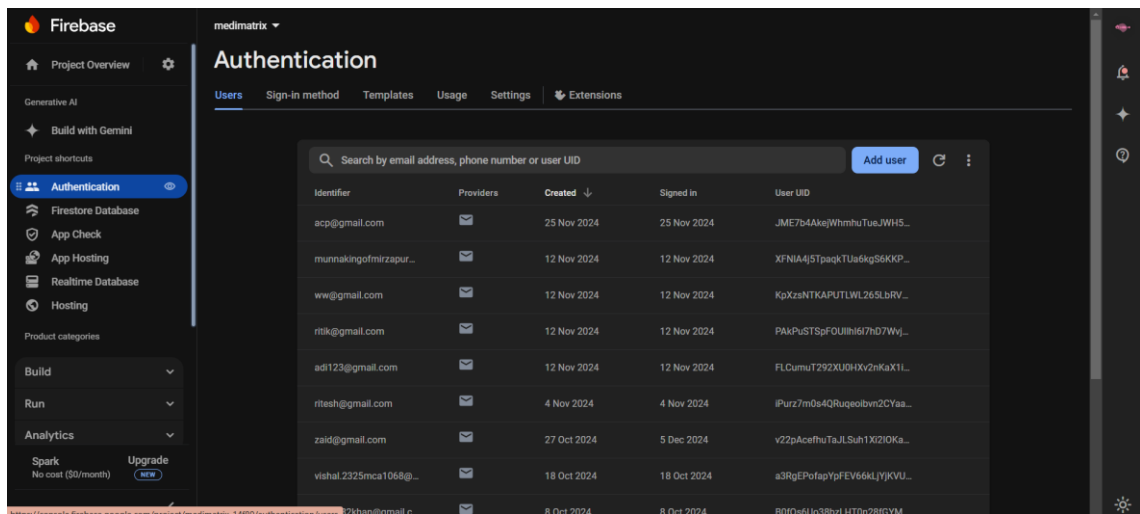


Fig 8.2 User

### 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 information.

### 8.3 Packages

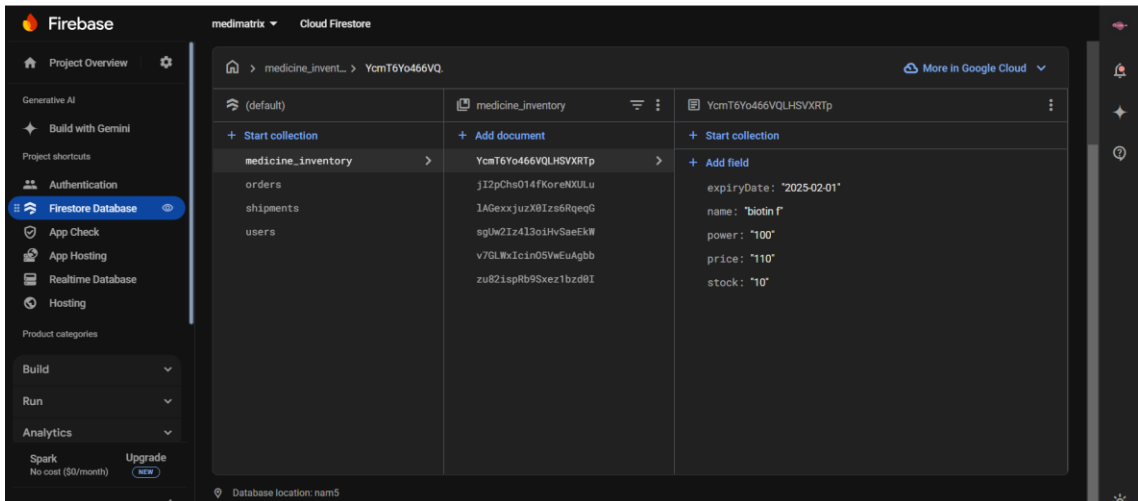


Fig 8.3 Packages

#### 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.

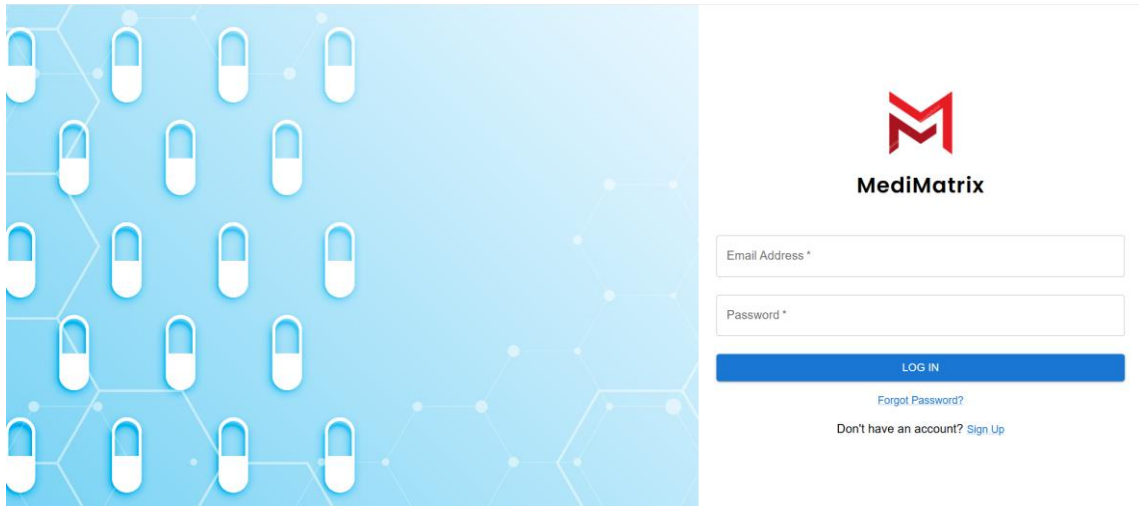
##### Desc (Destination):

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

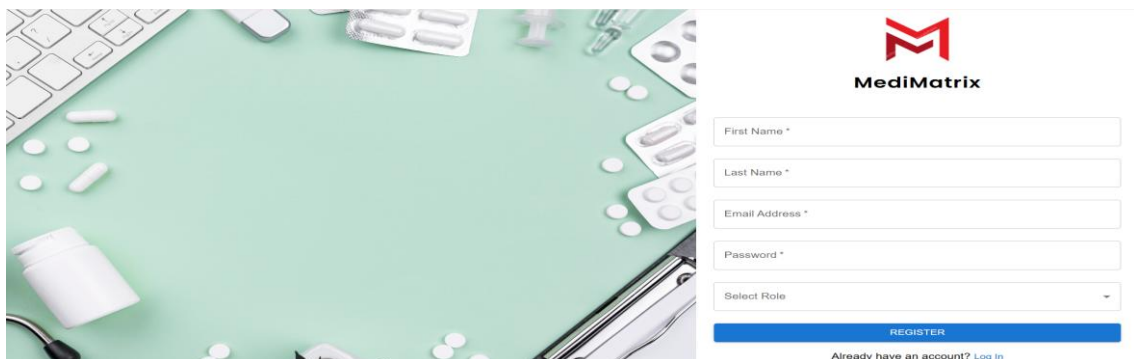
#### 9.1 Login

The image shows a login page for MediMatrix. On the left is a decorative blue background with a hexagonal pattern and white pill icons. On the right is a white form area. At the top is the MediMatrix logo, a red stylized 'M' above the text 'MediMatrix'. Below the logo are two input fields: 'Email Address \*' and 'Password \*'. A blue 'LOG IN' button is positioned below the password field. Under the button are two links: 'Forgot Password?' and 'Don't have an account? Sign Up'.

**Fig 9.1 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.

#### 9.2 SignUp

The image shows a sign-up page for MediMatrix. On the left is a decorative green background with a medical theme, including a keyboard, a mouse, and various pills. On the right is a white form area. At the top is the MediMatrix logo, a red stylized 'M' above the text 'MediMatrix'. Below the logo are five input fields: 'First Name \*', 'Last Name \*', 'Email Address \*', 'Password \*', and a 'Select Role' dropdown menu. A blue 'REGISTER' button is positioned below the dropdown. At the bottom is a link: 'Already have an account? Log In'.

**Fig 9.2 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

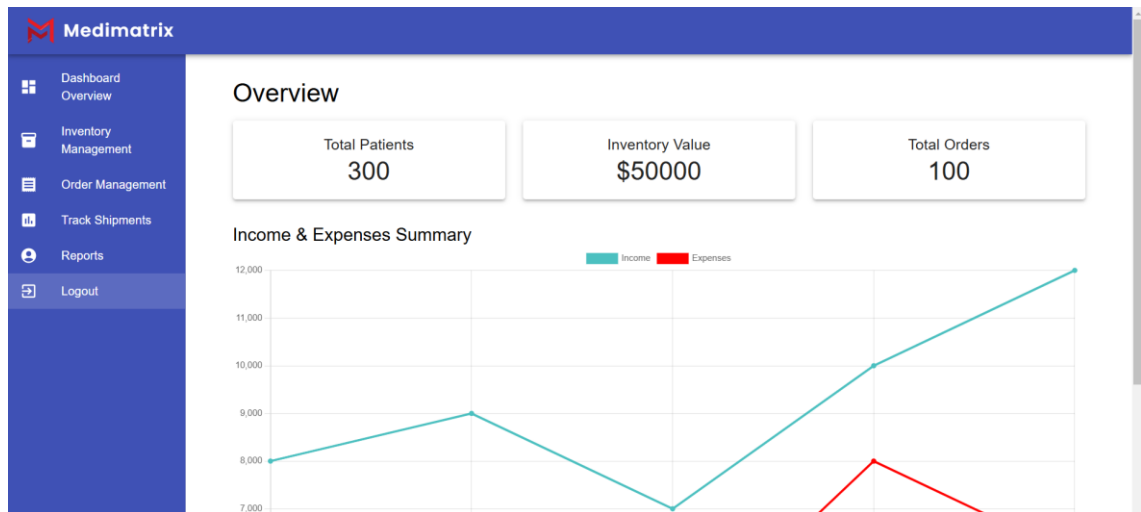
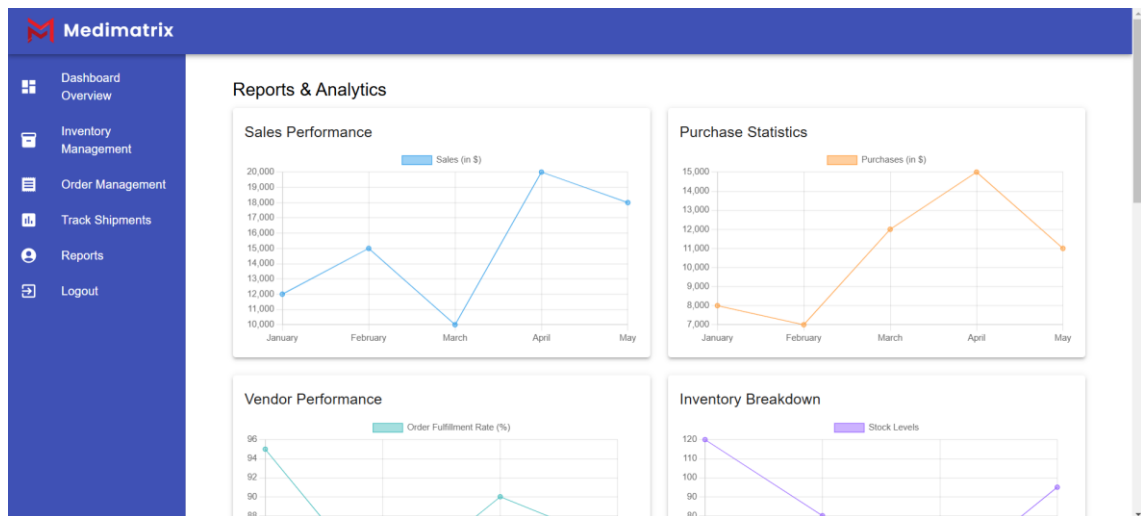


Fig 9.3 Dashboard page

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.

## 9.4 Reports Page



**Fig 9.4 Reports page**

This Medimatrix dashboard page provides a comprehensive overview of Reports & Analytics. It features visualized data through four charts: Sales Performance, Purchase Statistics, Vendor Performance, and Inventory Breakdown. Each graph presents trends over months, enabling users to monitor sales, purchases, order fulfillment rates, and stock levels for effective decision-making.

## 9.5 Inventory Management Page

The Medimatrix Inventory Management dashboard displays a table of medicine inventory. The left sidebar contains navigation links: Dashboard Overview, Inventory Management, Order Management, Track Shipments, Reports, and Logout.

**Medicine Inventory**

**Inventory List** [ADD NEW MEDICINE](#) [EXPORT TO EXCEL](#)

#	Medicine Name <sup>Power</sup>	Medicine Price	Stock	Expiry Date	Action
1	biotin 1 <sup>100</sup>	₹110	10	2025-02-01	<a href="#">Edit</a> <a href="#">Delete</a>
2	dolo 650	₹65	400	2024-02-12	<a href="#">Edit</a> <a href="#">Delete</a>
3	foracord 200	₹50	100	2026-01-20	<a href="#">Edit</a> <a href="#">Delete</a>
4	biotin 200	₹75	20	2025-02-02	<a href="#">Edit</a> <a href="#">Delete</a>
5	Ashwagandha 100	₹300	50	2025-01-16	<a href="#">Edit</a> <a href="#">Delete</a>

**Fig 9.5 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>
  }
/>
```



```

</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={
      23
      <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>
        </Box>
      );
    default:
      return null;
  }
};

```



```

<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>
</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" />

```

```

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>
      }
    />

```

```

</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>

```

```

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>

```

```

<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',
      },
    }}
  >

```

```

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>
      }
    />

```



```

<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" />

```

```

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>
      }
    />
  );
}

```

```

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>
      }
    />

```

```

<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',
      },
    }}
  >

```

```

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>

```

```

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>
      }
    />

```

```

<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" />

```

## **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.

## CHAPTER 12

### 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.

## CHAPTER 13

### CONCLUSION

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.

## CHAPTER 13

### REFERENCES

1. React. (n.d.). *React documentation*. React. <https://react.dev/learn>
2. W3Schools. (n.d.). *React tutorial*. W3Schools. <https://www.w3schools.com/react/>
3. Firebase. (n.d.). *Firebase documentation*. Google.  
<https://firebase.google.com/docs>
4. MUI. (n.d.). *Material UI documentation*. MUI. <https://mui.com/>
5. Hagberg, A., Schult, D., & Swart, P. (n.d.). *NetworkX documentation*. NetworkX.  
<https://networkx.github.io>
6. React. (n.d.). *Describing the UI*. React. <https://react.dev/learn/describing-the-ui>
7. W3Schools. (n.d.). *React components*. W3Schools.  
[https://www.w3schools.com/react/react\\_components.asp](https://www.w3schools.com/react/react_components.asp)
8. Firebase. (n.d.). *Get started with Firebase for web apps*. Google.  
<https://firebase.google.com/docs/web/setup>
9. MUI. (n.d.). *Getting started with Material UI*. MUI. <https://mui.com/material-ui/getting-started/overview/>
10. NetworkX. (n.d.). *Introduction to NetworkX*. NetworkX.  
<https://networkx.github.io/documentation/stable/tutorial.html>