1. ABSTRACT

This project focuses on the development of a **Pharmacy Information System (PIS)**, designed as a web application to simplify the process of ordering medicines. Recognizing the growing demand for convenient healthcare solutions, the system enables patients to log in securely, upload their prescriptions, and place orders for medications with ease.

By integrating features that support real-time tracking, the application enhances communication between patients and delivery personnel, ensuring that medication delivery is both reliable and timely. Delivery staff can efficiently manage incoming orders, improving operational workflows and reducing delays in medication access.

The system is built with a user-centric approach, prioritizing an intuitive interface that accommodates a variety of patient needs while safeguarding sensitive data through robust security measures. The analysis of the application reveals significant enhancements in order accuracy, patient engagement, and overall service efficiency. This report outlines the design process, the technologies employed, and the measurable impact of the web application, demonstrating its effectiveness in transforming the medication ordering experience in real-world pharmacy operations.

By automating key processes, this project aims to improve operational efficiency, reduce errors, and enhance the overall patient experience. The implementation of such a system not only improves pharmacy workflow but also contributes to better healthcare outcomes by ensuring timely and reliable access to medications.

2. PROBLEM STATEMENT

Traditional pharmacy systems often rely on manual processes for managing prescriptions and handling orders, which can lead to inefficiencies, long wait times, and errors in medication fulfilment. Patients are required to visit pharmacies in person to place orders, making the process inconvenient, especially for those with mobility or health challenges.

Additionally, basic inventory management systems can result in medication shortages and delays. The absence of real-time order tracking and limited communication between patients and pharmacy staff further contribute to an unsatisfactory patient experience.

1. **Manual Prescription Handling**: Leads to errors and delays, especially during high demand periods.
2. **In-Person Ordering**: Results in long wait times and inconvenience for patients with mobility or health issues.
3. **Basic Inventory Management**: Causes medication stock shortages and delays in order fulfilment.
4. **Lack of Real-Time Tracking**: Patients have no visibility into their order status, leading to uncertainty.
5. **Limited Communication**: Inefficient communication channels between patients and pharmacy staff cause delays in addressing queries.

3. INTRODUCTION

Pharmacies are essential in providing timely access to medications, yet many still rely on traditional manual processes, leading to operational inefficiencies and a poor patient experience. Manual prescription handling, in-person ordering, and limited inventory management often result in long wait times, medication errors, and stock shortages. Additionally, the lack of real-time tracking leaves patients uncertain about the status of their medication orders, while communication gaps between patients and pharmacy staff can further slowdown the process.

To address these challenges, this project introduces a **Pharmacy Information System (PIS)** designed to streamline and optimize pharmacy operations. The PIS allows patients to securely log in, upload prescriptions, and place medication orders online, reducing the need for in-person visits. It also integrates real-time order tracking, ensuring that patients can monitor their orders and receive updates on the status of their medications. Improved communication tools within the system enable quicker and more efficient interactions between pharmacy staff and patients.

This report provides an in-depth look at the design, implementation, and anticipated impact of the Pharmacy Information System. By automating key processes and enhancing communication, the system aims to improve operational efficiency, reduce errors, and provide a more seamless experience for patients, ultimately contributing to better healthcare outcomes.

The implementation of the **Pharmacy Information System (PIS)** is not only a step toward improving operational efficiency but also a crucial advancement in patient-centered care.

4. EXISTING METHODS

1. **Manual Prescription Processing**: Pharmacies often rely on manual input for processing prescriptions. Patients visit in person, and the staff manually enters prescription details into their system
2. **In-Person Medication Ordering**: Patients typically need to be physically present at the pharmacy to place their orders, leading to long wait times, inconvenience for those with mobility issues, and a lack of flexibility for chronic patients who need regular refills.
3. **Basic Inventory Management**: Inventory is usually tracked manually or through simple digital systems, which often do not provide real-time data. This can lead to stock shortages or overstocking
4. **No Real-Time Order Tracking**: Patients are left in the dark about when their medication will be ready, resulting in frequent phone calls or visits to check on their orders.
5. **Limited Patient-Pharmacy Communication**: Communication between patients and pharmacies is mostly limited to in-person or phone conversations, making it difficult to resolve issues, ask questions, or receive updates efficiently.

4.1 DISADVANTAGES

1. Higher error rates in prescription processing.
2. Inconvenience for patients needing in-person visits.
3. Long wait times for medication orders.
4. Lack of real-time order tracking.
5. Poor communication between patients and pharmacy staff.

5. PROPOSED WORK

The proposed work involves the development and implementation of a **Pharmacy Information System (PIS)** designed to enhance the efficiency and effectiveness of pharmacy operations. The key features of the system include:

1. **Online Prescription Upload**: Patients can securely log in and upload their prescriptions through the web application, eliminating the need for in-person visits.
2. **Medication Ordering**: The system allows patients to place medication orders online, reducing wait times and improving accessibility.
3. **Real-Time Order Tracking**: Patients can track the status of their orders in real-time, providing transparency and reducing uncertainty about when medications will be ready.
4. **Automated Inventory Management**: The system will integrate an automated inventory management feature to ensure accurate stock levels and timely restocking of medications.
5. **Enhanced Communication Tools**: The PIS will include communication features that facilitate efficient interaction between patients and pharmacy staff

5.1 ADVANTAGES

1. Intuitive user interface for enhanced usability.
2. Integration with electronic health record systems.
3. Data analytics for informed decision-making insights.
4. Support for telehealth consultations with pharmacists.
5. Automated reminders for prescription refills and renewals.

6. SYSTEM REQUIREMENTS

The proposed Pharmacy Information System (PIS) requires a robust hardware setup, including a server with a dual-core processor and 8GB RAM. Key functionalities will support user registration, prescription uploads, order management, and inventory tracking for a seamless experience. Non-functional requirements like security, performance, scalability, and usability will also be prioritized. This comprehensive approach ensures an effective and efficient system for both patients and pharmacy staff.

6.1 HARDWARE REQUIREMENTS

- System: Dual core
- Hard Disk: 500 GB
- RAM: 8 GB
- Monitor: 15" VGA Colour
- Mouse: Logitech
- Network: High-speed router (10 Mbps minimum)

6.2 SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11, Linux (Ubuntu), or macOS
- Database: MySQL, PostgreSQL, or MongoDB
- Development Environment: NetBeans
- Database Management: SQLyog
- Programming Languages: HTML, CSS, JavaScript, and a server-side language
- Frameworks: React, Angular, or Vue.js for front end; Node.js or Django for
- Software: Antivirus and firewall protection

7. SYSTEM ARCHITECTURE

7.1 ARCHITECTURE OVERVIEW

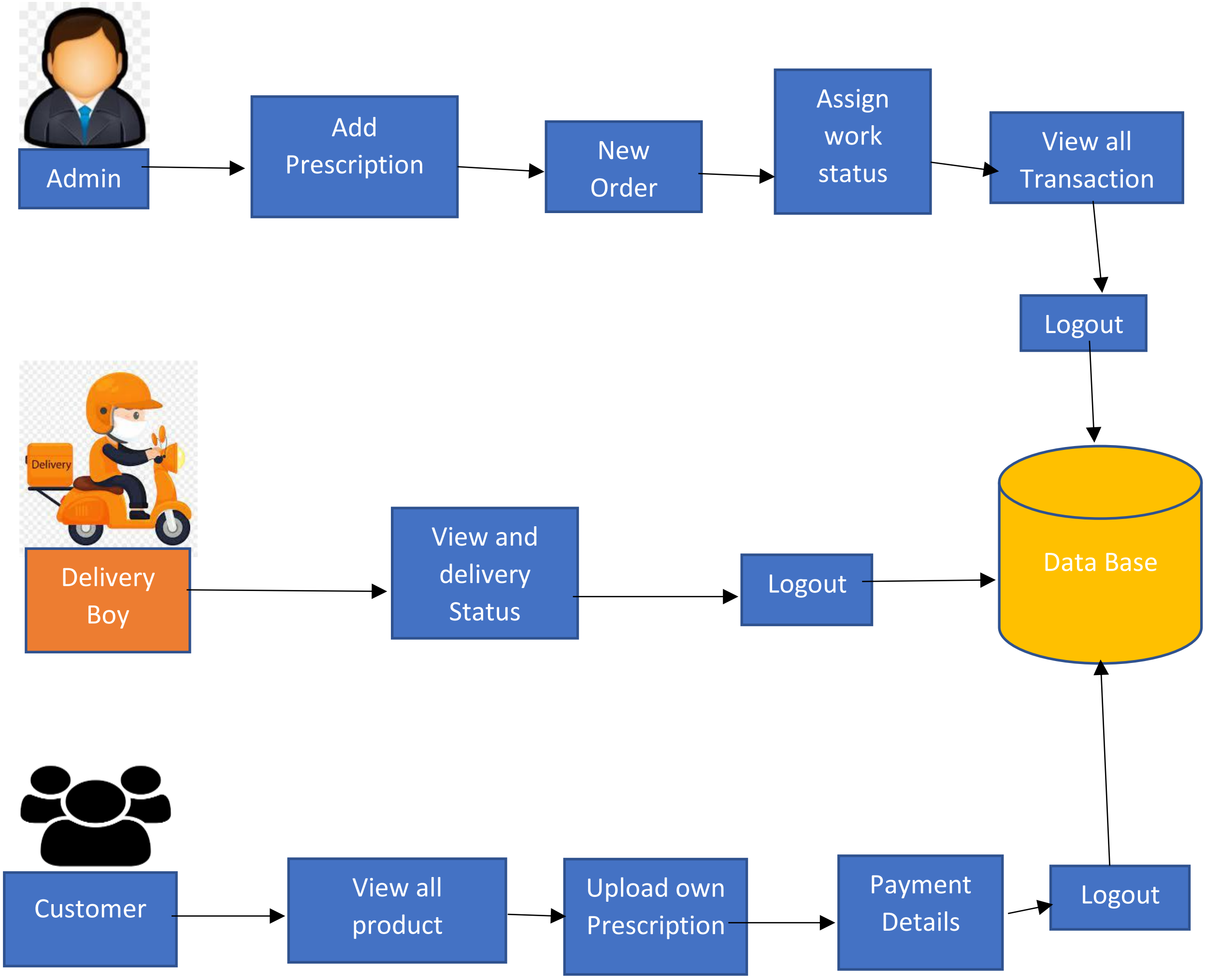


Fig 1. Architecture diagram

The architecture of the Pharmacy Information System (PIS) consists of several layers that ensure efficient operation and user interaction. This is the front end where users (patients and pharmacy staff) interact with the system through web pages designed with HTML, CSS, and JavaScript. This layer contains the business logic, processing user requests and managing functionalities such as order management and inventory checks, typically built with languages like Python or PHP. Responsible for data storage.

7.2 MODULE DESCRPTION

In this project has three Modules:

1.Admin

2.Customer

3.Delivery Boy

Admin:

1. Admin

- **Login**:

  Admin can log in using the correct username and password.

- **Add Prescription**:

  Add product details along with their costs.

- **View and Update**:

  View all updated products categorized; options to delete or edit products.

- **Add Delivery Boy**:

  Add new delivery boys with their basic information.

- **New Order**:

  View all new orders, update product costs, and assign work to delivery boys.

- **Assign Work Status**:

  Monitor and update the status of delivery tasks for delivery boys.

- **View All Transactions**:

  View all delivered products and related transactions.

- **Logout**: Admin can log out of the system.

2. Customer

- **Registration**:

  Customers can register an account with basic information.

- **Login**:

  Customers can log in with the correct username and password.

- **View Products**:

  View all available products and place orders.

- **Upload Prescription**:

  Upload prescriptions in image format for ordered products.

- **Payment Details**:

  View all orders and cost updates.

- **Logout**:

  Customers can log out of their accounts.

3. Delivery Boy

- **Login**:

  Delivery boys can log in with the correct username and password.

- **View Delivery Status**:

  View current delivery assignments and their statuses.

- **Logout**:

  Delivery boys can log out of the system.

4. Cloud Server Model

The Cloud Server Module facilitates the overall functionality and data management of the Pharmacy Information System.

- **Data Storage**:

  Securely stores user information, product details, prescriptions, and transaction data in the cloud.

- **User Authentication**:

  Manages login sessions for admin, customers, and delivery boys, ensuring secure access to the system.

- **Real-Time Updates**:

  Provides real-time data synchronization across all user interfaces, ensuring that all users have access to the latest information.

- **Scalability**:

  Easily scales resources based on user demand and system load, ensuring optimal performance.

- **API Integration**:

  Supports integration with external services for payment processing and other functionalities, enhancing system capabilities.

- **Performance Monitoring**:

  Monitors server performance and user activity to optimize resource allocation and improve response times.

8. SYSTEM DESIGN

The **Pharmacy Information System** is designed with a modular architecture to enhance functionality, maintainability, and scalability. The system incorporates a user-friendly web application that is accessible on various devices, including desktops, tablets, and smartphones, providing tailored dashboards for different user roles such as admins, customers, and delivery boys. The application layer employs robust server-side frameworks like Node.js or Django to efficiently handle business logic and manage user requests. Communication between the front end and back end is facilitated through RESTful APIs, ensuring seamless data exchange. Data is structured and managed using a relational database system like MySQL or PostgreSQL, allowing for efficient storage and retrieval of user accounts, product information, orders, and transactions. The system is hosted on a cloud infrastructure, providing scalability and high availability,

- **User Interface**: Responsive web application with tailored dashboards for different roles.
- **Application Layer**: Uses server-side frameworks (e.g., Node.js, Django) and RESTful APIs for efficient communication.
- **Database Layer**: Structured data management with MySQL or PostgreSQL for easy storage and retrieval.
- **Cloud Infrastructure**: Hosted on cloud servers, ensuring scalability and high availability.
- **Security**: Incorporates authentication, authorization, and data encryption for sensitive data protection.
- **Performance Monitoring**: Utilizes monitoring tools to track application health and user activity for proactive maintenance.

8.1 USE CASE DIAGRAM

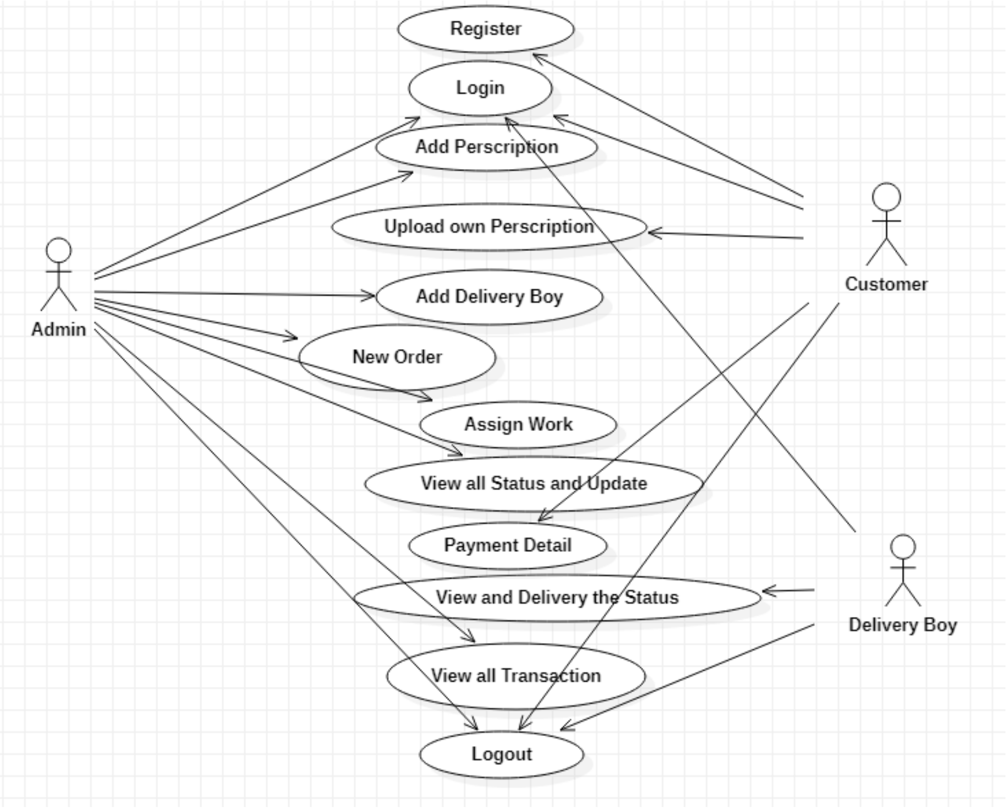


Fig 2. Use Case diagram

The diagram illustrates a use case for a pharmacy information system with three main roles: Admin, Customer, and Delivery Boy. The **Admin** has the most control, managing tasks such as registering, logging in, adding prescriptions, and uploading or adding delivery personnel. The admin also oversees new orders, assigns work to delivery boys, and updates statuses. The **Customer** can register, log in, upload their own prescription, view products, make payments, and view their order details. Finally, the **Delivery Boy** logs in to view the assigned orders and update the delivery status. The system facilitates seamless coordination between the admin, customer, and delivery personnel, streamlining the process of medicine ordering and delivery.

8.2 CLASS DIAGRAM

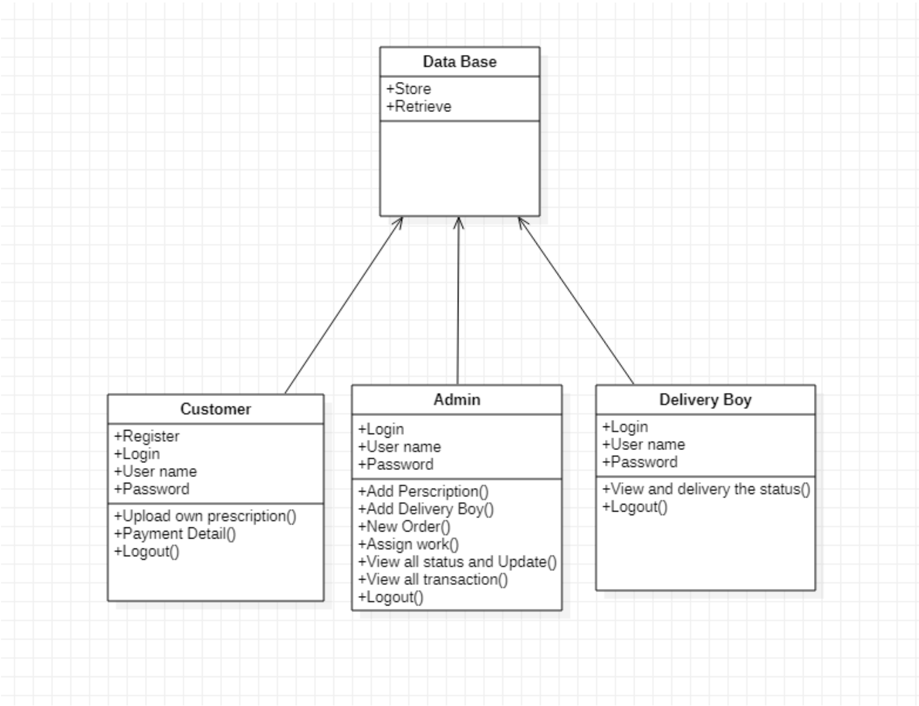


Fig 3. Class diagram

The diagram depicts the architecture of a Pharmacy Information System database, highlighting its central role in data management for three primary user roles: Customer, Admin, and Delivery Boy. At the core of the system is the Database, which provides essential functions for storing and retrieving information. Each user role interacts with the database through specific methods. The Customer can register, log in, upload prescriptions, view payment details, and log out, all of which involve data interactions with the database. The Admin has additional functionalities, such as managing prescriptions, adding delivery personnel, overseeing new orders, assigning work, and viewing transaction histories. The Delivery Boy is limited to logging in and checking their assigned delivery status. This structured approach ensures efficient data handling, user management, and overall system performance, allowing for a seamless experience in managing pharmacy orders and deliveries.

8.3 SEQUENCE DIAGRAM

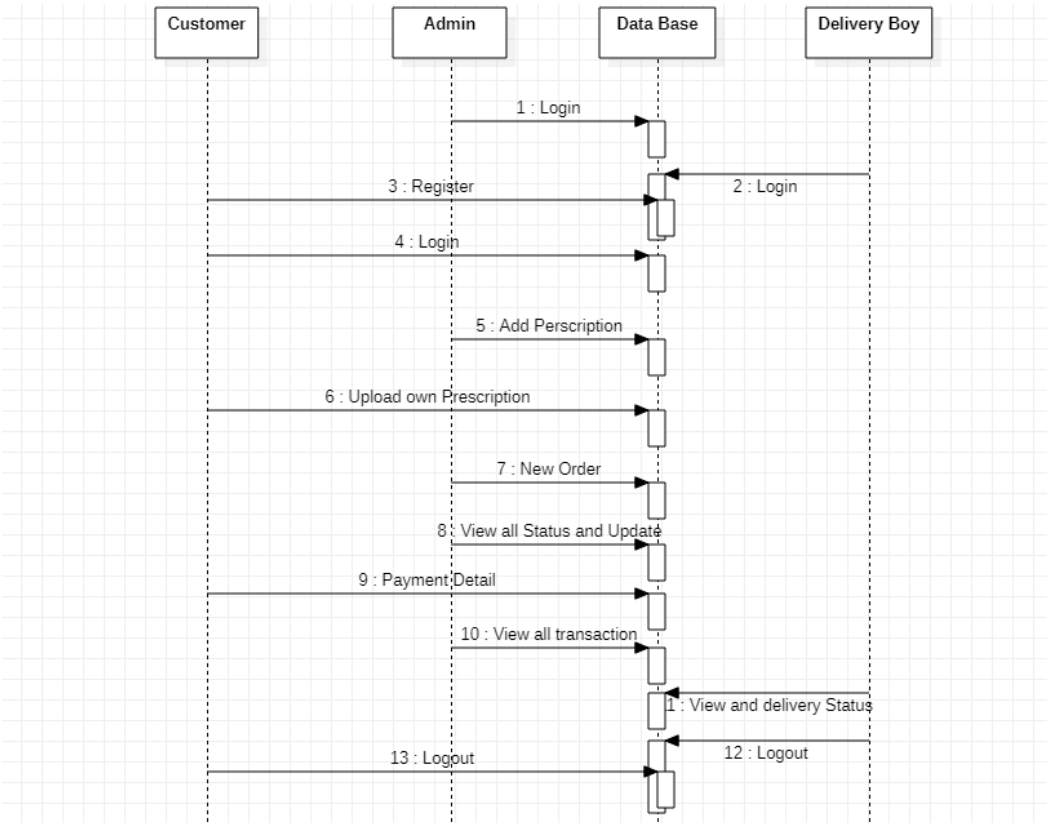


Fig 4. Sequence diagram

The sequence diagram illustrates the interactions within the Pharmacy Information System among the Customer, Admin, Delivery Boy, and the Database. It details the flow of actions from logging in to logging out for each user role. The customer starts by registering and logging into the system. They can upload their prescription, view payment details, and log out after completing their transactions. The admin logs in to manage prescriptions, view new orders, and update statuses. The delivery boy logs in to check the delivery status of their assigned orders The database serves as the central component, processing requests from all user roles, such as storing new prescriptions, retrieving order statuses, and managing user registrations.

8.4 COMMUNICATION DIAGRAM

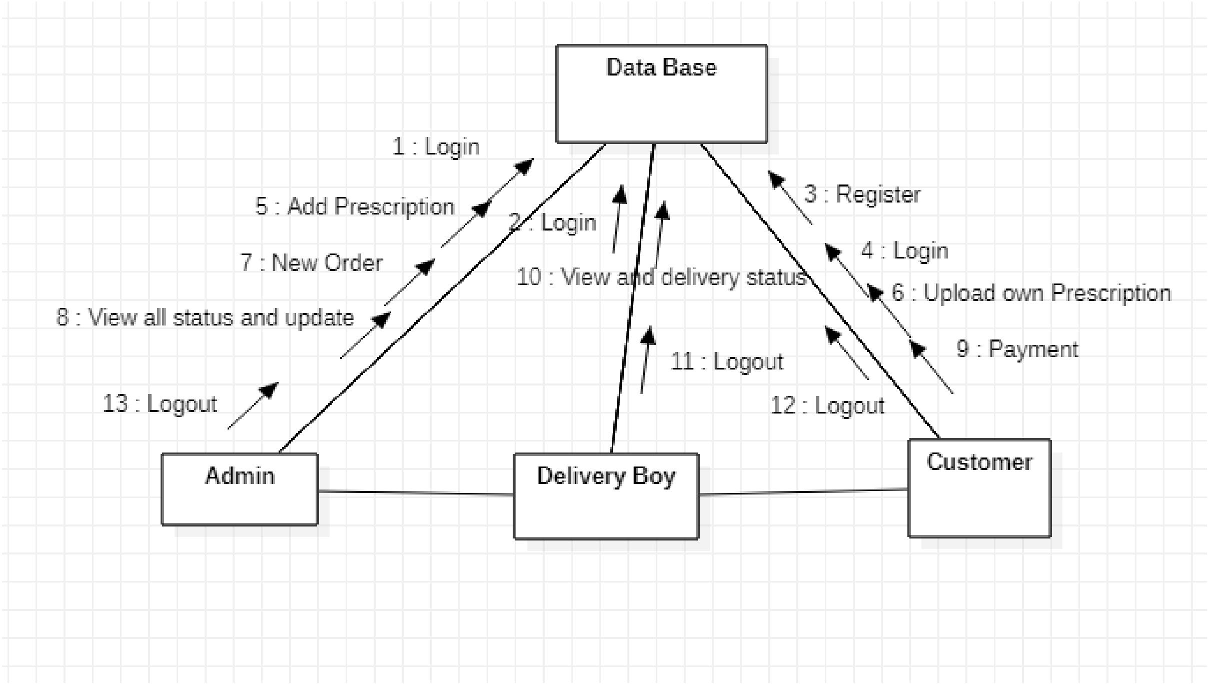


Fig 5. Communication diagram

The communication diagram illustrates the interactions between the Customer, Admin, Delivery Boy, and the Database in the Pharmacy Information System. Each user role initiates specific actions that require responses from the database, creating a seamless flow of information. The customer communicates with the system to register, log in, upload prescriptions, and view payment details. In parallel, the admin interacts with the database to add prescriptions, manage orders, and oversee transactions, ensuring all operations are up to date. Meanwhile, the delivery boy retrieves the delivery status for assigned orders. This diagram emphasizes the collaborative nature of the system, highlighting how each user role relies on the database for efficient processing and management of pharmacy operations.

8.5 DEPLOYMENT DIAGRAM

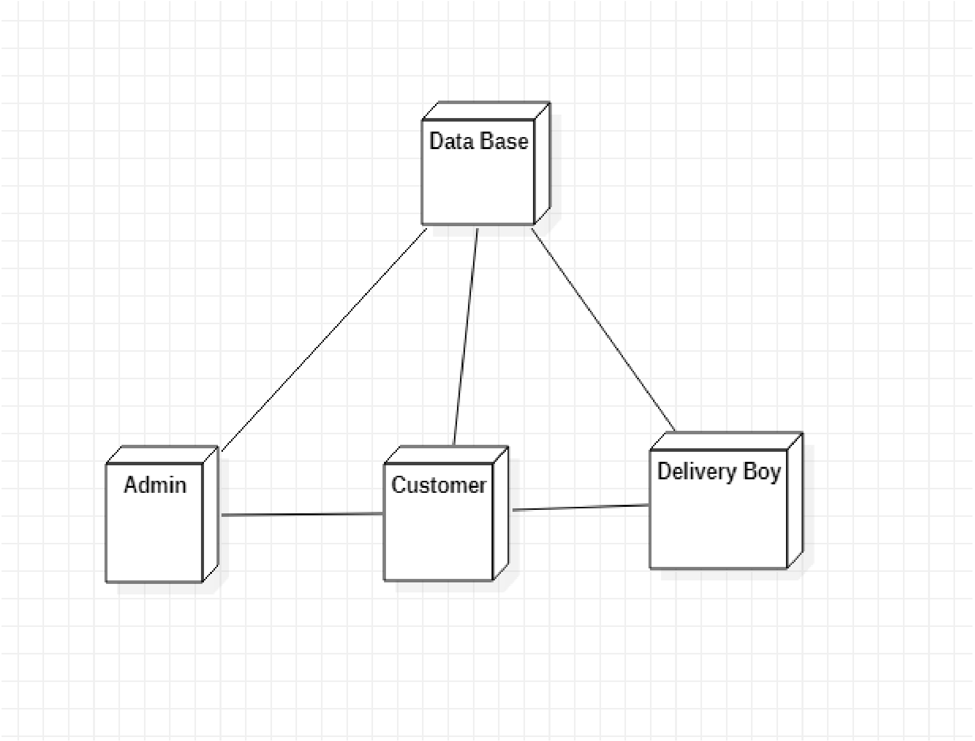


Fig 6. Deployment diagram

The deployment of the Pharmacy Information System involves setting up the application on a cloud server to ensure accessibility and scalability. The system is designed to provide users—customers, admin, and delivery personnel—with a seamless interface for ordering medicines, managing prescriptions, and tracking deliveries. After thorough testing in a controlled environment, the application will be deployed on the server, enabling real-time data access and storage in the database. This deployment will facilitate efficient user management, with secure login functionalities for different user roles. Additionally, regular updates and maintenance will be scheduled to enhance system performance and address any issues promptly, ensuring a reliable platform for all stakeholders involved in the pharmacy operations.

8.6 DATA FLOW DIAGRAM

LEVEL 0

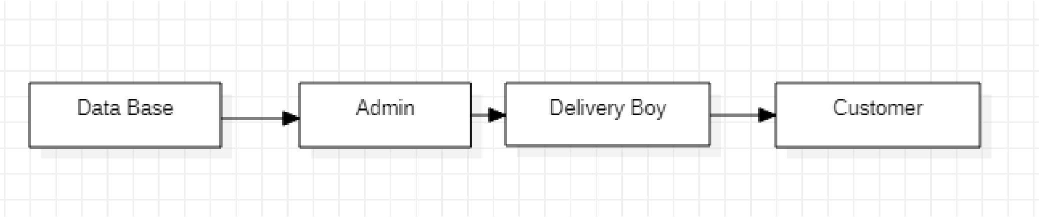


Fig 7. DFD Level 0

LEVEL 1

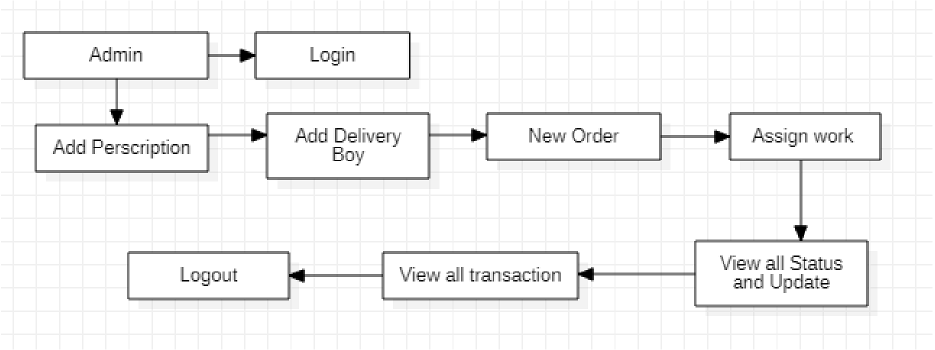


Fig 8. DFD Level 1

LEVEL 2

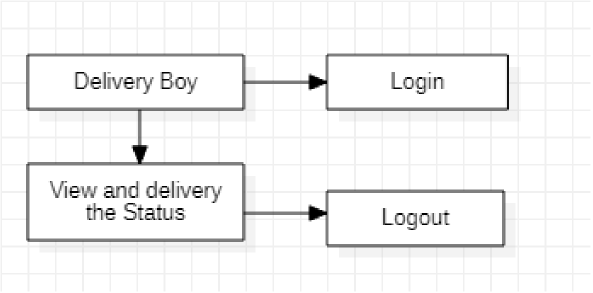


Fig 9. DFD Level 2

LEVEL 3

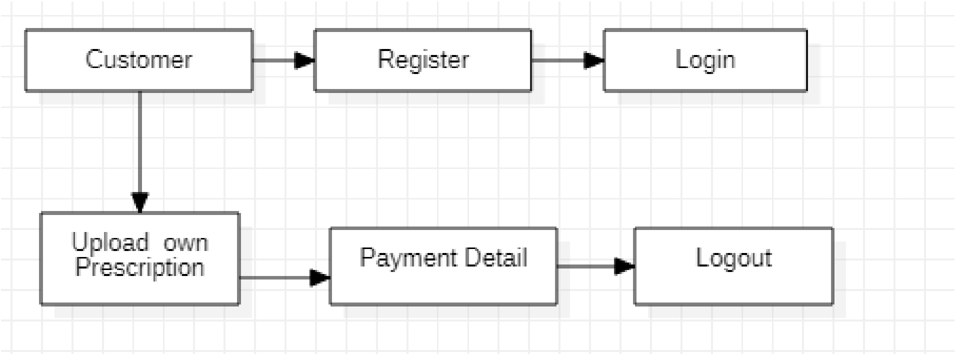


Fig 10. DFD Level 3

The Data Flow Diagrams (DFD) for the Pharmacy Information System illustrate the flow of information between various components of the system, delineating how data is processed and stored. DFD Level 1 provides a high-level overview, showcasing the primary entities such as the Admin, Customer, and Delivery Boy, and their interactions with the system's core processes, including registration, order management, and prescription uploads. Moving to DFD Level 2, the focus shifts to a more detailed representation of the Admin's functionalities, highlighting processes like adding prescriptions, managing orders, and updating transaction statuses, emphasizing the internal data movements within the database. Finally, DFD Level 3 delves deeper into the Customer and Delivery Boy interactions, detailing processes like viewing product listings, placing orders, and tracking delivery status, thus providing a comprehensive view of how data flows through the system and ensuring clarity in the interactions between users and system functionalities. Together, these DFDs serve as a critical tool for understanding the overall architecture and operational dynamics of the Pharmacy Information System.

8.7 ENTITY RELATIONSHIP DIAGRAM

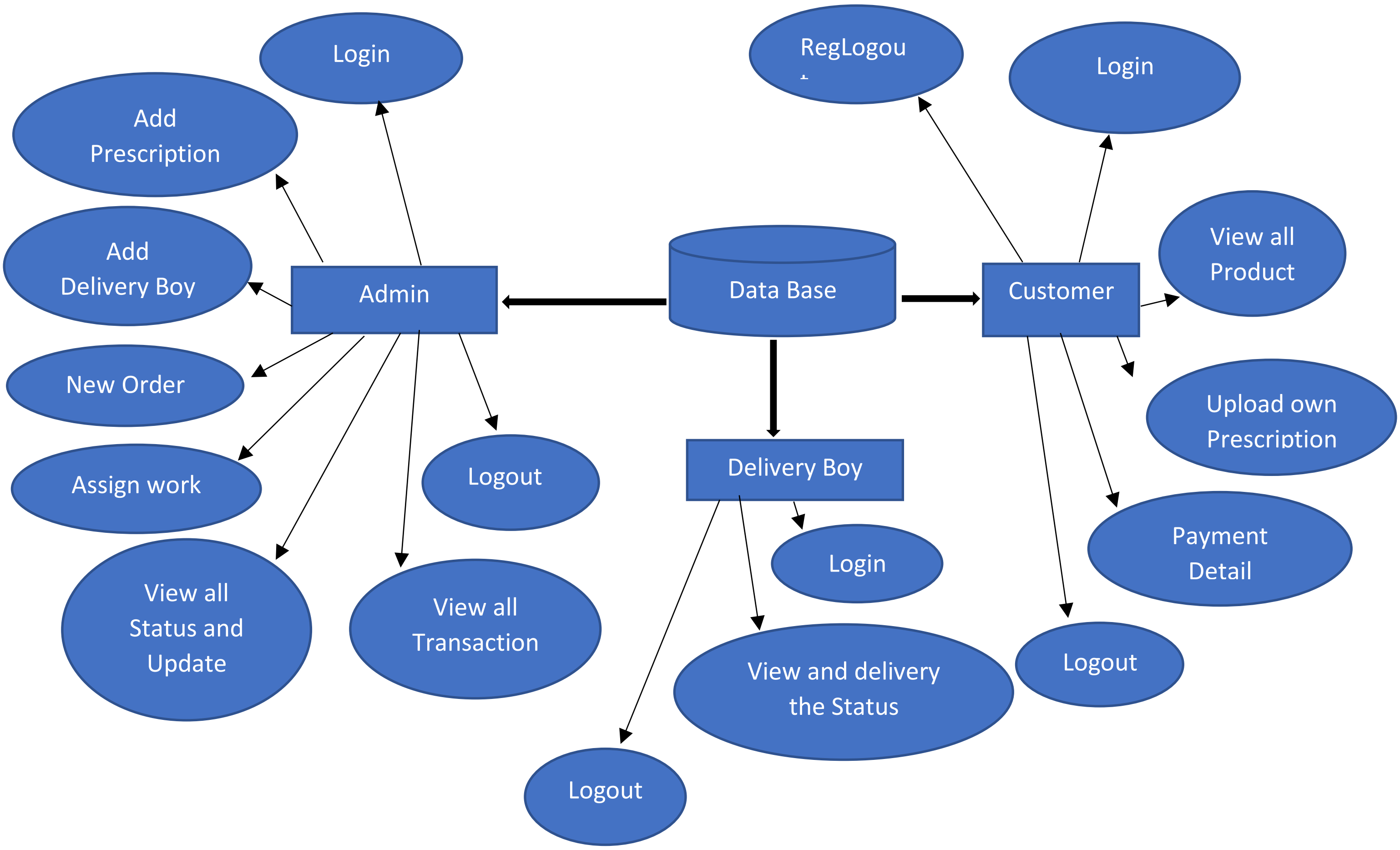


Fig 11. ER diagram

The Entity-Relationship Diagram (ERD) for the Pharmacy Information System visually represents the key entities and their relationships within the system. Central to the diagram are three primary entities: Admin, Customer, and Delivery Boy, each with distinct attributes. The Admin entity is responsible for managing product details, processing orders, and overseeing delivery assignments, while the Customer entity contains user information and tracks prescriptions and order histories. The Delivery Boy entity focuses on managing delivery tasks and updating order statuses.

9. SYSTEM IMPLEMENTATION

9.1 CLIENT-SIDE CODING (FRONT END)

```
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Connection"%>
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="TemplateMo">
<link href="https://fonts.googleapis.com/css?family=Poppins:100,200,300,400,500,600,700,800,900&display=swap" rel="stylesheet">

<title>Design  and Performance Implications of Pharmacy Information System</title>
<!--

Lava Landing Page

https://templatemo.com/tm-540-lava-landing-page
<!-- Additional CSS Files -->
<link rel="stylesheet" type="text/css" href="assets/css/bootstrap.min.css">
```

```
<link rel="stylesheet" type="text/css" href="assets/css/font-awesome.css">

<link rel="stylesheet" href="assets/css/templatemo-lava.css">

<link rel="stylesheet" href="assets/css/owl-carousel.css">

</head>

<body>

<!-- ***** Preloader Start ***** -->
<div id="preloader">
<div class="jumper">
<div></div>
<div></div>
<div></div>
</div>
</div>
<!-- ***** Preloader End ***** -->

<!-- ***** Header Area Start ***** -->
<header class="header-area header-sticky">
<div class="container">
<div class="row">
<div class="col-12">
                <nav class="main-nav">
```

```
    <!-- ***** Logo Start ***** -->
    <a href="index.html" class="logo">
        Pharmacy
    </a>
    <!-- ***** Logo End ***** -->
    <!-- ***** Menu Start ***** -->
    <ul class="nav" style="margin-left: 650px;">
        <li class="scroll-to-section"><a href=ahome.jsp>Home</a></li>
        <li class="submenu">
            <a href="javascript:;">Drop Down</a>
            <ul>
                <li><a href="add.jsp" >Add Prescription</a></li>
                <li><a href="viewupdate.jsp" >View & Updates</a></li>
                <li><a href="adddelivery.jsp" >Add Delivery Boy</a></li>
                <li><a href="neworder.jsp" >New Order</a></li>
                <li><a href="assign.jsp" >Assign work status</a></li>
                <li><a href="viewsta.jsp" >View all Status</a></li>
                <li><a href="viewtra.jsp" >View all Transaction</a></li>
                <li><a href="index.html" >Logout</a></li>
            </ul>
        </li>
    </ul>
    <a class='menu-trigger'>
        <span>Menu</span>
    </a>
    <!-- ***** Menu End ***** -->
</nav>
```

```
            </div>
          </div>
        </div>
      </header>
      <!-- ***** Header Area End ***** -->
    <style>
            tr,td{
                        font-family: cursive;
                        font-size: 20px;
                        color: black;
                        font-weight: bold;
                        padding: 5px;
            }
            table,tr,th,td{

                        height: auto;
                  border-collapse: collapse;
                  border:2px solid #d39e00;
                        border-radius: 20px;
                  padding:5px;
                  text-align:center;
                        background: white;
                        color: crimson;
            }
            th{
                        color: #F0542C;
            }
```

```
                table{
                        margin-left:220px;
                             height: auto;
                  }
   </style>


   <!-- ***** Welcome Area Start ***** -->
      <!-- ***** Header Text Start ***** -->
      <div class="header-text">
         <h2 align="center" style=" color:orange; margin-top: 15%; font-family: cursive;font-weight: bold; font-size: 30px;">Assign Delivery Boy Work Details !</h2><br><br>
   <div class="container">
      <form action="assign1.jsp"><table align="center" style="width:700px; ">
                        <tr><th>Prescription</th>
                        <th>P-ID</th>
                        <th>P-Name</th>
                        <th>Total</th>
                        <th style="width: 200px;">Action</th>
        </tr>
  <%
                  Class.forName("com.mysql.jdbc.Driver");
                  Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/pharmacy","root","root");
                  PreparedStatement query=con.prepareStatement("select * from upload where status='Assign'");
                  ResultSet rs=query.executeQuery();
                  System.out.println(query);
                  while(rs.next())
```

```
                    {
                      String pid=rs.getString("uid");
                      String pname=rs.getString("uname");
                      String type=rs.getString("upload");
                      int cost=rs.getInt("cost");
                      String status=rs.getString("status");
                    %>
        <!-- Page Content  -->


                      <tr><td><img src="upload/<%=type%>" alt="" style="width: 90px;
height: 90px; border-radius: 50px;"></td>
                      <td><%=pid%></td>
                      <td><%=pname%></td>
                      <td><%=cost%></td>
                      <td style="color: red"><%=status%></td>
                      </tr>
                      <%}%></table> </form>
                <br><br><br><br></div></div></div>


          </div>
        </div>
        <!-- ***** Header Text End ***** -->
      </div>
      <!-- ***** Welcome Area End ***** -->


      <!-- ***** Features Big Item Start ***** -->
      <section class="section" id="about">
        <div class="container">
          <div class="row">
            <div class="col-lg-4 col-md-6 col-sm-12 col-xs-12"
                data-scroll-reveal="enter left move 30px over 0.6s after 0.4s">
                <div class="features-item">
                    <div class="features-icon">
                      <h2>01</h2>
                      <img src="assets/images/features-icon-1.png" alt="">
                      <h4>Trend Analysis</h4>
                      <p>Curabitur pulvinar vel odio sed sagittis. Nam maximus ex diam,
nec consectetur diam.</p>
                      <a href="#testimonials" class="main-button">
                        Read More
                      </a>
                    </div>
                </div>
            </div>
            <div class="col-lg-4 col-md-6 col-sm-12 col-xs-12"
                data-scroll-reveal="enter bottom move 30px over 0.6s after 0.4s">
                <div class="features-item">
                    <div class="features-icon">
                      <h2>02</h2>
                      <img src="assets/images/features-icon-2.png" alt="">
                      <h4>Site Optimization</h4>
                      <p>Curabitur pulvinar vel odio sed sagittis. Nam maximus ex diam,
nec consectetur diam.</p>
                      <a href="#testimonials" class="main-button">
                        Discover More
                      </a>
```

```
                    </div>
                </div>
            </div>
            <div class="col-lg-4 col-md-6 col-sm-12 col-xs-12"
                data-scroll-reveal="enter right move 30px over 0.6s after 0.4s">
                <div class="features-item">
                    <div class="features-icon">
                        <h2>03</h2>
                        <img src="assets/images/features-icon-3.png" alt="">
                        <h4>Email Design</h4>
                        <p>Curabitur pulvinar vel odio sed sagittis. Nam maximus ex diam,
nec consectetur diam.</p>
                        <a href="#testimonials" class="main-button">
                            More Detail
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<!-- ***** Features Big Item End ***** -->

<div class="left-image-decor"></div>
 <!-- ***** Features Big Item Start ***** -->

<!-- ***** Features Big Item End ***** -->
```

```
    <div class="right-image-decor"></div>

    <!-- ***** Testimonials Starts ***** -->

    <!-- ***** Testimonials Ends ***** -->

    <!-- ***** Footer Start ***** -->

 <!-- jQuery -->
    <script src="assets/js/jquery-2.1.0.min.js"></script>


    <!-- Bootstrap -->
    <script src="assets/js/popper.js"></script>
    <script src="assets/js/bootstrap.min.js"></script>
<!-- Plugins -->
    <script src="assets/js/owl-carousel.js"></script>
    <script src="assets/js/scrollreveal.min.js"></script>
    <script src="assets/js/waypoints.min.js"></script>
    <script src="assets/js/jquery.counterup.min.js"></script>
    <script src="assets/js/imgfix.min.js"></script>
 <!-- Global Init -->
    <script src="assets/js/custom.js"></script>


</body>
</html>
```

9.2 SERVER-SIDE CODING (BACK END)

```
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const Admin = require('../models/Admin');

const router = express.Router();

// Admin Registration
router.post('/register', async (req, res) => {
    const { username, password } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);

    const newAdmin = new Admin({ username, password: hashedPassword });
    await newAdmin.save();
    res.status(201).send("Admin registered");
});

// Admin Login
router.post('/login', async (req, res) => {
    const { username, password } = req.body;
    const admin = await Admin.findOne({ username });

    if (admin && (await bcrypt.compare(password, admin.password))) {
        const token = jwt.sign({ id: admin._id }, process.env.JWT_SECRET, { expiresIn:
'1h' });
        res.json({ token });
    } else {
        res.status(401).send("Invalid credentials");
    }
});

module.exports = router;

const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const Customer = require('../models/Customer');

const router = express.Router();

// Customer Registration
router.post('/register', async (req, res) => {
    const { username, password } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);

    const newCustomer = new Customer({ username, password: hashedPassword });
    await newCustomer.save();
    res.status(201).send("Customer registered");
});

// Customer Login
router.post('/login', async (req, res) => {
    const { username, password } = req.body;
    const customer = await Customer.findOne({ username });

    if (customer && (await bcrypt.compare(password, customer.password))) {
        const token = jwt.sign({ id: customer._id }, process.env.JWT_SECRET, {
expiresIn: '1h' });
        res.json({ token });
    } else {
        res.status(401).send("Invalid credentials");
    }
});

// Upload prescription
router.post('/upload-prescription', async (req, res) => {
    const { userId, prescription } = req.body; // prescription is the file path or URL
    await Customer.findByIdAndUpdate(userId, { $push: { prescriptions: prescription }
});
    res.send("Prescription uploaded");
});

module.exports = router;
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const DeliveryBoy = require('../models/DeliveryBoy');

const router = express.Router();

// Delivery Boy Registration
```

```
router.post('/register', async (req, res) => {
    const { username, password } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);

    const newDeliveryBoy = new DeliveryBoy({ username, password: hashedPassword });
    await newDeliveryBoy.save();
    res.status(201).send("Delivery Boy registered");
});

// Delivery Boy Login
router.post('/login', async (req, res) => {
    const { username, password } = req.body;
    const deliveryBoy = await DeliveryBoy.findOne({ username });

    if (deliveryBoy && (await bcrypt.compare(password, deliveryBoy.password))) {
        const token = jwt.sign({ id: deliveryBoy._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
        res.json({ token });
    } else {
        res.status(401).send("Invalid credentials");
    }
});

module.exports = router;
```

10. SYSTEM TESTING

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 functionality of components, sub-assemblies, 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 test. Each test type addresses a specific testing requirement.

10.1 TEST CASES

UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows

SYSTEM TEST

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.

WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

10.2 PERFORMANCE ANALYSIS

The pharmacy management system is designed to manage interactions between customers, admin, delivery personnel, and a centralized database. Performance analysis is essential to ensure the system can handle the expected workload efficiently, maintain security, and provide a smooth user experience. Several aspects impact its overall performance, including response time, database management, scalability, and security.

Response Time

The system must handle actions such as logging in, uploading prescriptions, and managing orders quickly to avoid user frustration. Large file uploads, especially for prescriptions, can delay processing if not handled optimally. Implementing asynchronous operations and content delivery networks (CDNs) can mitigate these delays. Additionally, limiting the amount of data returned by API calls improves speed.

Database Performance

Efficient database management is vital as the system stores critical information like prescriptions, orders, and transactions. Indexing frequently accessed fields (such as order statuses improves the speed of data retrieval. Database sharding ensures that as the amount of data grows, the system remains fast and responsive. Optimized queries further enhance performance, especially when dealing with complex interactions like filtering and viewing transaction history.

Scalability and Load Handling

As the pharmacy system scales, it must handle increasing users and orders without compromising performance. Horizontal scaling, where more server instances are added, can distribute the workload more evenly. Load balancing also ensures that no single server is overwhelmed by traffic, maintaining system availability. Using caching mechanisms for frequently accessed data, such as delivery status updates, reduces the load on the database and accelerates response times.

Security and Authentication

Given the sensitive nature of medical and payment data, robust security measures are crucial. Encryption of data both at rest and in transit protects it from unauthorized access. Role-based access control ensures that only authorized personnel access certain information, preventing data leaks. Implementing two-factor authentication for admin accounts adds another layer of protection against breaches.

User Satisfaction

Optimized user interface design and efficient backend processes are key to providing a seamless experience. Features like real-time updates via WebSockets for order statuses reduce the need for manual refreshes, enhancing user interaction. Proper error handling ensures that users receive clear feedback when issues arise, improving overall satisfaction.

Conclusion

By optimizing response times, database performance, and scalability while ensuring robust security, the pharmacy management system can handle a growing user base effectively. These improvements guarantee smooth interactions for all user roles—customers, admin, and delivery personnel—leading to a better overall experience.

11. CONCLUSION

The pharmacy management system plays a pivotal role in transforming the way pharmacies operate in a digital world. By integrating core functionalities like customer order management, prescription uploads, real-time delivery tracking, and administrative control, the system addresses key challenges faced by traditional pharmacies. One of the most significant contributions is the streamlining of the medicine ordering process, which enhances convenience for customers who can now upload prescriptions, track their orders, and make secure payments, all through a web-based platform.

From a performance standpoint, the system is built to handle a variety of user roles—admin, customers, and delivery personnel—each with distinct tasks. To ensure a smooth experience for all users, the system incorporates asynchronous operations, effective database management, and scalability features. This allows for quick processing of user requests, even under heavy workloads. The use of CDNs for large file uploads, optimized database queries, and caching of frequently accessed data ensure that the system can handle high traffic without compromising performance.

Furthermore, the system's scalability is designed to accommodate the growing needs of a pharmacy, whether that involves handling a larger customer base, processing an increasing number of orders, or expanding its geographic reach. Horizontal scaling and load balancing mechanisms help manage server loads, ensuring that system performance remains stable even during peak times. These measures, coupled with real-time updates via WebSockets, create a dynamic and responsive user experience.

Security is another critical aspect of the system, particularly given the sensitive nature of the data it handles, such as medical prescriptions and payment details. The system employs robust security protocols, including SSL encryption for data in transit, role-based access control (RBAC) to limit user access to appropriate data, and two-factor authentication (2FA) for administrators. These features ensure that the system adheres to the highest standards of data privacy and protection, minimizing the risk of data breaches.

In terms of user satisfaction, the pharmacy management system significantly improves the user experience. Customers benefit from the convenience of an online ordering system, while administrators enjoy the efficiency of managing orders and delivery personnel from a single platform. Delivery personnel can easily access order details, update delivery statuses, and manage their workload, resulting in timely deliveries and improved service quality.

Overall, this pharmacy management system is a comprehensive solution that enhances the efficiency, security, and scalability of pharmacy operations. By leveraging advanced technologies and best practices in system design, it not only meets the immediate needs of the pharmacy but also positions it to adapt to future demands. The result is a more reliable, secure, and user-friendly platform that improves operational workflows and ensures a higher level of customer satisfaction in the long run.

In addition to its core functionalities, the system's integration of modern technologies positions it as a forward-thinking solution for pharmacies. The architecture allows for future enhancements, such as AI-driven inventory management and predictive analytics for customer demand, ensuring the pharmacy stays ahead of trends. Its modular design makes it adaptable to evolving business needs, and the strong emphasis on user experience ensures that both customers and staff benefit from its features. This comprehensive solution not only improves operational efficiency but also sets the foundation for continued growth and innovation in the pharmacy sector.

11.1 FUTURE ENHANCEMENT

Exponential change is accelerating disruption across the health care value chain and transforming the future of pharmacy. Clinical and technology breakthroughs are occurring at a record pace, building on the power of artificial intelligence (AI), robotics, and insights derived from radically interoperable data. As "imprecision medicine"1 shifts to precision treatments, the role of the pharmacist and the delivery channels we know today are likely to change. This combination may bring about a move from a fee-for-service reimbursement model to a value-based model, aligning pharmacy with the broader payer shifts underway.

Treatments would no longer be focused on chemical and biologic solutions, but instead focus on digital therapeutics, nutraceuticals, implants, gene editing,10 and programmable bacteria accelerated by clinical research.

Retail pharmacies could become consolidated health destinations with product distribution altered by 3D printing, kiosks, telehealth, and same-day delivery by driverless cars, autonomous bots, and drones.

Automation and AI algorithms would enhance pharmacists' responsibilities, allowing them to become recognized as care providers, ultimately prescribing acute medications and managing chronic diseases.

Massive data sets connected by Internet of Things (IoT) connected devices, cloud-based algorithms, and quantum computing could enable real-time diagnosis and insights that are integrated into our daily lives and shared across care providers.

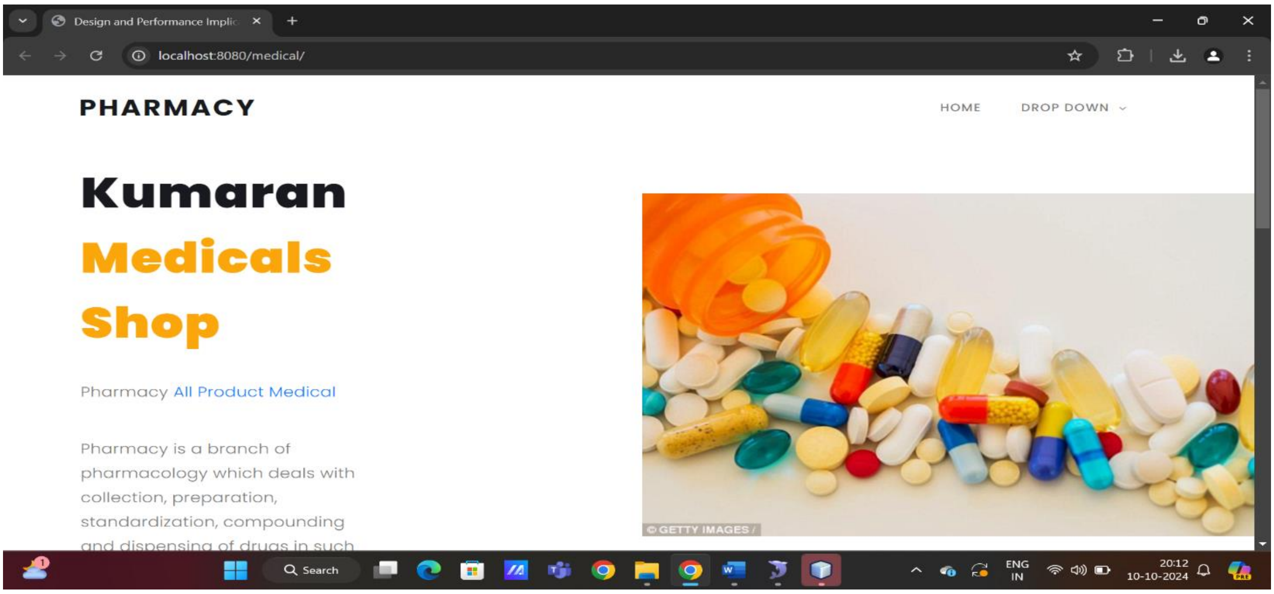
Innovation is happening across the life sciences:

- Researchers are developing smart mirrors that use advanced cameras and your breath to detect health variations.
- Multiple companies are testing and working on home health care bots that can perform basic services, while elderly workers in Japan are using exoskeletons to extend their ability to perform manual labor.
- Smartphones are evolving to allow them to act as point-of-care and home health diagnostic tools for conditions such as urinary tract infections or diabetic eye disease

12. APPENDICES

12.1 SAMPLE SCREENSOTS

HOME PAGE



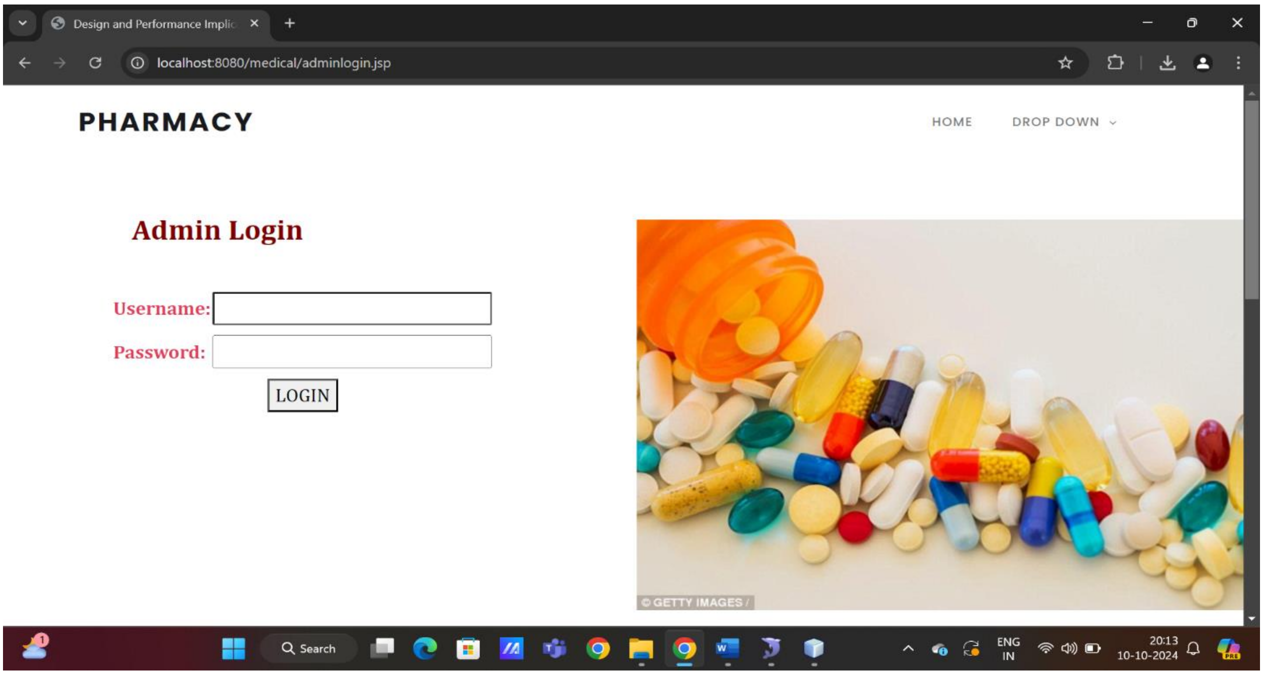
Kumaran Medicals Shop

Pharmacy All Product Medical

Pharmacy is a branch of pharmacology which deals with collection, preparation, standardization, compounding and dispensing of drugs in such

Fig 12. Home Page Screenshot

ADMIN LOGIN PAGE


Design and Performance Implic
localhost:8080/medical/adminlogin.jsp
PHARMACY
HOME
DROP DOWN
© GETTY IMAGES /
Search
ENG
IN
20:13
10-10-2024

Admin Login

Username:

Password:

LOGIN

Fig 13. Admin Login Page Screenshot

CUSTOMER REGISTRATION PAGE

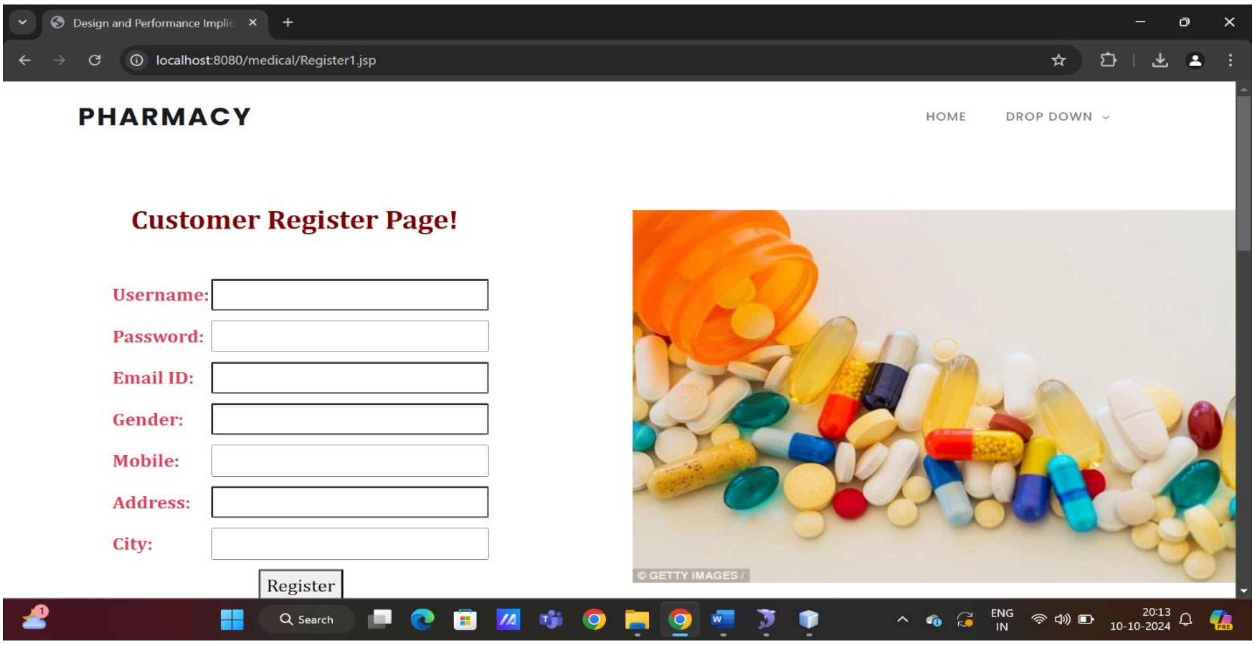


Fig 14 Customer Registration Page Screenshot

CUSTOMER LOGIN PAGE

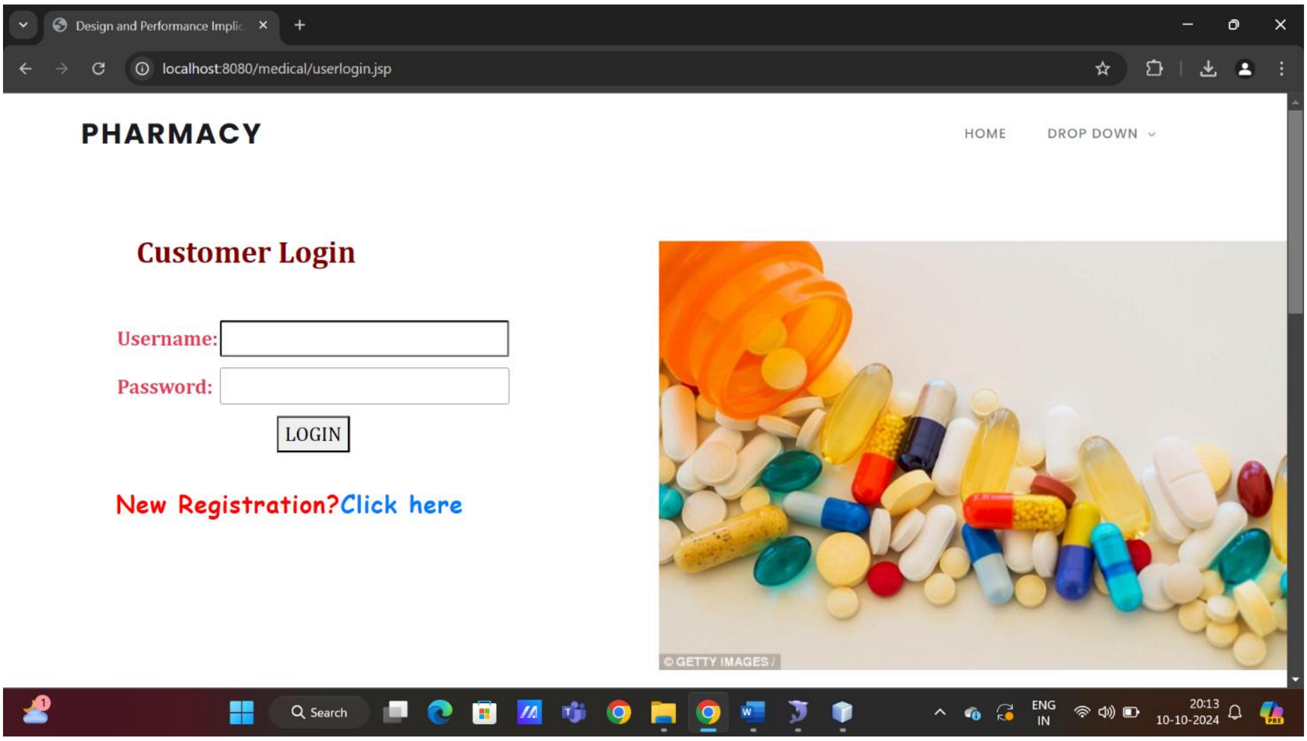
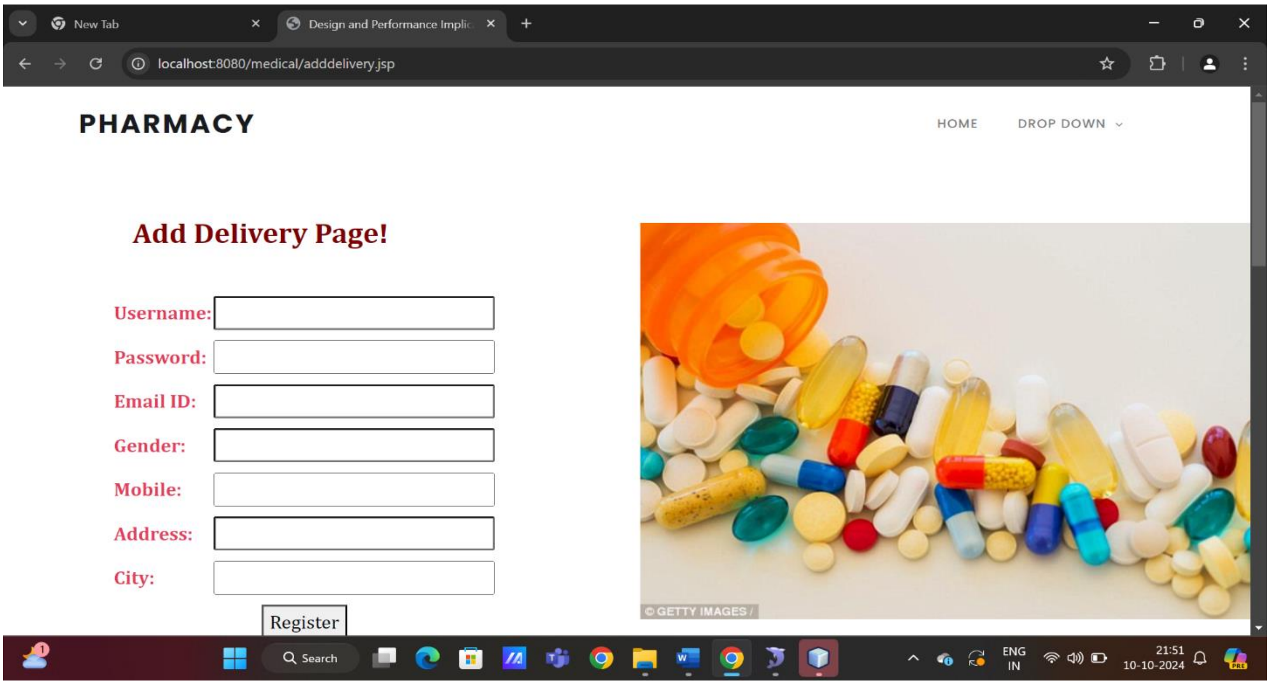


Fig 15 Customer Login Page Screenshot

DELIVERY BOY REGISTRATION



localhost:8080/medical/adddelivery.jsp

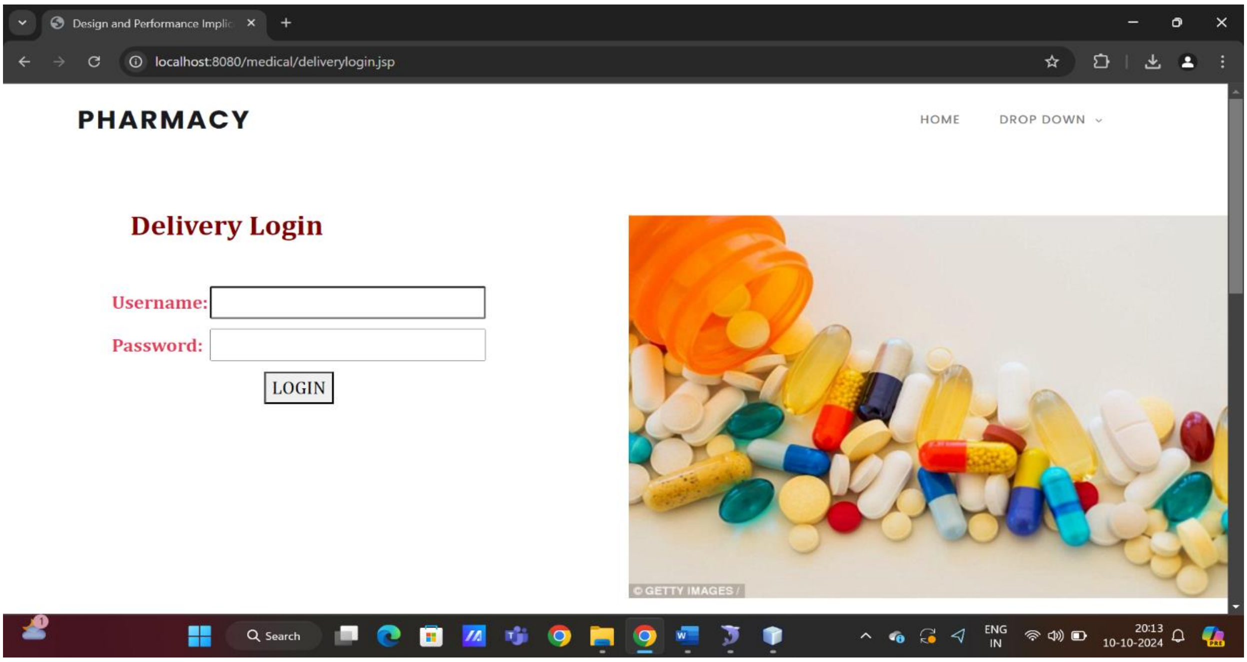
PHARMACY

Add Delivery Page!

Username:	
Password:	
Email ID:	
Gender:	
Mobile:	
Address:	
City:	

Fig 16. Delivery boy Registration Page Screenshot

DELIVERY LOGIN PAGE



localhost:8080/medical/deliverylogin.jsp

PHARMACY

Delivery Login

Username:	
Password:	

Fig 17. Delivery Page Login Screenshot

13. REFERENCES

1. Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. The Knowledge Engineering Review, 10
2. J. Fox, M. Beveridge and D. Glasspool. Understanding intelligent agents: analysis and synthesis. AI Communications, 2003, vol 16, pp 139-152.
3. Joseph Weizenbaum. Computer Power and Human Reason (Penguin Books Ltd, 1976).
4. A. Seipp. Intelligent Agent. School of Library, Archival and Information Studies, University of British Columbia, December 2001.
5. J. M. Epstein and R. Axtell. Growing artificial societies: social science from the bottom up. Brookings Institution Press. p. 224. ISBN 978-0- 262-55025-3, October 11, 1996).
6. Construct, Computational Analysis of Social Organizational Systems.
7. [7] Douglas A. Samuelson. Designing Organizations. OR/MS Today (Institute for Operations Research and the Management Sciences), December 2000.
8. Douglas A. Samuelson. Agents of Change. OR/MS Today (Institute for Operations Research and the Management Sciences), February 2005.
9. Douglas A. Samuelson and Charles M. Macal. Agent-Based Modeling Comes of Age. OR/MS Today (Institute for Operations Research and the Management Sciences), August 2006.
10. Ron Sun. Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation. Cambridge University Press. ISBN 0-521-83964- 5, 2006.
11. Katia P. Sycara. Multi agent systems. AI magazine Volume 19, No.2 Intelligent Agents Summer, 1998.
12. E. H. Durfee and V. Lesser. Negotiating Task Decorrposinon and Allication Using Partial Global Planning. In Distributed Artificial Intelligence, Volume 2, e ds. L Gasser and M. Hulms. 229-244. San Francisco,Calif: Morgan Kaufinann, 1989.

13. S. Rao and M. P. George. Toward a Formal Theory of Deliberation and Its Role in the Formation of Intentions. Technical Report, Australian Artificial Intelligence Institute, Victoria, Australia, 1991.

14. Yoav Shoham. Agent-Oriented Programming. Artificial Intelligence 60(1): 51-92, 1993.

15. K. Decker, A. Pamru, Katia P. Sycara. and M. williamson. Designing Behaviors for Information Agents. In Proceedings of the First International Conference on Autonomous Agents (.4gent-97), 404-412. New York: Association of Computng Mach:Dery, 1997.