DECLARATION

Hereby certify that the six months of training work which is being presented in this report by **Nitin Garg** in partial fulfilment of requirements for the award of the degree of B.Tech. In Computer Science and Engineering submitted in the CSE section, Yadavindra Department of Engineering, Punjabi University Guru Kashi Campus, Talwandi Sabo is done by me from the period of **14 June 2025 to 28 July 2025** under the supervision of **Ms. Akshita Bhradwaj, Trainer at** Excellence technology, Chandigarh**.**

Nitin Garg
Uni Roll No. 22301115
B.Tech C.S.E. 4th Sem

PREFACE

As a part of the B. TECH C.S.E. (Semester 4th) Curriculum and to gain practical Knowledge in the field of JAVA, I am required to make a report on **“PHARMACY HUB”** The Basic Objective behind this project report is to get knowledge about the project. In this project report, I have included the information about the project. Doing this Project report helped me to enhance my knowledge regarding the Project the working of the Project and how the Project is collecting funds. Through this report, I come to know about the importance of hard work and the role of devotion to the work. It’s been a great experience working on this project.

ABSTRACT

The ***PHARMA HUB*** is a Java-based desktop application designed to automate and manage the day-to-day operations of a medical store or pharmacy in an efficient and organized manner. The main objective of the system is to replace the traditional manual methods of managing medicines, billing, and customer records with a computerized solution that is fast, accurate, and user-friendly. The system provides an integrated platform where the pharmacy staff can handle inventory, generate bills, and manage medicine records without the need for complex paperwork.This application is developed using **Java and the Swing framework** to create an interactive and visually appealing graphical user interface. The system includes various modules such as **Login, Inventory Management, Add Item, Billing, Payment, and Report Generation**. A secure login system is implemented to differentiate between the *Admin* and the *Pharmacist*, with restricted access to certain features like adding new medicines for non-admin users. This role-based access control improves the security and integrity of the system.The inventory module stores information about medicines such as name, quantity, price, manufacture date, and expiry date. All medicines are maintained in a centralized database implemented using Java's **ArrayList** collection. The system automatically updates stock levels when a medicine is added to the bill or removed from the cart. A dynamic search feature allows the user to quickly find medicines from the inventory.The billing module is one of the most significant features of the system. It allows the user to select medicines from the inventory, enter the required quantity, calculate the subtotal, apply GST, and generate the final bill. A separate payment window is provided for entering the paid amount and calculating the balance or change. After successful payment, a **PDF invoice is automatically generated** using the iText library. The invoice contains full details of the pharmacy, customer information, medicine list, total amount, GST, and payment details, which can be stored for future reference.

ACKNOWLEDGEMENT

I am highly grateful to **Dr. Navdeep Goyal_Head of Yadavindra Department of Engineering, Talwandi Sabo,** for providing this opportunity to carry out the six months of industrial training at Excellence technology, Chandigarh. The constant guidance and encouragement received from **DR. MEENAKSHI BANSAL(Assistant Prof. Placement Coordinator and Head Incharge) YDoE, Talwandi Sabo** has been of great help in carrying out the project work and is acknowledged with reverential thanks. The help was rendered by **Ms. Akshita Bhradwaj, Trainer at** Excellence technology, Chandigarh for experimentation is greatly acknowledged. I express gratitude to all faculty members of the ECE Section, YDoE, Talwandi Sabo for their intellectual support, continuous encouragement, and supervision throughout the training period.

Nitin Garg

Uni Roll No. 22301115
B.Tech C.S.E. 4th Sem

COMPANY PROFILE

Excellence Technology is an ISO 9001:2015 certified software development and industrial training company based in Mohali, Punjab, India. Established in 2010, the company specializes in delivering high-quality software solutions and professional training programs. With a focus on innovation and client satisfaction, Excellence Technology serves both domestic and international clients across countries such as the USA, UK, Germany, and Australia.The company offers a comprehensive range of services, including mobile and web application development for iOS, Android, and Windows platforms, e-commerce solutions, back-end development, and Magento customization. Its technology stack incorporates modern frameworks and tools such as Bootstrap, AngularJS, MongoDB, JavaScript, Ionic, and React Native, ensuring robust and scalable software products. In addition to software services, Excellence Technology provides industry-oriented training programs designed to equip students and professionals with practical skills in emerging technologies. Training areas include PHP, Java, Python, web designing, machine learning, and software testing, helping learners stay competitive in today’s dynamic IT landscape. Through its commitment to quality, innovation, and continuous improvement, Excellence Technology has established itself as a trusted name in software development and IT training.

TABLE OF CONTENTS

INTRODUCTION

The Pharma Hub project is a comprehensive Java-based desktop application meticulously designed to digitalize and modernize the operations of a pharmacy. The software aims to minimize human effort, reduce manual errors, and provide a fast, efficient, and secure way of managing pharmaceutical data. Built using Java Swing, one of the most robust frameworks for desktop GUI development, Pharma Hub provides an interactive, user-friendly interface that caters to both administrators and pharmacists, enabling them to handle essential pharmacy functions such as inventory management, billing, and customer transactions in a seamless manner. The entire system has been conceptualized and developed following modular design principles, ensuring that it is highly scalable, maintainable, and adaptable to future requirements.

At its core, the Pharma Hub application is designed to replace traditional pen-and-paper methods of pharmacy management. In many small and medium-sized pharmacies, daily operations such as maintaining stock records, calculating bills, tracking expiry dates, and managing sales are still performed manually. This manual process often leads to human errors, time wastage, and difficulty in retrieving historical data. The motivation behind designing Pharma Hub was to overcome these limitations by developing a software solution that could store, retrieve, and process pharmacy-related information in a systematic, automated manner. The result is a well-organized system that centralizes all pharmacy operations under one platform, offering improved accuracy, security, and efficiency.

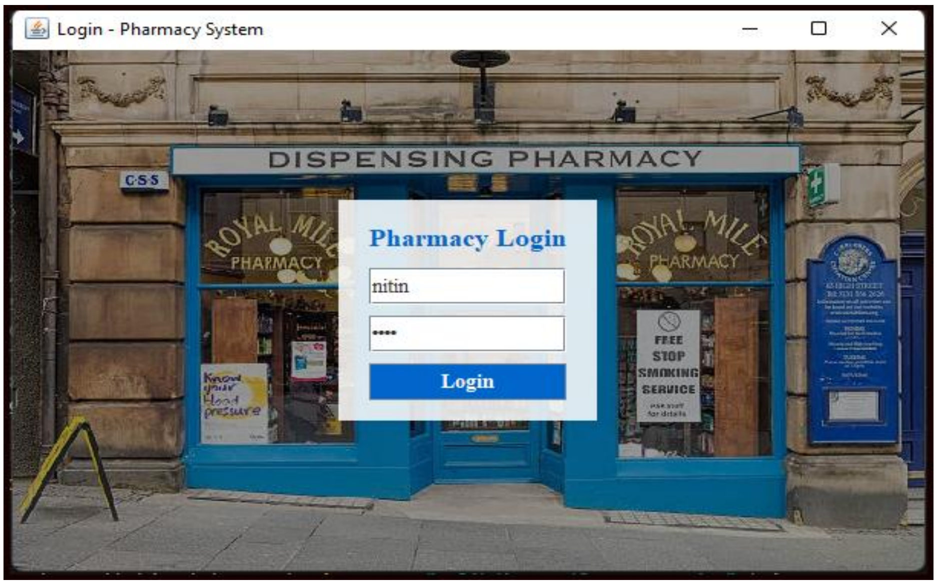
The design of Pharma Hub follows an object-oriented approach, where each component of the system is implemented as a separate class, encapsulating its specific responsibilities and behaviour. The application is composed of multiple interlinked Java classes, such as Login Window, MainMenuWindow, Inventory Window, AddItemWindow, Billing Window, Payment Window, Pharmacy Database, and Item. Each class plays a distinct role in the system's operation, yet they all interact harmoniously to provide a unified and consistent user experience. This modular design not only enhances readability and maintainability but also allows for future scalability. For instance, new functionalities such as supplier management, sales reporting, or online order tracking can be easily integrated without disrupting the existing code structure.

CHAPTER 1

LOGIN WINDOW

The Login Window class serves as the entry point to the system, ensuring that only authorized users can access the application. Two types of users are defined in the current version — the Admin and the Pharmacist. The admin user has full control over the system, including the ability to add new items to the inventory and modify existing records, while the pharmacist has limited access, restricted to viewing and billing operations. The login interface is aesthetically designed with a background image overlay, creating a professional and visually appealing environment. User credentials are verified through a simple authentication mechanism, which, though basic in this prototype version, demonstrates the concept of role-based access control. Once the credentials are validated, users are redirected to the MainMenuWindow, which acts as the central dashboard of the system.

```
import javax.swing.*;
import java.awt.*;

public class LoginWindow extends JFrame {
    JTextField userField;
    JPasswordField passField;

    public LoginWindow() {
        setTitle("Login - Pharmacy System");
        setSize(612, 421);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // Background image panel with light overlay
        BackgroundPanel panel = new BackgroundPanel("images/login.jpg") {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.setColor(new Color(0, 0, 0, 100)); // light transparent overlay
                g.fillRect(0, 0, getWidth(), getHeight());
            }
        };
        panel.setLayout(new GridBagLayout());

        // Login panel
        JPanel loginPanel = new JPanel(new GridLayout(4, 1, 8, 8));
        loginPanel.setBackground(new Color(255, 255, 255, 220));
        loginPanel.setBorder(BorderFactory.createEmptyBorder(15, 20, 15, 20));

        JLabel title = new JLabel("Pharmacy Login", JLabel.CENTER);
        title.setFont(new Font("Times new roman", Font.BOLD, 18));
        title.setForeground(new Color(0, 102, 204));
        userField = new JTextField();
        userField.setFont(new Font("Times new roman", Font.PLAIN, 14));

        passField = new JPasswordField();
        passField.setFont(new Font("Times new roman", Font.PLAIN, 14));

        JButton loginBtn = new JButton("Login");
        loginBtn.setBackground(new Color(0, 102, 204));
        loginBtn.setForeground(Color.WHITE);
        loginBtn.setFont(new Font("Times new roman", Font.BOLD, 14));
        loginBtn.setFocusPainted(false);
        loginBtn.addActionListener(e -> authenticate());

        loginPanel.add(title);
        loginPanel.add(userField);
        loginPanel.add(passField);
        loginPanel.add(loginBtn);

        panel.add(loginPanel);
        add(panel);
        setVisible(true);
    }

    void authenticate() {
        String u = userField.getText();
        String p = new String(passField.getPassword());

        if ((u.equals("nitin") && p.equals("0000")) || (u.equals("garg") && p.equals("1234"))) {
            JOptionPane.showMessageDialog(this, "Login successful as " + (u.equals("nitin") ?
"Admin" : "Pharmacist"));
            dispose();
            new MainMenuWindow(u.equals("nitin") ? "Admin" : "Pharmacist");
        } else {
            JOptionPane.showMessageDialog(this, "Invalid credentials!");
        }
    }
}
```



nitin

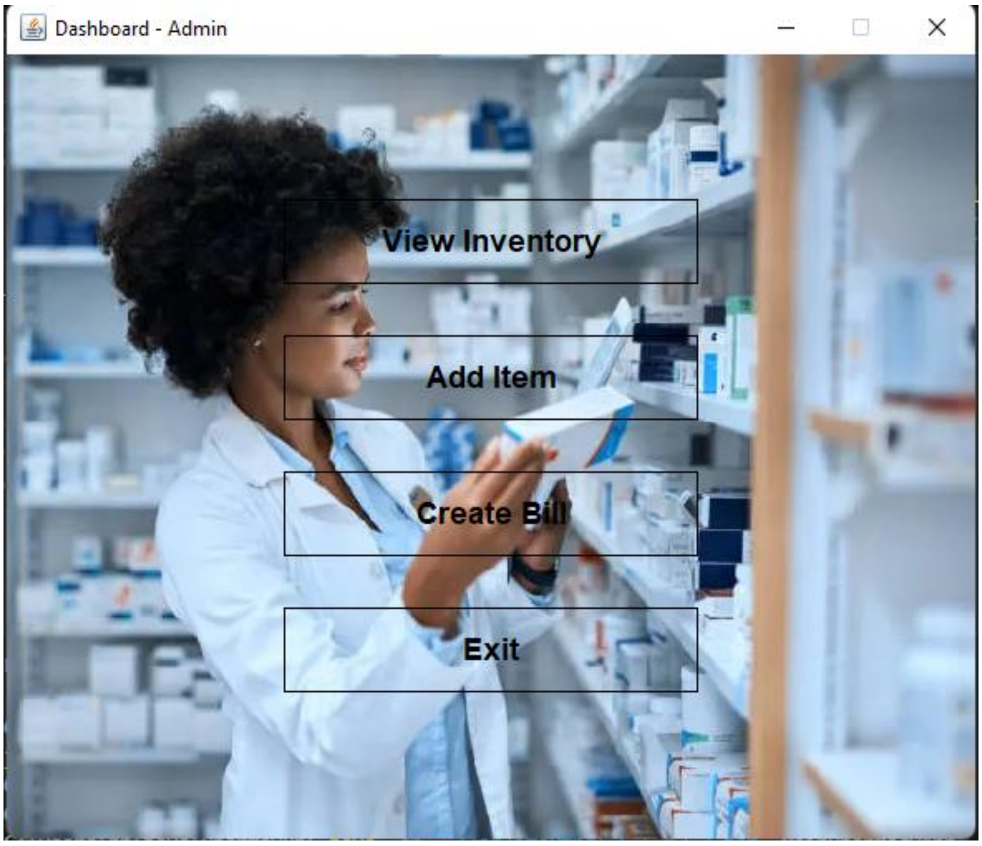
••••

CHAPTER 2

MAIN MENU WINDOW

The MainMenuWindow serves as the hub from which users can navigate to different functional areas of the application. It presents a clean and organized interface with large, easily accessible buttons for key operations — “View Inventory,” “Add Item,” “Create Bill,” and “Exit.” These options are visually aligned in the center of the screen against a background image, giving the interface a modern look. The buttons are styled using Swing’s font and color properties to maintain design consistency throughout the application. The design also incorporates role-based button control — the “Add Item” button is disabled when the user logs in as a Pharmacist, preventing unauthorized modifications to the inventory. Such design decisions demonstrate the use of access restriction at the user interface level, which enhances system security and data integrity.

```
import javax.swing.*;
import java.awt.*;

public class MainMenuWindow extends JFrame {
    public MainMenuWindow(String role) {
        setTitle("Dashboard - " + role);
        setSize(600, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);

        // Background image panel
        BackgroundPanel panel = new BackgroundPanel("images/inventory.jpg");
        panel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(15, 0, 15, 0); // space between buttons
        gbc.gridx = 0;

        // Button style
        Font buttonFont = new Font("Arial", Font.BOLD, 18);

        JButton inventoryBtn = createStyledButton("View Inventory", buttonFont);
        JButton addBtn = createStyledButton("Add Item", buttonFont);
        JButton billBtn = createStyledButton("Create Bill", buttonFont);
        JButton exitBtn = createStyledButton("Exit", buttonFont);

        // Add actions
        inventoryBtn.addActionListener(e -> new InventoryWindow());
        addBtn.addActionListener(e -> new AddItemWindow());
        billBtn.addActionListener(e -> new BillingWindow());
        exitBtn.addActionListener(e -> System.exit(0));

        // Add buttons to layout as rows
        gbc.gridy = 0; panel.add(inventoryBtn, gbc);
        gbc.gridy = 1; panel.add(addBtn, gbc);
        gbc.gridy = 2; panel.add(billBtn, gbc);
        gbc.gridy = 3; panel.add(exitBtn, gbc);

        // Disable addBtn if role is Pharmacist
        if (role.equalsIgnoreCase("Pharmacist")) addBtn.setEnabled(false);

        add(panel);
        setVisible(true);
    }

    // Helper to create styled transparent buttons
    private JButton createStyledButton(String text, Font font) {
        JButton btn = new JButton(text);
        btn.setFont(font);
        btn.setPreferredSize(new Dimension(250, 50));
        btn.setForeground(Color.BLACK);
        btn.setFocusPainted(false);
        btn.setOpaque(false);
        btn.setContentAreaFilled(false);
        btn.setBorder(BorderFactory.createLineBorder(Color.BLACK));
        return btn;
    }
}
}
```

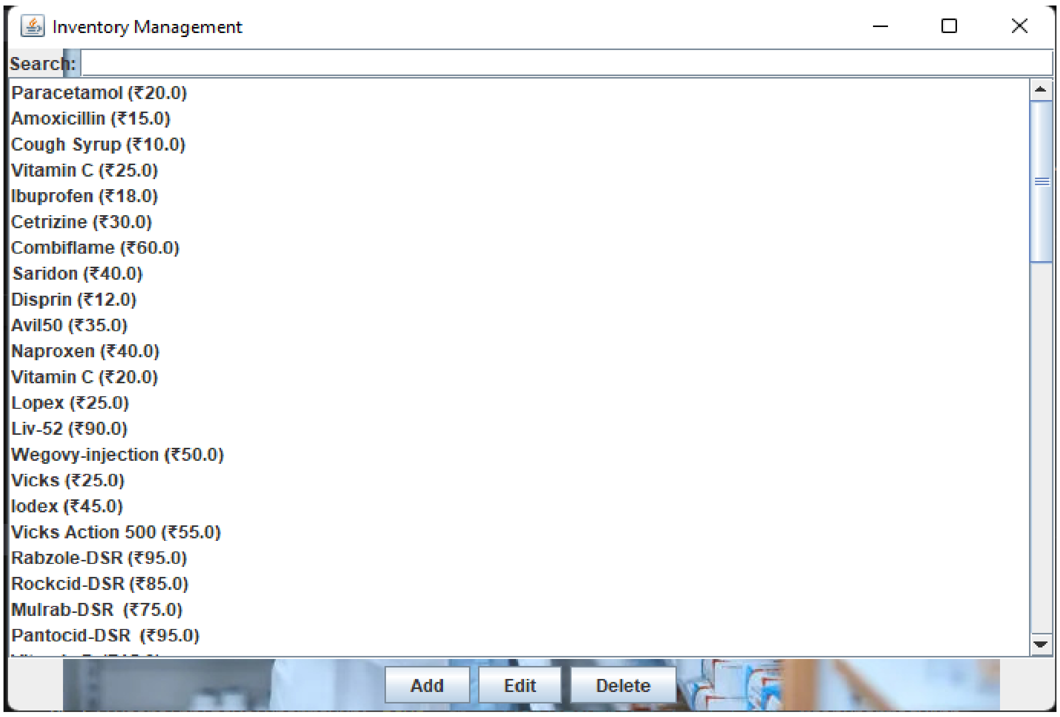


CHAPTER 3

INVENTORY WINDOW

The Inventory Window is one of the most crucial modules in the Pharma Hub system, designed to handle all aspects of inventory management. It displays the current list of medicines available in the store, along with details such as quantity, price, manufacture date, and expiry date. The window employs a JList component to show the medicines in a scrollable format, while a DefaultListModel dynamically updates the list whenever changes are made. The design includes a real-time search feature, allowing users to quickly locate medicines by typing partial or full names into a search bar. This enhances user efficiency and eliminates the need for manual scrolling through long lists.One of the major strengths of the InventoryWindow design lies in its integration with the PharmacyDatabase class, which stores all medicine data in a static list accessible by all modules. This shared data structure ensures that every modification in one part of the system — whether adding, editing, or deleting an item — is instantly reflected across all other windows. The InventoryWindow also supports CRUD (Create, Read, Update, Delete) operations. Users can add new items through the AddItemWindow, edit selected items to update their details, or remove expired or discontinued products from the database. The design employs clear prompts and validation to prevent invalid data entry. For instance, when a user tries to input non-numeric data for quantity or price, the system displays an error message, maintaining the integrity of the stored data.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class InventoryWindow extends JFrame {
    private DefaultListModel<Item> inventoryModel = new DefaultListModel<>();
    private JList<Item> inventoryList = new JList<>(inventoryModel);
    private JTextField searchField = new JTextField();
    private JButton addBtn = new JButton("Add");
    private JButton editBtn = new JButton("Edit");
    private JButton deleteBtn = new JButton("Delete");

    public InventoryWindow() {
        setTitle("Inventory Management");
        setSize(700, 500);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        // Background image
        ImageIcon bgImg = new ImageIcon("images/inventory.jpg"); // <-- put your image path here
        JLabel bgLabel = new JLabel(bgImg);
        bgLabel.setLayout(new BorderLayout());
        setContentPane(bgLabel);

        loadInventoryItems();

        // Top panel for search
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.setOpaque(false);
        topPanel.add(new JLabel("Search: "), BorderLayout.WEST);
        topPanel.add(searchField, BorderLayout.CENTER);
        add(topPanel, BorderLayout.NORTH);

        // Center panel for item list
        JScrollPane scrollPane = new JScrollPane(inventoryList);
        scrollPane.setOpaque(false);
        scrollPane.getViewport().setOpaque(false);
        add(scrollPane, BorderLayout.CENTER);

        // Bottom panel for buttons
        JPanel bottomPanel = new JPanel(new FlowLayout());
        bottomPanel.setOpaque(false);
        bottomPanel.add(addBtn);
        bottomPanel.add(editBtn);
        bottomPanel.add(deleteBtn);
        add(bottomPanel, BorderLayout.SOUTH);

        // Event: search
        searchField.getDocument().addDocumentListener(new
javax.swing.event.DocumentListener() {
            public void insertUpdate(javax.swing.event.DocumentEvent e) { filterList(); }
            public void removeUpdate(javax.swing.event.DocumentEvent e) { filterList(); }
            public void changedUpdate(javax.swing.event.DocumentEvent e) { filterList(); }
        });

        addBtn.addActionListener(e -> addNewItem());
        editBtn.addActionListener(e -> editSelectedItem());
        deleteBtn.addActionListener(e -> deleteSelectedItem());

        setVisible(true);
    }

    private void loadInventoryItems() {
        inventoryModel.clear();
        for (Item i : PharmacyDatabase.inventory) {
            inventoryModel.addElement(i);
        }
    }

    private void filterList() {
        String query = searchField.getText().toLowerCase();
        inventoryModel.clear();
        for (Item i : PharmacyDatabase.inventory) {
            if (i.name.toLowerCase().contains(query)) {
                inventoryModel.addElement(i);
            }
        }
    }

    private void addNewItem() {
        String name = JOptionPane.showInputDialog(this, "Enter Medicine Name:");
        if (name == null || name.trim().isEmpty()) return;
        try {
            int qty = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter Quantity:"));
            double price = Double.parseDouble(JOptionPane.showInputDialog(this, "Enter Price:"));
            String mfg = JOptionPane.showInputDialog(this, "Enter Manufacture Date (dd-mm-
yyyy):");
            String exp = JOptionPane.showInputDialog(this, "Enter Expiry Date (dd-mm-yyyy):");
            Item newItem = new Item(name, qty, price, mfg, exp);
            PharmacyDatabase.inventory.add(newItem);
            loadInventoryItems();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Invalid input.");
        }
    }

    private void editSelectedItem() {
        Item selected = inventoryList.getSelectedValue();
        if (selected == null) return;
        try {
            String name = JOptionPane.showInputDialog(this, "Edit Name:", selected.name);
            int qty = Integer.parseInt(JOptionPane.showInputDialog(this, "Edit Quantity:",
selected.quantity));
            double price = Double.parseDouble(JOptionPane.showInputDialog(this, "Edit Price:",
selected.price));
            String mfg = JOptionPane.showInputDialog(this, "Edit Manufacture Date:",
selected.manufactureDate);
            String exp = JOptionPane.showInputDialog(this, "Edit Expiry Date:",
selected.expiryDate);
            selected.name = name;
            selected.quantity = qty;
            selected.price = price;
            selected.manufactureDate = mfg;
            selected.expiryDate = exp;
            loadInventoryItems();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Invalid input.");
        }
    }

    private void deleteSelectedItem() {
        Item selected = inventoryList.getSelectedValue();
        if (selected != null) {
            int confirm = JOptionPane.showConfirmDialog(this, "Are you sure?");
            if (confirm == JOptionPane.YES_OPTION) {
                PharmacyDatabase.inventory.remove(selected);
                loadInventoryItems();
            }
        }
    }
}
```



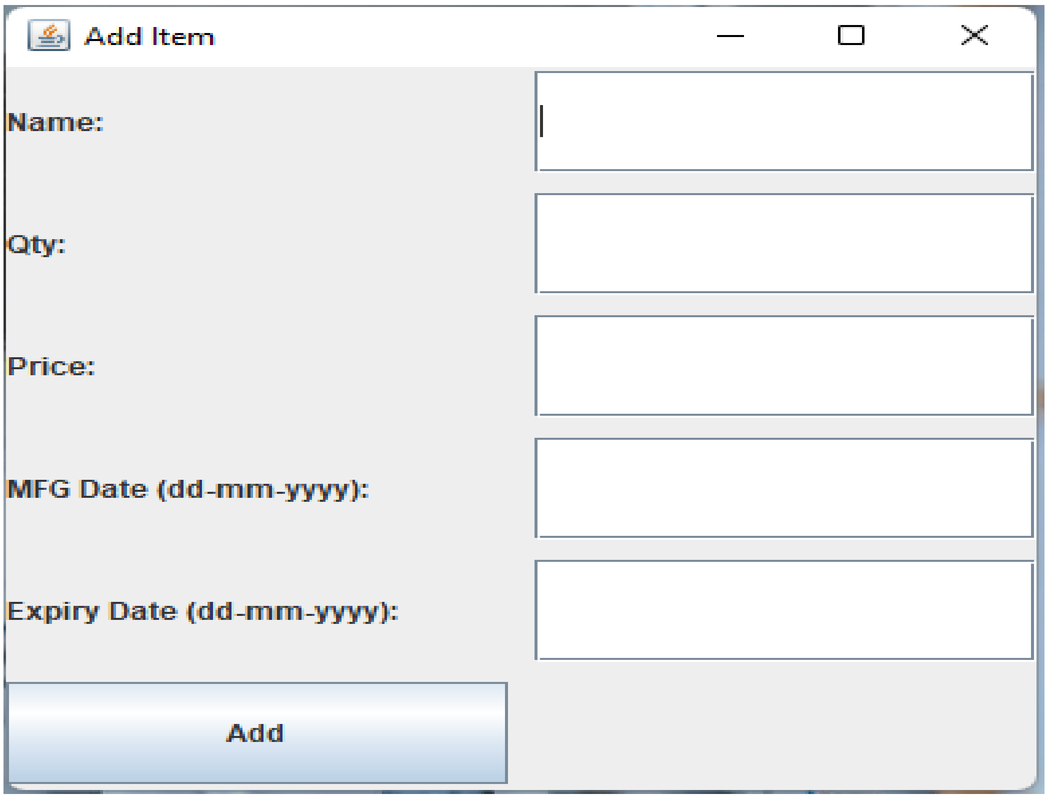
Paracetamol (₹20.0)
Amoxicillin (₹15.0)
Cough Syrup (₹10.0)
Vitamin C (₹25.0)
Ibuprofen (₹18.0)
Cetrizine (₹30.0)
Combiflame (₹60.0)
Saridon (₹40.0)
Disprin (₹12.0)
Avil50 (₹35.0)
Naproxen (₹40.0)
Vitamin C (₹20.0)
Lopex (₹25.0)
Liv-52 (₹90.0)
Wegovy-injection (₹50.0)
Vicks (₹25.0)
Iodex (₹45.0)
Vicks Action 500 (₹55.0)
Rabzole-DSR (₹95.0)
Rockcid-DSR (₹85.0)
Mulrab-DSR (₹75.0)
Pantocid-DSR (₹95.0)

CHAPTER 4

ADD ITEM WINDOW

The AddItemWindow module allows the admin to introduce new medicines into the inventory. Its design follows a simple, grid-based layout with labeled text fields for each attribute — name, quantity, price, manufacture date, and expiry date. A dedicated “Add” button processes the input, validates the data, and updates the centralized inventory list. The window also includes a status label that provides immediate feedback to the user, such as “Item Added Successfully” or “Error: Invalid Input.” This enhances the system’s usability and provides a smooth experience for users with varying levels of technical knowledge. The minimalistic design of this window ensures that even first-time users can easily understand the process of adding new items to the database.

```
import javax.swing.*;
import java.awt.*;

public class AddItemWindow extends JFrame {
    public AddItemWindow() {
        setTitle("Add Item");
        setSize(400, 400);
        setLocationRelativeTo(null);
        setLayout(new GridLayout(6, 2, 10, 10));

        // Input fields
        JTextField name = new JTextField();
        JTextField qty = new JTextField();
        JTextField price = new JTextField();
        JTextField mfgDate = new JTextField();
        JTextField expiryDate = new JTextField();
        JLabel status = new JLabel();

        // Buttons
        JButton addBtn = new JButton("Add");

        // Labels + Fields
        add(new JLabel("Name:")); add(name);
        add(new JLabel("Qty:")); add(qty);
        add(new JLabel("Price:")); add(price);
        add(new JLabel("MFG Date (dd-mm-yyyy):")); add(mfgDate);
        add(new JLabel("Expiry Date (dd-mm-yyyy):")); add(expiryDate);
        add(addBtn); add(status);

        // Action
        addBtn.addActionListener(e -> {
            try {
                String itemName = name.getText().trim();
                int itemQty = Integer.parseInt(qty.getText().trim());
                double itemPrice = Double.parseDouble(price.getText().trim());
                String mfg = mfgDate.getText().trim();
                String exp = expiryDate.getText().trim();

                if (itemName.isEmpty() || mfg.isEmpty() || exp.isEmpty()) {
                    status.setText("Please fill all fields!");
                    return;
                }

                // Add to inventory
                PharmacyDatabase.inventory.add(
                    new Item(itemName, itemQty, itemPrice, mfg, exp)
                );
                status.setText("Item Added!");

                // clear fields
                name.setText("");
                qty.setText("");
                price.setText("");
                mfgDate.setText("");
                expiryDate.setText("");
            } catch (Exception ex) {
                status.setText("Error: Invalid Input!");
            }
        });

        setVisible(true);
    }
}
```



CHAPTER 5

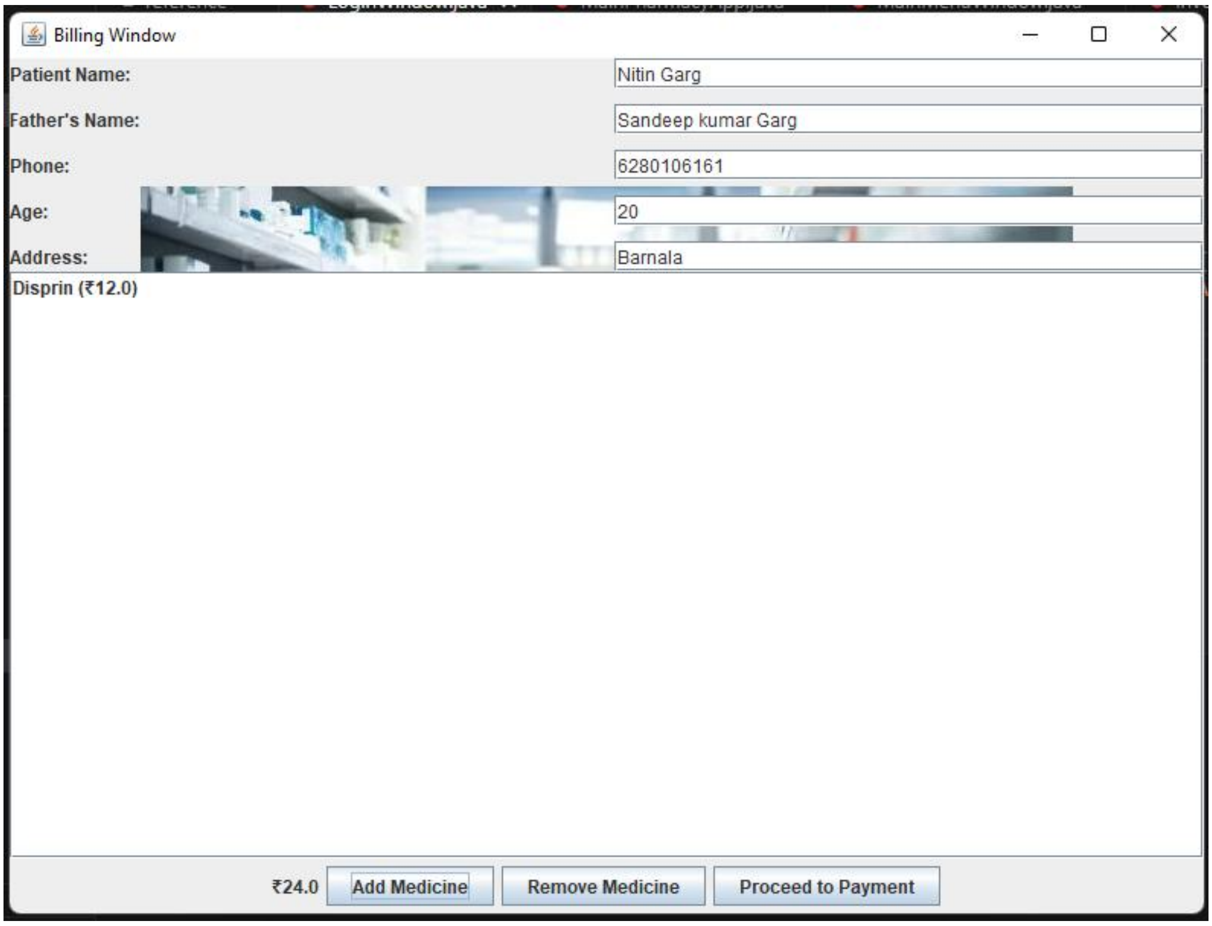
BILLINGWINDOW

The BillingWindow represents the financial core of the Pharma Hub system. It is designed to handle the entire sales process — from adding medicines to the customer’s cart, calculating totals, applying taxes, and generating bills. The design of this window is significantly more complex, as it integrates multiple components including medicine selection, payment calculation, and PDF generation. Customers’ details, such as name, father’s name, phone number, age, and address, are collected through neatly arranged text fields. The interface displays a cart list where selected items appear with their respective quantities and prices.

A notable aspect of the BillingWindow design is its integration with the iText PDF library, which enables the automatic generation of detailed invoice PDFs. Once the transaction is completed, the system compiles all relevant information — including the pharmacy’s GST number, itemized list of medicines, manufacturing and expiry dates, subtotal, GST amount, total payable, and balance — into a professionally formatted PDF file. The file is saved in a “bills” directory, providing digital documentation for every sale. This feature not only enhances professionalism but also ensures reliable recordkeeping, allowing pharmacies to maintain organized sales archives.

The BillingWindow also includes features for stock management during sales. When an item is added to the customer’s cart, its available quantity in the inventory automatically decreases. If the user tries to sell more than the available quantity, the system immediately displays an alert message, ensuring stock integrity. Similarly, when a medicine is removed from the cart, its quantity is restored to the inventory, reflecting real-time updates. The inclusion of such validation and synchronization logic demonstrates thoughtful system design aimed at preventing inventory inconsistencies.

```
// File: BillingWindow.java
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.FileOutputStream;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.text.SimpleDateFormat;
import java.util.Date;
import com.itextpdf.text.*;
import com.itextpdf.text.pdf.*;
import com.itextpdf.text.Font;
import com.itextpdf.text.Font.FontFamily;
import com.itextpdf.text.Rectangle;

public class BillingWindow extends JFrame {

    DefaultListModel<Item> cart = new DefaultListModel<>();
    JList<Item> cartList = new JList<>(cart);
    JLabel totalLabel = new JLabel("₹0.0");

    JTextField customerNameField = new JTextField();
    JTextField fatherNameField = new JTextField();
    JTextField customerPhoneField = new JTextField();
    JTextField customerAgeField = new JTextField();
    JTextField customerAddressField = new JTextField();

    private static final String GST_NUMBER = "29ABCDE1234F1Z5";
    private double finalAmount = 0.0;  // updated after payment

    public BillingWindow() {
        setTitle("Billing Window");
        setSize(800, 600);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        try {
            setContentPane(new JLabel(new ImageIcon("images/pharmacy.jpg")));
        } catch (Exception e) {
            setContentPane(new JPanel());
        }
        setLayout(new BorderLayout());

        // === Customer Info ===
        JPanel customerPanel = new JPanel(new GridLayout(5, 2, 10, 10));
        customerPanel.setOpaque(false);

        customerPanel.add(new JLabel("Patient Name:"));
        customerPanel.add(customerNameField);
        customerPanel.add(new JLabel("Father's Name:"));
        customerPanel.add(fatherNameField);
        customerPanel.add(new JLabel("Phone:"));
        customerPanel.add(customerPhoneField);
        customerPanel.add(new JLabel("Age:"));
        customerPanel.add(customerAgeField);
        customerPanel.add(new JLabel("Address:"));
        customerPanel.add(customerAddressField);
        add(customerPanel, BorderLayout.NORTH);

        // === Cart ===
        add(new JScrollPane(cartList), BorderLayout.CENTER);

        // === Bottom Panel ===
        JPanel bottomPanel = new JPanel(new FlowLayout());
        bottomPanel.setOpaque(false);
        JButton addBtn = new JButton("Add Medicine");
        JButton removeBtn = new JButton("Remove Medicine");
        JButton generateBtn = new JButton("Proceed to Payment");

        bottomPanel.add(totalLabel);
        bottomPanel.add(addBtn);
        bottomPanel.add(removeBtn);
        bottomPanel.add(generateBtn);
        add(bottomPanel, BorderLayout.SOUTH);

        // === Add Medicine ===
        addBtn.addActionListener(e -> {
            Item selected = (Item) JOptionPane.showInputDialog(
                    this,
                    "Select Medicine",
                    "Add Medicine",
                    JOptionPane.PLAIN_MESSAGE,
                    null,
                    PharmacyDatabase.inventory.toArray(),
                    null
            );
            if (selected != null) {
                String qtyStr = JOptionPane.showInputDialog(this, "Enter Quantity:");
                if (qtyStr == null) return;
                try {
                    int qty = Integer.parseInt(qtyStr);
                    if (qty > selected.quantity) {
                        JOptionPane.showMessageDialog(this, "Not enough stock!");
                        return;
                    }
                    selected.quantity -= qty;
                    Item cartItem = new Item(selected.name, qty, selected.price,
    selected.manufactureDate, selected.expiryDate);
                    cart.addElement(cartItem);
                    updateTotal();
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(this, "Invalid quantity!");
                }
            }
        });

        // === Remove Medicine ===
        removeBtn.addActionListener(e -> {
            Item selected = cartList.getSelectedValue();
            if (selected != null) {
                for (Item i : PharmacyDatabase.inventory) {
                    if (i.name.equals(selected.name) && i.price == selected.price) {
                        i.quantity += selected.quantity;
                        break;
                    }
                }
                cart.removeElement(selected);
                updateTotal();
            } else {
                JOptionPane.showMessageDialog(this, "Select a medicine to remove!");
            }
        });

        // === Payment + PDF Generation ===
        generateBtn.addActionListener(e -> {
            double subtotal = calculateTotal();
            double gst = subtotal * 0.18;
            double totalPayable = subtotal + gst;

            PaymentWindow paymentWindow = new PaymentWindow(totalPayable, (paidAmount) -
> {
                this.finalAmount = paidAmount;
                generateBillPDF();
            });
            paymentWindow.setVisible(true);
        });

        setVisible(true);
    }

    private void updateTotal() {
        totalLabel.setText("₹" + calculateTotal());
    }

    private double calculateTotal() {
        double total = 0;
        for (int i = 0; i < cart.size(); i++) {
            Item it = cart.get(i);
            total += it.price * it.quantity;
        }
        return total;
    }
    private void generateBillPDF() {
    String customerName = customerNameField.getText();
    String fatherName = fatherNameField.getText();
    String phone = customerPhoneField.getText();
    String age = customerAgeField.getText();
    String address = customerAddressField.getText();

    if (customerName.isEmpty() || fatherName.isEmpty() || phone.isEmpty() || age.isEmpty() ||
address.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Enter all customer details!");
        return;
    }

    try {
        File billsFolder = new File("bills");
        if (!billsFolder.exists()) billsFolder.mkdir();

        Document document = new Document();
        String fileName = "bills/Bill_" + System.currentTimeMillis() + ".pdf";
        PdfWriter.getInstance(document, new FileOutputStream(fileName));
        document.open();

        Font normalFont = new Font(Font.FontFamily.HELVETICA, 12, Font.NORMAL);
        Font boldFont = new Font(Font.FontFamily.HELVETICA, 12, Font.BOLD);
        Font titleFont = new Font(Font.FontFamily.HELVETICA, 16, Font.BOLD);

        Paragraph header = new Paragraph("=== Pharmacy Invoice ===", titleFont);
        header.setAlignment(Element.ALIGN_CENTER);
        document.add(header);
        document.add(new Paragraph("\n"));

        //  === Current Date & Time ===
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
        String dateTime = LocalDateTime.now().format(dtf);
        document.add(new Paragraph("Date & Time: " + dateTime, boldFont));
        document.add(new Paragraph("\n"));

        // === Customer Info ===
        document.add(new Paragraph("Patient: " + customerName, normalFont));
        document.add(new Paragraph("Father's Name: " + fatherName, normalFont));
        document.add(new Paragraph("Phone: " + phone, normalFont));
        document.add(new Paragraph("Age: " + age, normalFont));
        document.add(new Paragraph("Address: " + address, normalFont));
        document.add(new Paragraph("GST No: " + GST_NUMBER, normalFont));
        document.add(new Paragraph("\n"));

        // === Items ===
        PdfPTable table = new PdfPTable(5);
        table.setWidthPercentage(100);
        table.setWidths(new int[]{2, 1, 1, 2, 2});

        table.addCell(new PdfPCell(new Phrase("Medicine", boldFont)));
        table.addCell(new PdfPCell(new Phrase("Qty", boldFont)));
        table.addCell(new PdfPCell(new Phrase("Price", boldFont)));
        table.addCell(new PdfPCell(new Phrase("Mfg Date", boldFont)));
        table.addCell(new PdfPCell(new Phrase("Expiry Date", boldFont)));

        double subtotal = 0;
        for (int i = 0; i < cart.size(); i++) {
            Item it = cart.get(i);
            table.addCell(new Phrase(it.name, normalFont));
            table.addCell(new Phrase(String.valueOf(it.quantity), normalFont));
            table.addCell(new Phrase("₹" + it.price, normalFont));
            table.addCell(new Phrase(it.manufactureDate, normalFont));
            table.addCell(new Phrase(it.expiryDate, normalFont));
            subtotal += it.price * it.quantity;
        }
        document.add(table);

        // === Calculations ===
        double gstAmount = subtotal * 0.18;
        double totalBill = subtotal + gstAmount;
        double paidAmount = finalAmount;  // comes from PaymentWindow
        double exchangeAmount = paidAmount - totalBill;

        document.add(new Paragraph("\nSubtotal: ₹" + subtotal, boldFont));
        document.add(new Paragraph("GST (18%): ₹" + gstAmount, boldFont));
        document.add(new Paragraph("Total Bill: ₹" + totalBill, boldFont));
        document.add(new Paragraph("Customer Paid: ₹" + paidAmount, boldFont));

        if (exchangeAmount >= 0) {
            document.add(new Paragraph("Exchange/Return: ₹" + exchangeAmount, boldFont));
        } else {
            document.add(new Paragraph("Pending Due: ₹" + Math.abs(exchangeAmount),
boldFont));
        }

        // === Footer ===
        PdfPTable footerTable = new PdfPTable(1);
        footerTable.setWidthPercentage(100);
        footerTable.getDefaultCell().setBorder(Rectangle.NO_BORDER);

        footerTable.addCell(new Phrase("Pharmacy Name: HealthCare Medical Store",
normalFont));
        footerTable.addCell(new Phrase("Address: Main Market, Patiala, Punjab", normalFont));
        footerTable.addCell(new Phrase("GST Number: " + GST_NUMBER, normalFont));
        footerTable.addCell(new Phrase("Thank you for your purchase! Visit Again.", boldFont));

        document.add(new Paragraph("\n"));
        document.add(footerTable);

        document.close();
        JOptionPane.showMessageDialog(this, " Bill saved in: " + fileName);

    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, " Error generating PDF: " + e.getMessage());
    }
}
}
```



Billing Window

Patient Name:	Nitin Garg
Father's Name:	Sandeep kumar Garg
Phone:	6280106161
Age:	20
Address:	Barnala

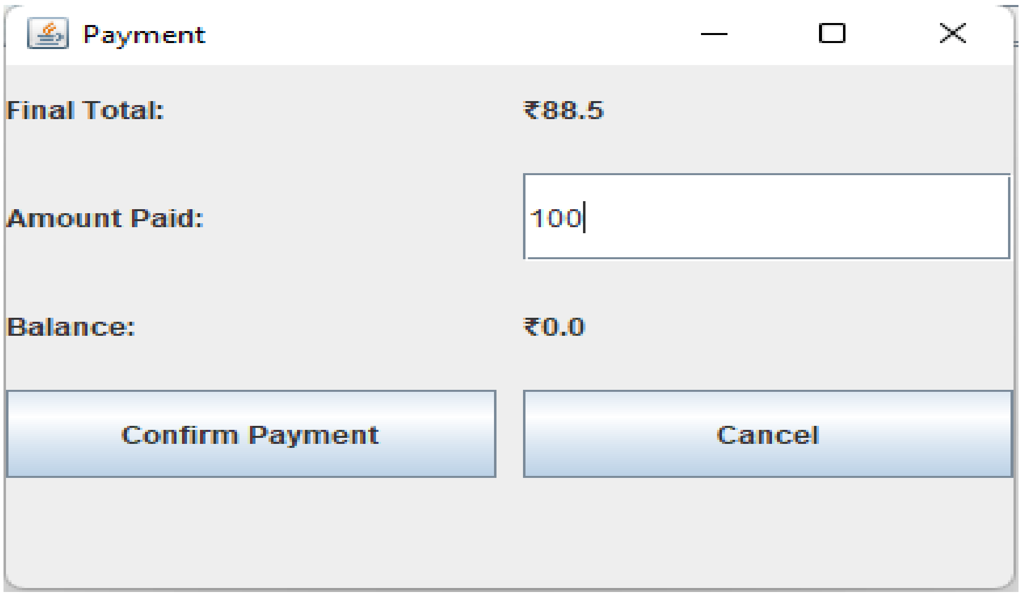
Disprin (₹12.0)

CHAPTER 6

PAYMENT WINDOW

Another key component of the billing process is the PaymentWindow, which manages the financial transaction aspect of the sale. It prompts the user to enter the amount received from the customer and automatically calculates the balance or change to be returned. The payment is currently handled in cash mode for simplicity. Once the payment is confirmed, the system calls back to the BillingWindow, triggering the PDF generation process. This modular interaction between PaymentWindow and BillingWindow exemplifies the use of callback functions (implemented via Java's functional interface DoubleConsumer) to facilitate smooth inter-class communication without tightly coupling the components.

```
import javax.swing.*;
import java.awt.*;
import java.util.function.DoubleConsumer;  // add this

public class PaymentWindow extends JFrame {
    private final double finalAmount;
    private final DoubleConsumer onConfirm;   // callback that receives paidAmount

    private JTextField paidAmountField = new JTextField();
    private JLabel balanceLabel = new JLabel("₹0.0");
    private JLabel totalLabel = new JLabel("₹0.0");

    public PaymentWindow(double finalAmount, DoubleConsumer onConfirm) {
        this.finalAmount = finalAmount;
        this.onConfirm = onConfirm;

        setTitle("Payment");
        setSize(400, 300);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        JPanel panel = new JPanel(new GridLayout(5, 2, 10, 10));

        panel.add(new JLabel("Final Total: "));
        totalLabel.setText("₹" + finalAmount);
        panel.add(totalLabel);

        panel.add(new JLabel("Amount Paid: "));
        panel.add(paidAmountField);

        panel.add(new JLabel("Balance: "));
        panel.add(balanceLabel);

        JButton confirmBtn = new JButton("Confirm Payment");
        JButton cancelBtn = new JButton("Cancel");

        confirmBtn.addActionListener(e -> confirmPayment());
        cancelBtn.addActionListener(e -> dispose());
        panel.add(confirmBtn);
        panel.add(cancelBtn);

        add(panel, BorderLayout.CENTER);
    }

    private void confirmPayment() {
        try {
            double paid = Double.parseDouble(paidAmountField.getText());
            if (paid < finalAmount) {
                JOptionPane.showMessageDialog(this, "Paid amount is less than total bill!", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            double balance = paid - finalAmount;
            balanceLabel.setText("₹" + balance);

            JOptionPane.showMessageDialog(this,
                "Payment Successful!\nBalance/Change: ₹" + balance,   //payment can given in cash
only
                "Success", JOptionPane.INFORMATION_MESSAGE);

            if (onConfirm != null) onConfirm.accept(paid);   // send paidAmount back
            dispose();
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid amount entered!", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
}
```


Payment

Final Total: ₹88.5

Amount Paid: 100

Balance: ₹0.0

Confirm Payment

Cancel

CHAPTER 7

PHARMACY DATABASE

At the foundation of the entire application lies the PharmacyDatabase class, which acts as an in-memory database holding all the medicine information. It uses a static list of Item objects that can be accessed and modified globally by all windows. The design includes a static initialization block that preloads the database with a diverse set of medicines, complete with details such as name, quantity, price, and validity dates. This feature allows the application to start with sample data, improving the testing and demonstration process. Since the database is memory-resident, it ensures rapid access times and seamless data sharing between different modules. In a future enhancement, this could be replaced with an external database such as MySQL or SQLite for persistent data storage, but for a desktop prototype, the static structure suffices effectively.

The Item class serves as the fundamental data model for the entire application. It encapsulates the attributes of each medicine — name, quantity, price, manufacture date, and expiry date — and overrides methods like toString() to present the item details meaningfully in GUI components. Additionally, it implements equality checks to ensure that duplicate items with the same name and price are recognized as the same entity. This class adheres to object-oriented principles of encapsulation and abstraction, simplifying the handling of complex inventory data.

```
import java.util.ArrayList;
import java.util.List;

public class PharmacyDatabase {
    // Shared inventory list accessible by all windows
    public static List<Item> inventory = new ArrayList<>();

    // Static block to preload some medicines
    static {
        inventory.add(new Item("Paracetamol", 50, 20, "01-01-2025", "01-01-2026"));
        inventory.add(new Item("Amoxicillin", 100, 15, "15-03-2024", "15-03-2027"));
        inventory.add(new Item("Cough Syrup", 120, 10, "10-04-2024", "10-04-2027"));
        inventory.add(new Item("Vitamin C", 200, 25, "20-02-2024", "20-02-2026"));
        inventory.add(new Item("Ibuprofen", 80, 18, "05-05-2025", "05-05-2028"));
        inventory.add(new Item("Cetrizine", 30, 30.0 , "25-06-2024", "25-06-2026"));
        inventory.add(new Item("Combiflame", 30, 60.0 , "25-06-2024", "25-06-2027"));
        inventory.add(new Item("Saridon", 30, 40.0 , "25-06-2024", "25-06-2026"));
        inventory.add(new Item("Disprin", 30, 12.0 , "25-06-2024", "25-06-2026"));
        inventory.add(new Item("Avil50", 30, 35.0 ,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Naproxen", 30, 40.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Vitamin C", 30, 20.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Lopex", 30, 25.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Liv-52", 30, 90.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Wegovy-injection", 30, 50.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Vicks", 30, 25.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Iodex", 30, 45.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Vicks Action 500", 30, 55.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Rabzole-DSR", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Rockcid-DSR", 30, 85.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Mulrab-DSR ", 30, 75.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Pantocid-DSR ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Vitamin E ", 30, 15.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Diclofenac", 30, 90.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Doxycycline ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Atenolol ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Ceftriaxone injection", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Fluoxetine", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Gentamicin eye drops ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Metformin", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Ranitidine ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Simvastatin", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Velax SR", 30, 78.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("NICOTIN", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Rotavirus vaccine ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Limci", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Gabavent-300 ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("becosules", 30, 60.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Mebizid-C ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Ranitidine", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Latanoprost ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Amikacin ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Ivermectin ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Amlodipine ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Levocetirizine ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Prednisolone ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Dexamethasone ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Pregabalin", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Promethazine ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Protonix ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Quetiapine ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Risperdal ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Ritalin ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Sudafed ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Sulfamethoxazo ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Crocin ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("COZYMIN ", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Mahamox-CV 625", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("I-pill", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Norethisterone", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Vigra", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("ZENFLOX-UTI", 30, 95.0,"01-01-2025" ,"01-01-2028"));
```

```
        inventory.add(new Item("Dettol Antiseptic", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Otrivin Oxy Fast Relief", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Saridon Advance", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Dabur Shilajit Gold", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("BOROLINE", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Ashwagandha", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Revital Women", 30, 95.0,"01-01-2025" ,"01-01-2028"));
        inventory.add(new Item("Hidroeyes-Eye drops", 50, 365.0,"01-01-2026" ,"01-01-2028"));
        inventory.add(new Item("Refresh Tears-Eye drops", 50, 252.0,"01-01-2026" ,"01-01-
2028"));
        inventory.add(new Item("BePos-Eye drops", 50, 360.0,"01-01-2026" ,"01-01-2028"));
        inventory.add(new Item("LotelGel-Eye drops", 50, 429.0,"01-01-2026" ,"01-01-2028"));

        // Add more items as needed
    } }
```

CHAPTER 8

BACKGROUND PANEL

The visual aspect of the application is enhanced by the BackgroundPanel class, which manages background images across different windows. This class extends JPanel and overrides the paintComponent() method to draw images scaled to fit the window size. The use of soft overlays, transparency effects, and color harmony in the design creates a pleasant visual experience for the user. Such attention to the graphical design is not just aesthetic but also improves usability by guiding user focus and ensuring a professional presentation.

```
// File: BackgroundPanel.java
import javax.swing.*;
import java.awt.*;

public class BackgroundPanel extends JPanel {
    private Image backgroundImage;

    public BackgroundPanel(String imagePath) {
        backgroundImage = new ImageIcon(imagePath).getImage();
        setLayout(new BorderLayout());
    }

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
    }
}
```

CHAPTER 9

DESIGN OF PHARMA HUB

Overall, the user interface design of Pharma Hub emphasizes clarity, simplicity, and consistency. Each window follows a coherent theme using similar fonts, color schemes, and layouts. The fonts like “Times New Roman” and “Arial” are carefully chosen to convey readability and a professional look. Buttons are styled consistently with blue or black tones, maintaining uniformity across all modules. The layout follows grid-based alignment principles, avoiding clutter and ensuring that each input field and control element has sufficient spacing. The result is a system that is both visually appealing and functionally intuitive.From a data flow perspective, the design is logical and streamlined. The system begins with the login stage, where user credentials determine the level of access. Once authenticated, the user navigates through the main menu to access inventory, billing, or exit functions. Each operation interacts with the central PharmacyDatabase, ensuring synchronized data across all modules. When a sale is made, the billing module deducts quantities from the inventory, processes payment, and records the transaction as a PDF file. The consistent data flow between modules minimizes redundancy and enhances overall system integrity.In terms of security and validation, the design incorporates multiple layers of protection. The login mechanism ensures that only authorized personnel can access sensitive operations such as adding or editing items. Input validation in all fields prevents the entry of invalid or incomplete data, reducing runtime errors. Although the current prototype does not include encryption or external database authentication, the modular design allows these features to be easily integrated in future versions.From a software engineering perspective, the Pharma Hub project adheres to core design principles such as modularity, abstraction, and reusability. Each functional unit is implemented as an independent class with well-defined interfaces. This design structure makes debugging and maintenance easier, as issues can be isolated to specific modules. The system also demonstrates scalability, meaning additional modules—such as supplier management, medicine expiry alerts, or sales reports—can be integrated without restructuring the existing code.The project also emphasizes usability and accessibility. The GUI is intuitive enough for non-technical users, such as pharmacists and cashiers, to operate without extensive training. The use of clear prompts, feedback messages, and structured layouts ensures that users can quickly perform tasks.

CHAPTER 10
TECHNOLOGIES USED

The Pharmacy Management System is a comprehensive, Java-based desktop application developed to manage day-to-day operations of a medical store or pharmacy. The system integrates multiple technologies that work together to provide a complete solution for inventory management, billing, user authentication, and PDF bill generation. Each technology has been selected to fulfill a specific role in the overall architecture of the system.

1. Java Programming Language

The core technology used in this project is Java, which is an object-oriented programming language known for its robustness, portability, and security. All modules such as LoginWindow, InventoryWindow, BillingWindow, PaymentWindow, AddItemWindow, and MainMenuWindow are written in Java. The use of Java allows the application to run on any system that has a Java Virtual Machine (JVM), making the project platform-independent.The project follows the Object-Oriented Programming (OOP) principles of Java such as classes, objects, encapsulation, and abstraction. For example, the Item class encapsulates the details of a medicine including name, price, quantity, manufacture date, and expiry date. The use of classes such as PharmacyDatabase demonstrates abstraction by modeling the inventory as a centralized data source. These OOP concepts help in organizing code efficiently, increasing maintainability and scalability.

2. Java Swing (GUI Framework)

For the graphical user interface, the project uses Java Swing, a part of the Java Foundation Classes (JFC). Swing provides lightweight and platform-independent GUI components such as JFrame, JPanel, JButton, JLabel, JTextField, JList, JScrollPane, and JOptionPane.Each window of the application, such as the login screen, inventory screen, billing window, and payment window, is created using Swing components. Layout managers like GridLayout, BorderLayout, FlowLayout, and GridBagLayout are used to organize components neatly within each window. These layouts make the user interface well-structured and user-friendly.

The project also implements custom background images using a BackgroundPanel class. This class extends JPanel and overrides the paintComponent() method to draw an image as the background for various windows. This adds a professional and attractive appearance to the application.

3. Event Handling using ActionListener

The project uses event-driven programming through ActionListener and DocumentListener. Buttons such as *Login*, *Add Item*, *Delete*, *Proceed to Payment*, and *Confirm Payment* are linked with ActionListener events to perform specific tasks when clicked.

For example:

- The Login button verifies user credentials.
- The Add Medicine button adds selected medicines to the cart.
- The Confirm Payment button calculates balance and finalizes the sale.
- The search field uses DocumentListener to show real-time filtering of medicines.

This technique allows the system to respond to user actions dynamically, improving interactivity and usability.

4. iText PDF Library

The system uses the iText PDF library to generate professional invoices in PDF format. In the BillingWindow class, the program imports the iText API to create formatted PDF documents that include:

- Pharmacy name and GST number
- Customer details
- Medicine list (with quantity, price, manufacture date, expiry date)
- Subtotal, GST amount, total bill
- Paid amount and balance/return

The generated PDF is saved automatically in a “bills” folder with a time-stamped filename. This feature allows the pharmacy to maintain permanent digital billing records, which is useful for accounting and legal purposes.

5. In-Memory Database Using Java Collections

Instead of using an external database such as MySQL or SQLite, this project uses an in-memory database structure implemented with Java’s ArrayList collection. The PharmacyDatabase class contains a static List<Item> which stores all medicine records.This list is shared among all windows of the application. When a medicine is sold, the stock is reduced. When an item is removed from the cart, the stock is restored. This demonstrates effective use of Java Collections Framework (List, ArrayList) and static data sharing to manage real-time inventory.Although this is not a permanent database, it works efficiently for simulation and academic project purposes.

6. File Handling and System IO

The project also involves file input/output operations when generating invoices. The File and FileOutputStream classes are used to create folders and save the generated bills in PDF format. This demonstrates knowledge of Java’s file-handling capabilities and interaction with the operating system’s file system.

7. Anonymous Functions and Lambda Expressions

In certain parts of the program, modern Java features such as lambda expressions and functional interfaces (like DoubleConsumer) are used. For example, the PaymentWindow returns the paid amount back to the BillingWindow through a callback. This makes the code more concise and demonstrates the use of modern Java programming practices.

8. Development Environment

This application is designed to be run inside modern Java IDEs such as Visual Studio Code, IntelliJ IDEA, or Eclipse. These IDEs provide support for compiling, debugging, GUI preview, and project structuring, making development more efficient.

CHAPTER 11
SYSTEM ADVANTAGES

1. **User-Friendly Interface**
   The system offers a clean, graphical user interface developed using Java Swing. All functions such as login, billing, inventory management, and item addition are accessible through clearly labeled buttons and organized windows, making it easy for users with basic computer knowledge to operate.
2. **Fast and Accurate Billing**
   The automated billing module instantly calculates the total amount, GST (18%), and final payable amount. This eliminates manual calculation errors and speeds up the billing process. The balance or change is also calculated automatically in the Payment window.
3. **Efficient Inventory Management**
   The application maintains a central inventory list using Java’s ArrayList. Stock levels are automatically updated when medicines are added to or removed from the cart. This ensures real-time tracking of medicine availability and prevents over-selling.
4. **Role-Based Access Control**
   The login system separates responsibilities of *Admin* and *Pharmacist*. Only Admin users can add new medicines, which increases system security and prevents unauthorized changes to the inventory.
5. **PDF Bill Generation**
   One of the key advantages is the automatic generation of professional PDF invoices using the iText library. Each bill contains customer details, medicine list, GST number, date & time, and total amount. This makes record-keeping more professional and reliable.
6. **Centralized Data Handling**
   All modules share the same inventory through the PharmacyDatabase class. This centralized data handling keeps information consistent throughout the system, whether accessed from the Inventory Window or the Billing Window.
7. **Cost-Effective Solution**
   Since the system uses open-source technologies such as Java and iText, it does not require expensive licenses or paid software, making it suitable for small pharmacies and educational institutions.

8. **Modular Design**
   The system is divided into independent modules such as Login, Inventory Management, Billing, and Payment. This modular structure makes the application easier to understand, maintain, and extend in the future.
9. **Offline Functionality**
   The application works without an internet connection. All operations such as billing and inventory management are performed locally, which is highly beneficial in areas with limited internet access.
10. **Expandable Architecture**
    The existing implementation can be easily upgraded by integrating a database (MySQL/MongoDB), barcode scanners, or online ordering features in the future, without changing the core structure.

CHAPTER 12
SYSTEM LIMITATIONS

1. **No Permanent Database Storage**
   The current system stores data in memory using an ArrayList, which means that all inventory and billing data will be lost when the application is closed. A real database such as MySQL is not yet integrated.
2. **Limited Security Features**
   Although the system has a login window, it uses hard-coded usernames and passwords. There is no password encryption, user registration, or advanced security mechanisms, which can be a risk in real-world implementation.
3. **Single-User Environment**
   The system is designed as a standalone desktop application, which means it supports only one user at a time. It cannot be used by multiple users simultaneously over a network.
4. **Limited Reporting Features**
   The system only generates billing reports in PDF format. It does not include advanced analytical reports such as daily sales summary, monthly profit graphs, or inventory reports.
5. **No Barcode Support**
   Modern pharmacy systems commonly use barcode scanning for quick item selection, but this application currently selects medicines manually from a list, which can be slower for large pharmacies.
6. **Basic User Interface Styling**
   Although background images are used, the user interface is still basic compared to modern web or mobile applications. It can be further enhanced using JavaFX or web technologies.
7. **No Expiry Alert System**
   Even though the expiry date is stored, the system does not automatically warn the user about near-expiry or expired medicines.
8. **No Backup and Recovery**
   There is no automatic backup system. If the system crashes or the computer fails, all data could be lost permanently.

9. **No Supplier or Purchase Record Management**
   The application does not include modules for managing suppliers, purchase orders, or restocking records.
10. **Limited Scalability**
    The system is ideal for small to medium-sized pharmacies. For large, multi-branch pharmacies, it would require significant upgrades such as server-based architecture and cloud database integration.

CONCLUSION

The Pharmacy Management System successfully achieves its primary goal of providing a reliable, efficient, and user-friendly solution for managing the daily operations of a pharmacy. By automating essential tasks such as inventory maintenance, billing, customer information handling, and payment processing, the system significantly reduces manual workload and minimizes the possibility of human errors in calculations and record keeping.

Developed using Java and the Swing framework, the application demonstrates effective implementation of object-oriented programming concepts and event-driven programming. The system is structured into well-defined modules including Login, Main Dashboard, Inventory Management, Add Item, Billing, and Payment Module, each functioning independently while sharing a common database. This modular approach increases the maintainability and readability of the code, making future upgrades easier.

One of the most important achievements of the system is the automated PDF bill generation feature. The use of the iText library to generate professional invoices adds real-world value to the project and makes it suitable for practical use in small to medium-sized medical stores. The inclusion of GST, customer details, medicine information, and payment balance in the bill reflects a realistic billing process followed in actual pharmacies.

Although the current version of the system uses an in-memory database and has limited security options, it still serves as a strong foundation for a complete, advanced pharmacy management solution. With minor upgrades such as connecting to a permanent database, adding barcode scanning, implementing advanced security protocols, and generating analytical reports, this system can be converted into a fully commercial application.

In conclusion, the project effectively demonstrates how software technologies can be applied to simplify and modernize pharmacy operations. It fulfills academic objectives while also providing a practical, real-life application that can be enhanced further according to future requirements.