

**PROJECT REPORT**

**COURSE NAME: OBJECT ORIENTED PROGRAMMING**

**COURSE CODE: 24SJPBCST304**

SUBMITTED BY:

**ALEENA VT**

SREELAKSHMI P

NEVIN KUNJUMON

NEVIN SHANY

****

**Landmark App is a robust desktop application built in Java SE using Java Swing / AWT to serve as the client framework for the user interface. It follows a 3-Tier Architecture model that separates the presentation, business logic, and database layers to ensure scalability, maintainability, and better organization of the system. The Frontend Layer is developed using Swing and AWT components, designed with standard layout managers such as BorderLayout and GridLayout to create a clean, responsive, and user-friendly interface. Through this interface, users can register, log in, add property listings, and view or manage properties seamlessly. The UI ensures smooth interaction between the user and the system while maintaining data validation for secure operations.The Business Logic Layer is implemented using Object-Oriented Programming (OOP) principles in Java, allowing modular and reusable code. This layer handles the main functionalities of the application such as user authentication, property management, and fraud detection. It ensures that property transactions are processed securely while enforcing application rules and maintaining system integrity. The Database Layer uses MySQL for structured and relational data storage, maintaining records of users, properties, and transactions. Connectivity between the Java application and the MySQL database is established through JDBC (MySQL Connector/J), providing stable database access and enabling secure SQL transaction processing. This guarantees data consistency, integrity, and atomicity during property booking or update operations. An integrated Admin Panel provides the administrator with tools to monitor user activities, detect suspicious or fraudulent listings, and block unreliable users. This ensures that the platform remains a safe and trustworthy environment for both buyers and sellers. The application also employs Git and GitHub for collaborative development and version control, allowing multiple developers to contribute efficiently and maintain a clear update history throughout the project’s lifecycle. Overall, Landmark App demonstrates strong design principles by combining OOP concepts, 3-tier architecture, and secure database connectivity to deliver a reliable and efficient property management system.**

**INDEX**

**1.** [**Introduction………………………………………………………………….** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319785)

[**2. Objectives…………………………………………………………………….** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319786)

[**3. Requirement Specification……………………………………………** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319787)

[**Hardware Requirements………………………………………………….** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319788)

[**Software Requirements……………………………………………………** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319789)

[**4. Design Document………………………………………………………… 5**](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319790)

**a)**[**Class Diagam…………………………………………………………** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319791)

**b) Data base design………………………………………………... -6 -**

[**5. Source Code………………………………………………………………... 6**](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319793)

[**6. User Manual………………………………………………………………. 52**](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319794)

[**A. Customer Flow……………………………………………………** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319795)

[**B. Admin Flow………………………………………………………..** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319796)

[**7. Test Cases…………………………………………………………………..** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319797)

[**8. Conclusion……………………………………………………………… 54**](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319798)

**9.** [**Future Scope…………………………………………………………..** Error! Bookmark not defined.](file:///C:\Users\Aleena%20VT\Downloads\Project_Report_(_Java_)%5b1%5d.docx#_Toc212319799)

**10.Reference……………………………………………………………… -55-**

**1.Introduction**

**Landmark app is an application built in Java using Java Swing and AWT that offers a complete digital solution for property buying, selling, and renting in a secure and user-friendly environment. It is designed to bridge the gap between property owners and buyers while ensuring safety through proper user verification and fraud detection mechanisms. The app allows users to easily upload property details, view verified listings, and communicate directly with interested parties through a smooth and interactive interface. Additionally, the admin panel plays a vital role in monitoring user activities and blocking suspicious accounts, ensuring a trustworthy experience. With its smart search options, easy navigation, and efficient data handling, Landmark stands out as a modern property management system that focuses on both convenience and security.**

**2. Objectives**

**The primary objectives of the Landmark App project are:**

**● Allow users to buy, sell, and rent properties securely through a verified and trustworthy platform.**

**● Display detailed property information including price, location, and owner contact in a user-friendly**

**Interface.**

**● Prevent fraudulent listings by ensuring user authentication and data validation.**

**● Provide an easy and efficient way to manage property listings and user accounts.**

**● Demonstrate core Java concepts such as Object-Oriented Programming, exception handling, and**

**database connectivity using Java Swing or AWT.**

**3. Requirement Specification**

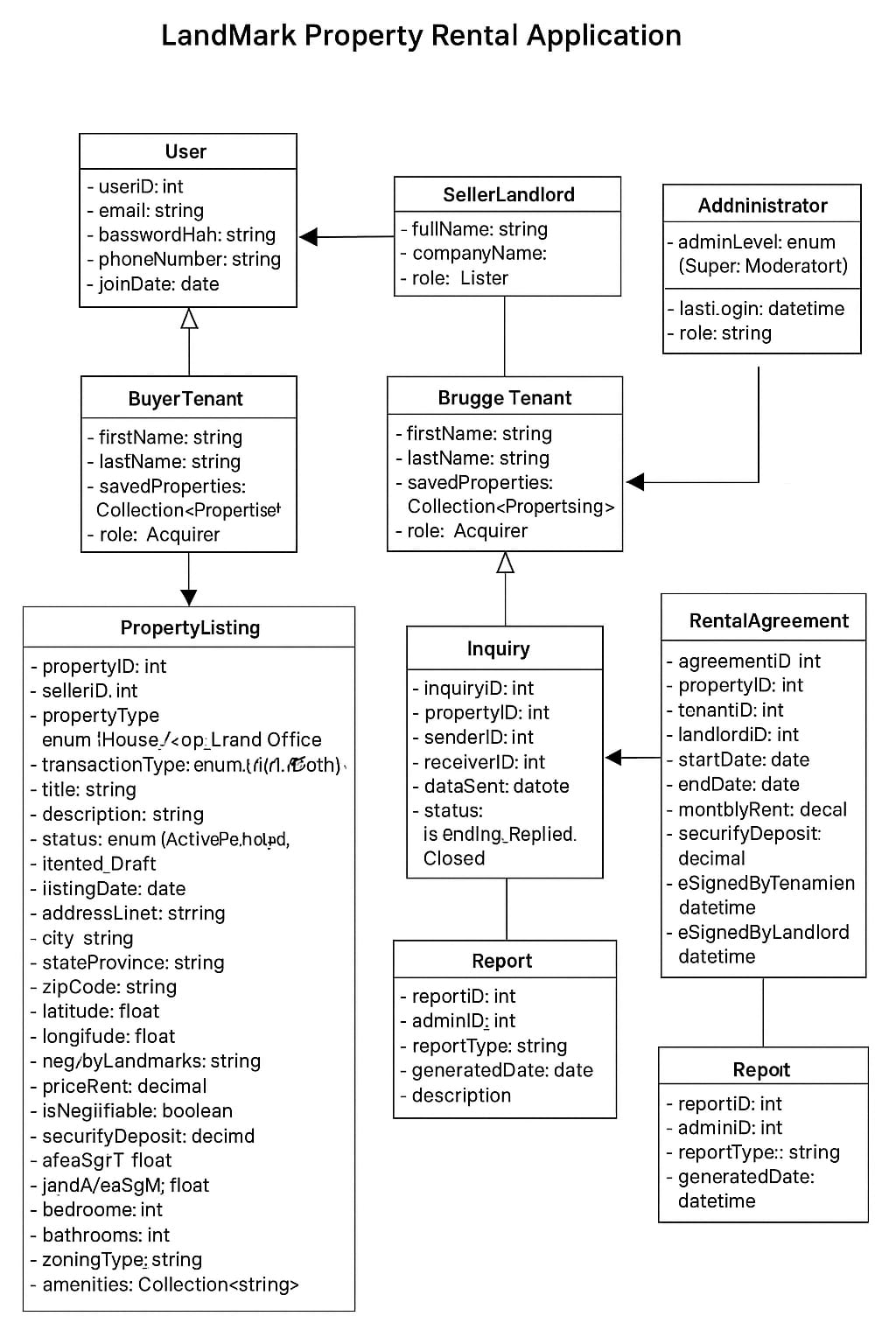
1. Hardware Requirements

* Processor: Intel Core i3 or equivalent (minimum), Intel Core i5 or higher recommended.
* RAM: Minimum 4 GB (8 GB recommended for smooth performance).
* Storage: At least 200 MB of free disk space.
* Display: 1366 × 768 resolution or higher.
* Input Devices: Keyboard and mouse (or touchpad).
* Network: Basic internet connection (for database connectivity or updates).

B.Software requirements

* Operating System: Windows 10 or later / Linux (Ubuntu 20.04 or higher).
* Programming Language: Java SE (Standard Edition) version 8 or above.
* Frontend Framework: Java Swing / AWT for graphical user interface.
* Backend: MySQL Database for storing landmark and user details.
* Database Connectivity: JDBC (Java Database Connectivity).
* Development Environment: NetBeans IDE / Eclipse IDE / IntelliJ IDEA.
* Database Tools: XAMPP / MySQL Workbench.
* Version Control System: Git with GitHub or GitLab for code versioning and collaboration.
* JDK Requirement: Java Development Kit (JDK 8 or above) installed and configured.

**4.Design Document**

**a. Class Diagram**

**b. Database Design**

**The database design of the Landmark App ensures efficient storage, retrieval, and management of user and landmark information. It follows a relational database model using MySQL as the backend, maintaining data integrity and relationships between multiple entities.**

**Entity–Relationship (ER) Overview**

**The major entities in the Landmark App are:**

* **User – Stores user login and registration details.**
* **Landmark – Stores information about various landmarks.**
* **Feedback – Stores feedback and comments from users.**
* **Admin – Manages landmark details and verifies data entries.**

| **Table Name** | **Field Name** | **Data Type** | **Description** |
| --- | --- | --- | --- |
| **User** | **user\_id (PK)** | **INT (Auto Increment)** | **Unique user identifier.** |
|  | **name** | **VARCHAR(100)** | **Name of the user.** |
|  | **email** | **VARCHAR(100)** | **User’s email address (unique).** |
|  | **password** | **VARCHAR(50)** | **Encrypted password for login.** |
|  | **city** | **VARCHAR(50)** | **User’s city.** |

**+----------------+-------------------+-------------------+-------------------------------------------+**

**5. Source Code**

**The source code of the Landmark App is written in Java using Swing and AWT for the frontend, and it integrates with a MySQL database through JDBC for backend operations and data management.**

**5.1 Openingpage.java**

**package com.landmark.ui;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class OpeningPage extends JWindow {**

**private final int displayMillis;**

**private final JProgressBar progressBar;**

**private final JLabel statusLabel;**

**private final JPanel contentPanel;**

**contentPanel = new RoundedPanel(20, new Color(245, 248, 244));**

**contentPanel.setLayout(new BorderLayout(20, 20));**

**contentPanel.setBorder(BorderFactory.createEmptyBorder(24, 24, 24, 24));**

**getContentPane().add(contentPanel);**

**JPanel topPanel = new JPanel(new BorderLayout(12, 0));**

**topPanel.setOpaque(false);**

**ImageIcon rawIcon = loadScaledIcon("/assets/landmark\_logo.png", 96, 96);**

**JLabel logoLabel = new JLabel();**

**if (rawIcon != null) {**

**logoLabel.setIcon(rawIcon);**

**} else {**

**logoLabel.setPreferredSize(new Dimension(96, 96));**

**logoLabel.setHorizontalAlignment(SwingConstants.CENTER);**

**logoLabel.setVerticalAlignment(SwingConstants.CENTER);**

**logoLabel.setOpaque(true);**

**logoLabel.setBackground(new Color(21, 77, 30));**

**logoLabel.setForeground(Color.WHITE);**

**logoLabel.setFont(new Font("Segoe UI", Font.BOLD, 28));**

**logoLabel.setText("LM");**

**logoLabel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));**

**}**

**topPanel.add(logoLabel, BorderLayout.WEST);**

**JPanel titleBlock = new JPanel();**

**titleBlock.setOpaque(false);**

**titleBlock.setLayout(new BoxLayout(titleBlock, BoxLayout.Y\_AXIS));**

**JLabel title = new JLabel("LandMark");**

**title.setFont(new Font("Segoe UI", Font.BOLD, 32));**

**title.setForeground(new Color(21, 77, 30));**

**JLabel tagline = new JLabel("Find. Sell. Settle — your property partner");**

**tagline.setFont(new Font("Segoe UI", Font.PLAIN, 14));**

**tagline.setForeground(new Color(80, 80, 80));**

**titleBlock.add(title);**

**titleBlock.add(Box.createVerticalStrut(6));**

**titleBlock.add(tagline);**

**topPanel.add(titleBlock, BorderLayout.CENTER);**

**contentPanel.add(topPanel, BorderLayout.NORTH);**

**// Center: friendly message or spinner area**

**JPanel center = new JPanel(new BorderLayout());**

**center.setOpaque(false);**

**JLabel welcome = new JLabel("<html><div style='text-align:center;'>Welcome to LandMark — loading your personalised experience</div></html>", SwingConstants.CENTER);**

**welcome.setFont(new Font("Segoe UI", Font.PLAIN, 13));**

**welcome.setForeground(new Color(75, 75, 75));**

**center.add(welcome, BorderLayout.CENTER);**

**contentPanel.add(center, BorderLayout.CENTER);**

**JPanel bottom = new JPanel(new BorderLayout(10, 10));**

**bottom.setOpaque(false);**

**progressBar = new JProgressBar(0, 100);**

**progressBar.setStringPainted(false);**

**progressBar.setPreferredSize(new Dimension(540, 16));**

**progressBar.setBorder(BorderFactory.createEmptyBorder(6, 6, 6, 6));**

**progressBar.setBackground(new Color(230, 230, 230));**

**progressBar.setForeground(new Color(21, 77, 30));**

**bottom.add(progressBar, BorderLayout.CENTER);**

**statusLabel = new JLabel("Initializing modules...");**

**statusLabel.setFont(new Font("Segoe UI", Font.PLAIN, 11));**

**statusLabel.setForeground(new Color(100, 100, 100));**

**bottom.add(statusLabel, BorderLayout.SOUTH);**

**contentPanel.add(bottom, BorderLayout.SOUTH);**

**setBackground(new Color(0,0,0,0));**

**setFocusableWindowState(true);**

**}**

**public void showSplash() {**

**setVisible(true);**

**int steps = 100;**

**int delay = Math.max(10, displayMillis / steps);**

**Timer timer = new Timer(delay, null);**

**timer.addActionListener(new ActionListener() {**

**private int value = 0;**

**private final int stepInc = 1;**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**value += stepInc;**

**progressBar.setValue(value);**

**if (value < 25) statusLabel.setText("Loading UI components...");**

**else if (value < 50) statusLabel.setText("Connecting services...");**

**else if (value < 75) statusLabel.setText("Preparing data...");**

**else statusLabel.setText("Almost ready...");**

**if (value >= 100) {**

**timer.stop();**

**SwingUtilities.invokeLater(() -> {**

**statusLabel.setText("Ready");**

**});**

**new Timer(300, ev -> {**

**((Timer) ev.getSource()).stop();**

**openNext();**

**}).start();**

**}**

**}**

**});**

**timer.start();**

**}**

**private void openNext() {**

**try {**

**SwingUtilities.invokeLater(() -> {**

**setVisible(false);**

**dispose();**

**try {**

**new HomePage(); // <-- if your HomePage requires args, change this later**

**} catch (Throwable t)**

**JFrame fallback = new JFrame("LandMark - Home (Fallback)");**

**fallback.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**fallback.setSize(800, 600);**

**fallback.setLocationRelativeTo(null);**

**JLabel lbl = new JLabel("<html><div style='text-align:center;padding:20px;'>HomePage not implemented yet.<br/>Proceed to implement com.landmark.ui.HomePage</div></html>", SwingConstants.CENTER);**

**lbl.setFont(new Font("Segoe UI", Font.PLAIN, 14));**

**fallback.add(lbl, BorderLayout.CENTER);**

**fallback.setVisible(true);**

**}**

**});**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**}**

**}**

**private ImageIcon loadScaledIcon(String resourcePath, int w, int h) {**

**try {**

**java.net.URL imgUrl = getClass().getResource(resourcePath);**

**if (imgUrl == null) return null;**

**Image img = Toolkit.getDefaultToolkit().getImage(imgUrl);**

**Image scaled = img.getScaledInstance(w, h, Image.SCALE\_SMOOTH);**

**return new ImageIcon(scaled);**

**} catch (Exception ex) {**

**return null;**

**}**

**}**

**private static class RoundedPanel extends JPanel {**

**private final int cornerRadius;**

**private final Color backgroundColor;**

**public RoundedPanel(int cornerRadius, Color backgroundColor) {**

**super();**

**this.cornerRadius = cornerRadius;**

**this.backgroundColor = backgroundColor;**

**setOpaque(false);**

**}**

**@Override**

**protected void paintComponent(Graphics g) {**

**Graphics2D g2 = (Graphics2D) g.create();**

**g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);**

**g2.setColor(new Color(0, 0, 0, 20));**

**g2.fillRoundRect(4, 6, getWidth() - 8, getHeight() - 8, cornerRadius, cornerRadius);**

**g2.setColor(backgroundColor);**

**g2.fillRoundRect(0, 0, getWidth() - 8, getHeight() - 10, cornerRadius, cornerRadius);**

**g2.dispose();**

**super.paintComponent(g);**

**}**

**@Override**

**public Dimension getPreferredSize() {**

**return new Dimension(640, 360);**

**}**

**}**

**// Quick manual tester when running this class directly**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> {**

**OpeningPage splash = new OpeningPage(1600);**

**splash.showSplash();**

**});**

**}**

**}**

**5.2 homepage.java**

**package com.landmark.ui;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**public class HomePage extends JFrame {**

**private final Color themeGreen = new Color(21, 77, 30);**

**private final Color lightGreen = new Color(210, 240, 210);**

**private final Color textColor = new Color(30, 30, 30);**

**public HomePage() {**

**setTitle("LandMark - Home");**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setSize(1000, 600);**

**setLocationRelativeTo(null);**

**setLayout(new BorderLayout());**

**JPanel sidebar = new JPanel();**

**sidebar.setBackground(themeGreen);**

**sidebar.setLayout(new GridLayout(10, 1, 5, 5));**

**sidebar.setBorder(BorderFactory.createEmptyBorder(20, 10, 20, 10));**

**JLabel logo = new JLabel("🏠 LandMark", SwingConstants.CENTER);**

**logo.setForeground(Color.WHITE);**

**logo.setFont(new Font("Segoe UI", Font.BOLD, 22));**

**sidebar.add(logo);**

**sidebar.add(Box.createVerticalStrut(10));**

**JButton purchaseBtn = createMenuButton("Purchase Property");**

**JButton sellBtn = createMenuButton("Sell Property");**

**JButton cartBtn = createMenuButton("My Cart");**

**JButton adsBtn = createMenuButton("My Advertisements");**

**JButton profileBtn = createMenuButton("Profile");**

**JButton aboutBtn = createMenuButton("About");**

**JButton contactBtn = createMenuButton("Contact");**

**JButton logoutBtn = createMenuButton("Logout");**

**sidebar.add(purchaseBtn);**

**sidebar.add(sellBtn);**

**sidebar.add(cartBtn);**

**sidebar.add(adsBtn);**

**sidebar.add(profileBtn);**

**sidebar.add(aboutBtn);**

**sidebar.add(contactBtn);**

**sidebar.add(Box.createVerticalGlue());**

**sidebar.add(logoutBtn);**

**add(sidebar, BorderLayout.WEST);**

**JPanel mainPanel = new JPanel();**

**mainPanel.setBackground(Color.WHITE);**

**mainPanel.setLayout(new BorderLayout());**

**JLabel welcome = new JLabel("<html><center><h1 style='color:#154d1e;'>Welcome to LandMark</h1>"**

**+ "<p style='font-size:13pt;'>Your trusted property marketplace</p></center></html>", SwingConstants.CENTER);**

**mainPanel.add(welcome, BorderLayout.CENTER);**

**add(mainPanel, BorderLayout.CENTER);**

**purchaseBtn.addActionListener(e -> openSection("Purchase"));**

**sellBtn.addActionListener(e -> openSection("Sell"));**

**cartBtn.addActionListener(e -> openSection("MyCart"));**

**adsBtn.addActionListener(e -> openSection("MyAdvertisement"));**

**profileBtn.addActionListener(e -> openSection("Profile"));**

**aboutBtn.addActionListener(e -> JOptionPane.showMessageDialog(this,**

**"LandMark helps users buy and sell properties easily.\nCreated as part of sustainable campus project.",**

**"About LandMark", JOptionPane.INFORMATION\_MESSAGE));**

**contactBtn.addActionListener(e -> JOptionPane.showMessageDialog(this,**

**"Contact us:\nEmail: support@landmarkapp.com\nPhone: +91 98765 43210",**

**"Contact", JOptionPane.INFORMATION\_MESSAGE));**

**logoutBtn.addActionListener(e -> {**

**int choice = JOptionPane.showConfirmDialog(this, "Are you sure you want to logout?",**

**"Confirm Logout", JOptionPane.YES\_NO\_OPTION);**

**if (choice == JOptionPane.YES\_OPTION) {**

**dispose();**

**new OpeningPage(1500).showAndLaunch();**

**}**

**});**

**setVisible(true);**

**}**

**private JButton createMenuButton(String text) {**

**JButton btn = new JButton(text);**

**btn.setFont(new Font("Segoe UI", Font.BOLD, 13));**

**btn.setBackground(themeGreen.darker());**

**btn.setForeground(Color.WHITE);**

**btn.setFocusPainted(false);**

**btn.setBorder(BorderFactory.createEmptyBorder(8, 15, 8, 15));**

**btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));**

**btn.addMouseListener(new java.awt.event.MouseAdapter() {**

**public void mouseEntered(java.awt.event.MouseEvent evt) {**

**btn.setBackground(themeGreen.brighter());**

**}**

**public void mouseExited(java.awt.event.MouseEvent evt) {**

**btn.setBackground(themeGreen.darker());**

**}**

**});**

**return btn;**

**}**

**private void openSection(String sectionName) {**

**try {**

**switch (sectionName) {**

**case "Purchase":**

**new Purchase();**

**break;**

**case "Sell":**

**new Sell();**

**break;**

**case "MyCart":**

**new MyCart();**

**break;**

**case "Profile":**

**new Profile();**

**break;**

**case "MyAdvertisement":**

**new MyAdvertisement();**

**break;**

**default:**

**JOptionPane.showMessageDialog(this,**

**"Section '" + sectionName + "' is not implemented yet.",**

**"Info", JOptionPane.INFORMATION\_MESSAGE);**

**}**

**} catch (Exception ex) {**

**JOptionPane.showMessageDialog(this, "Error opening " + sectionName + ": " + ex.getMessage());**

**ex.printStackTrace();**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(HomePage::new);**

**}**

**}**

**5.3 DatabaseConnection.java**

**package com.landmark.service;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.SQLException;**

**public class DatabaseConnection {**

**private static final String URL = "jdbc:mysql://localhost:3306/landmark\_db";**

**private static final String USER = "root";**

**private static final String PASSWORD = "aleena";**

**static {**

**try {**

**Class.forName("com.mysql.cj.jdbc.Driver");**

**} catch (ClassNotFoundException e) {**

**System.err.println("❌ MySQL JDBC Driver not found: " + e.getMessage());**

**}**

**}**

**public static Connection getConnection() {**

**try {**

**return DriverManager.getConnection(URL, USER, PASSWORD);**

**} catch (SQLException e) {**

**System.err.println("❌ Database connection failed: " + e.getMessage());**

**return null;**

**}**

**}**

**}**

**5.4 Purchase.java**

**package com.landmark.ui;**

**import com.landmark.service.DatabaseConnection;**

**import javax.swing.\*;**

**import javax.swing.table.DefaultTableModel;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.sql.\*;**

**public class Purchase extends JFrame {**

**private final Color themeGreen = new Color(21, 77, 30);**

**private final Color lightGreen = new Color(220, 240, 220);**

**private final JTextField searchField;**

**private final JTable propertyTable;**

**private final DefaultTableModel model;**

**public Purchase() {**

**setTitle("LandMark - Purchase Property");**

**setSize(900, 550);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLocationRelativeTo(null);**

**setLayout(new BorderLayout(10, 10));**

**JPanel header = new JPanel(new BorderLayout());**

**header.setBackground(themeGreen);**

**JLabel title = new JLabel("🏠 Browse Properties", SwingConstants.CENTER);**

**title.setForeground(Color.WHITE);**

**title.setFont(new Font("Segoe UI", Font.BOLD, 22));**

**header.add(title, BorderLayout.CENTER);**

**add(header, BorderLayout.NORTH);**

**JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10));**

**searchPanel.add(new JLabel("Search by Location: "));**

**searchField = new JTextField(20);**

**JButton searchBtn = new JButton("Search");**

**JButton refreshBtn = new JButton("Show All");**

**styleButton(searchBtn);**

**styleButton(refreshBtn);**

**searchPanel.add(searchField);**

**searchPanel.add(searchBtn);**

**searchPanel.add(refreshBtn);**

**add(searchPanel, BorderLayout.SOUTH);**

**String[] columns = {"Property ID", "Title", "Location", "Price", "Owner Email"};**

**model = new DefaultTableModel(columns, 0);**

**propertyTable = new JTable(model);**

**propertyTable.setRowHeight(25);**

**propertyTable.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD, 13));**

**propertyTable.setSelectionBackground(lightGreen);**

**JScrollPane scrollPane = new JScrollPane(propertyTable);**

**add(scrollPane, BorderLayout.CENTER);**

**JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 10));**

**JButton addCartBtn = new JButton("Add to Cart 🛒");**

**styleButton(addCartBtn);**

**btnPanel.add(addCartBtn);**

**add(btnPanel, BorderLayout.SOUTH);**

**loadProperties("");**

**searchBtn.addActionListener(e -> {**

**String keyword = searchField.getText().trim();**

**loadProperties(keyword);**

**});**

**refreshBtn.addActionListener(e -> {**

**searchField.setText("");**

**loadProperties("");**

**});**

**addCartBtn.addActionListener(e -> addToCart());**

**setVisible(true);**

**}**

**private void styleButton(JButton btn) {**

**btn.setBackground(themeGreen);**

**btn.setForeground(Color.WHITE);**

**btn.setFont(new Font("Segoe UI", Font.BOLD, 13));**

**btn.setFocusPainted(false);**

**btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));**

**}**

**private void loadProperties(String locationFilter) {**

**model.setRowCount(0);**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) {**

**JOptionPane.showMessageDialog(this, "Database not connected!");**

**return;**

**}**

**String sql = "SELECT property\_id, title, location, price, owner\_email FROM properties";**

**if (!locationFilter.isEmpty()) {**

**sql += " WHERE location LIKE ?";**

**}**

**PreparedStatement stmt = conn.prepareStatement(sql);**

**if (!locationFilter.isEmpty()) {**

**stmt.setString(1, "%" + locationFilter + "%");**

**}**

**ResultSet rs = stmt.executeQuery();**

**while (rs.next()) {**

**Object[] row = {**

**rs.getInt("property\_id"),**

**rs.getString("title"),**

**rs.getString("location"),**

**rs.getString("price"),**

**rs.getString("owner\_email")**

**};**

**model.addRow(row);**

**}**

**} catch (SQLException e) {**

**JOptionPane.showMessageDialog(this, "Error loading properties: " + e.getMessage());**

**}**

**}**

**private void addToCart() {**

**int selectedRow = propertyTable.getSelectedRow();**

**if (selectedRow == -1) {**

**JOptionPane.showMessageDialog(this, "Please select a property to add to cart!");**

**return;**

**}**

**int propertyId = (int) model.getValueAt(selectedRow, 0);**

**String title = (String) model.getValueAt(selectedRow, 1);**

**String price = (String) model.getValueAt(selectedRow, 3);**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) {**

**JOptionPane.showMessageDialog(this, "Database not connected!");**

**return;**

**}**

**String insert = "INSERT INTO cart (property\_id, title, price) VALUES (?, ?, ?)";**

**PreparedStatement ps = conn.prepareStatement(insert);**

**ps.setInt(1, propertyId);**

**ps.setString(2, title);**

**ps.setString(3, price);**

**ps.executeUpdate();**

**JOptionPane.showMessageDialog(this, "Property added to cart successfully!");**

**} catch (SQLException e) {**

**JOptionPane.showMessageDialog(this, "Error adding to cart: " + e.getMessage());**

**}**

**}**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(Purchase::new);**

**}**

**}**

**5.5 Sell.java**

**package com.landmark.ui;**

**import com.landmark.model.User;**

**import com.landmark.service.DatabaseConnection;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**public class Sell extends JFrame {**

**private final User owner;**

**private final JTextField titleField;**

**private final JTextField locationField;**

**private final JTextField priceField;**

**private final JTextArea descArea;**

**private final Color themeGreen = new Color(21, 77, 30);**

**public Sell(User owner) {**

**this.owner = owner;**

**setTitle("LandMark - Sell Property");**

**setSize(560, 520);**

**setLocationRelativeTo(null);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLayout(null);**

**JLabel header = new JLabel("List your Property", SwingConstants.CENTER);**

**header.setBounds(20, 10, 500, 50);**

**header.setFont(new Font("Segoe UI", Font.BOLD, 20));**

**header.setForeground(themeGreen);**

**add(header);**

**JLabel lblTitle = new JLabel("Title:");**

**lblTitle.setBounds(30, 80, 100, 25);**

**add(lblTitle);**

**titleField = new JTextField();**

**titleField.setBounds(140, 80, 360, 28);**

**add(titleField);**

**JLabel lblLocation = new JLabel("Location:");**

**lblLocation.setBounds(30, 120, 100, 25);**

**add(lblLocation);**

**locationField = new JTextField();**

**locationField.setBounds(140, 120, 360, 28);**

**add(locationField);**

**JLabel lblPrice = new JLabel("Price (INR):");**

**lblPrice.setBounds(30, 160, 100, 25);**

**add(lblPrice);**

**priceField = new JTextField();**

**priceField.setBounds(140, 160, 200, 28);**

**add(priceField);**

**JLabel lblDesc = new JLabel("Description:");**

**lblDesc.setBounds(30, 200, 100, 25);**

**add(lblDesc);**

**descArea = new JTextArea();**

**JScrollPane jsp = new JScrollPane(descArea);**

**jsp.setBounds(140, 200, 360, 180);**

**add(jsp);**

**JButton saveBtn = new JButton("Save Listing");**

**saveBtn.setBounds(140, 400, 150, 36);**

**styleButton(saveBtn);**

**add(saveBtn);**

**JButton cancelBtn = new JButton("Cancel");**

**cancelBtn.setBounds(320, 400, 120, 36);**

**styleButton(cancelBtn);**

**add(cancelBtn);**

**saveBtn.addActionListener(e -> saveListing());**

**cancelBtn.addActionListener(e -> dispose());**

**setVisible(true);**

**}**

**private void styleButton(JButton b) {**

**b.setBackground(themeGreen);**

**b.setForeground(Color.WHITE);**

**b.setFocusPainted(false);**

**b.setFont(new Font("Segoe UI", Font.BOLD, 12));**

**b.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));**

**}**

**private void saveListing() {**

**String title = titleField.getText().trim();**

**String location = locationField.getText().trim();**

**String priceText = priceField.getText().trim();**

**String description = descArea.getText().trim();**

**if (title.isEmpty() || location.isEmpty() || priceText.isEmpty()) {**

**JOptionPane.showMessageDialog(this, "Please fill Title, Location and Price.", "Validation", JOptionPane.WARNING\_MESSAGE);**

**return;**

**}**

**double price;**

**try {**

**price = Double.parseDouble(priceText);**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(this, "Invalid price value.", "Validation", JOptionPane.WARNING\_MESSAGE);**

**return;**

**}**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) {**

**JOptionPane.showMessageDialog(this, "Database not connected.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String sql = "INSERT INTO properties (title, location, price, description, owner\_email) VALUES (?, ?, ?, ?, ?)";**

**try (PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setString(1, title);**

**ps.setString(2, location);**

**ps.setDouble(3, price);**

**ps.setString(4, description);**

**ps.setString(5, owner.getEmail());**

**int updated = ps.executeUpdate();**

**if (updated == 1) {**

**JOptionPane.showMessageDialog(this, "Property listed successfully!");**

**dispose();**

**} else {**

**JOptionPane.showMessageDialog(this, "Failed to list property.", "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "DB error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**public static void main(String[] args) {**

**User demo = new User(); demo.setEmail("demo@landmark.com"); demo.setName("Demo Seller");**

**SwingUtilities.invokeLater(() -> new Sell(demo));**

**}**

**}**

**5.6** **MyCart.java**

**package com.landmark.ui;**

**import com.landmark.model.User;**

**import com.landmark.service.DatabaseConnection;**

**import javax.swing.\*;**

**import javax.swing.table.DefaultTableModel;**

**import java.awt.\*;**

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class MyCart extends JFrame {**

**private final User currentUser;**

**private final DefaultTableModel model;**

**private final JTable table;**

**private final Color themeGreen = new Color(21, 77, 30);**

**public MyCart(User currentUser) {**

**this.currentUser = currentUser;**

**setTitle("My Cart - " + currentUser.getName());**

**setSize(800, 480);**

**setLocationRelativeTo(null);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLayout(new BorderLayout(8, 8));**

**JLabel header = new JLabel("🛒 My Cart", SwingConstants.CENTER);**

**header.setFont(new Font("Segoe UI", Font.BOLD, 20));**

**header.setForeground(themeGreen);**

**add(header, BorderLayout.NORTH);**

**String[] cols = {"Cart ID", "Property ID", "Title", "Location", "Price", "Added At"};**

**model = new DefaultTableModel(cols, 0);**

**table = new JTable(model);**

**table.setRowHeight(26);**

**add(new JScrollPane(table), BorderLayout.CENTER);**

**JPanel actions = new JPanel(new FlowLayout(FlowLayout.RIGHT));**

**JButton removeBtn = new JButton("Remove Selected");**

**JButton refreshBtn = new JButton("Refresh");**

**styleButton(removeBtn);**

**styleButton(refreshBtn);**

**actions.add(refreshBtn);**

**actions.add(removeBtn);**

**add(actions, BorderLayout.SOUTH);**

**refreshBtn.addActionListener(e -> loadCart());**

**removeBtn.addActionListener(e -> removeSelected());**

**loadCart();**

**setVisible(true);**

**}**

**private void styleButton(JButton b) {**

**b.setBackground(themeGreen);**

**b.setForeground(Color.WHITE);**

**b.setFont(new Font("Segoe UI", Font.BOLD, 12));**

**b.setFocusPainted(false);**

**b.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));**

**}**

**private void loadCart() {**

**model.setRowCount(0);**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) {**

**JOptionPane.showMessageDialog(this, "DB not connected", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String sql = "SELECT c.id AS cart\_id, c.property\_id, p.title, p.location, p.price, c.added\_at " +**

**"FROM cart\_items c JOIN properties p ON c.property\_id = p.id WHERE c.user\_id = ? ORDER BY c.added\_at DESC";**

**try (PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setInt(1, currentUser.getId());**

**try (ResultSet rs = ps.executeQuery()) {**

**while (rs.next()) {**

**Object[] row = {**

**rs.getInt("cart\_id"),**

**rs.getInt("property\_id"),**

**rs.getString("title"),**

**rs.getString("location"),**

**rs.getDouble("price"),**

**rs.getTimestamp("added\_at")**

**};**

**model.addRow(row);**

**}**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "Error loading cart: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**private void removeSelected() {**

**int sel = table.getSelectedRow();**

**if (sel == -1) {**

**JOptionPane.showMessageDialog(this, "Select an item to remove.");**

**return;**

**}**

**int cartId = (int) model.getValueAt(sel, 0);**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**String del = "DELETE FROM cart\_items WHERE id = ?";**

**try (PreparedStatement ps = conn.prepareStatement(del)) {**

**ps.setInt(1, cartId);**

**int r = ps.executeUpdate();**

**if (r == 1) {**

**JOptionPane.showMessageDialog(this, "Removed from cart.");**

**loadCart();**

**} else {**

**JOptionPane.showMessageDialog(this, "Failed to remove.");**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "DB error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**id)**

**public static void main(String[] args) {**

**User u = new User(); u.setId(1); u.setName("Demo Buyer"); u.setEmail("demo@landmark.com");**

**SwingUtilities.invokeLater(() -> new MyCart(u));**

**}**

**}**

**5.7** **Profile.java**

**package com.landmark.ui;**

**import com.landmark.model.User;**

**import com.landmark.service.DatabaseConnection;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**public class Profile extends JFrame {**

**private final User user;**

**private final JTextField nameField;**

**private final JTextField phoneField;**

**private final JPasswordField oldPassField;**

**private final JPasswordField newPassField;**

**private final Color themeGreen = new Color(21, 77, 30);**

**public Profile(User user) {**

**this.user = user;**

**setTitle("Profile - " + user.getName());**

**setSize(520, 420);**

**setLocationRelativeTo(null);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLayout(null);**

**JLabel lblHeader = new JLabel("My Profile", SwingConstants.CENTER);**

**lblHeader.setBounds(20, 10, 480, 40);**

**lblHeader.setFont(new Font("Segoe UI", Font.BOLD, 20));**

**lblHeader.setForeground(themeGreen);**

**add(lblHeader);**

**JLabel lblName = new JLabel("Full Name:"); lblName.setBounds(30, 70, 100, 25); add(lblName);**

**nameField = new JTextField(user.getName()); nameField.setBounds(140, 70, 320, 28); add(nameField);**

**JLabel lblEmail = new JLabel("Email:"); lblEmail.setBounds(30, 110, 100, 25); add(lblEmail);**

**JTextField emailTf = new JTextField(user.getEmail()); emailTf.setBounds(140, 110, 320, 28); emailTf.setEditable(false); add(emailTf);**

**JLabel lblPhone = new JLabel("Phone:"); lblPhone.setBounds(30, 150, 100, 25); add(lblPhone);**

**phoneField = new JTextField(user.getPhone()); phoneField.setBounds(140, 150, 320, 28); add(phoneField);**

**JButton saveBtn = new JButton("Save Profile"); saveBtn.setBounds(140, 190, 150, 36); styleBtn(saveBtn); add(saveBtn);**

**JButton closeBtn = new JButton("Close"); closeBtn.setBounds(320, 190, 140, 36); styleBtn(closeBtn); add(closeBtn);**

**JLabel lblOld = new JLabel("Old password:"); lblOld.setBounds(30, 250, 120, 25); add(lblOld);**

**oldPassField = new JPasswordField(); oldPassField.setBounds(160, 250, 220, 28); add(oldPassField);**

**JLabel lblNew = new JLabel("New password:"); lblNew.setBounds(30, 290, 120, 25); add(lblNew);**

**newPassField = new JPasswordField(); newPassField.setBounds(160, 290, 220, 28); add(newPassField);**

**JButton changePassBtn = new JButton("Change Password"); changePassBtn.setBounds(160, 330, 220, 36); styleBtn(changePassBtn); add(changePassBtn);**

**saveBtn.addActionListener(e -> saveProfile());**

**closeBtn.addActionListener(e -> dispose());**

**changePassBtn.addActionListener(e -> changePassword());**

**setVisible(true);**

**}**

**private void styleBtn(JButton b) {**

**b.setBackground(themeGreen);**

**b.setForeground(Color.WHITE);**

**b.setFont(new Font("Segoe UI", Font.BOLD, 12));**

**b.setFocusPainted(false);**

**}**

**private void saveProfile() {**

**String newName = nameField.getText().trim();**

**String newPhone = phoneField.getText().trim();**

**if (newName.isEmpty()) {**

**JOptionPane.showMessageDialog(this, "Name cannot be empty.", "Validation", JOptionPane.WARNING\_MESSAGE);**

**return;**

**}**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) { JOptionPane.showMessageDialog(this, "DB not connected."); return; }**

**String sql = "UPDATE users SET name = ?, phone = ? WHERE id = ?";**

**try (PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setString(1, newName);**

**ps.setString(2, newPhone);**

**ps.setInt(3, user.getId());**

**int r = ps.executeUpdate();**

**if (r == 1) {**

**JOptionPane.showMessageDialog(this, "Profile updated.");**

**user.setName(newName);**

**user.setPhone(newPhone);**

**} else {**

**JOptionPane.showMessageDialog(this, "No changes saved.");**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "DB error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**private void changePassword() {**

**String oldP = new String(oldPassField.getPassword()), newP = new String(newPassField.getPassword());**

**if (oldP.isEmpty() || newP.length() < 4) {**

**JOptionPane.showMessageDialog(this, "Provide correct old password and new password (>=4 chars).");**

**return;**

**}**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) { JOptionPane.showMessageDialog(this, "DB not connected."); return; }**

**String check = "SELECT id FROM users WHERE id = ? AND password = ?";**

**try (PreparedStatement checkPs = conn.prepareStatement(check)) {**

**checkPs.setInt(1, user.getId());**

**checkPs.setString(2, oldP);**

**try (ResultSet rs = checkPs.executeQuery()) {**

**if (!rs.next()) {**

**JOptionPane.showMessageDialog(this, "Old password incorrect.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**}**

**}**

**String upd = "UPDATE users SET password = ? WHERE id = ?";**

**try (PreparedStatement ps = conn.prepareStatement(upd)) {**

**ps.setString(1, newP);**

**ps.setInt(2, user.getId());**

**int r = ps.executeUpdate();**

**if (r == 1) {**

**JOptionPane.showMessageDialog(this, "Password changed. Please remember it.");**

**oldPassField.setText("");**

**newPassField.setText("");**

**} else {**

**JOptionPane.showMessageDialog(this, "Failed to change password.", "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "DB error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**}**

**5.8** **MyAdvertisement.java**

**package com.landmark.ui;**

**import com.landmark.model.Property;**

**import com.landmark.model.User;**

**import com.landmark.service.DatabaseConnection;**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class MyAdvertisement extends JFrame {**

**private final User seller;**

**private final JPanel listPanel;**

**private final Color themeGreen = new Color(21, 77, 30);**

**public MyAdvertisement(User seller) {**

**this.seller = seller;**

**setTitle("My Advertisements - " + seller.getName());**

**setSize(900, 520);**

**setLocationRelativeTo(null);**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLayout(new BorderLayout(8, 8));**

**JLabel header = new JLabel("My Advertisements", SwingConstants.CENTER);**

**header.setFont(new Font("Segoe UI", Font.BOLD, 20));**

**header.setForeground(themeGreen);**

**add(header, BorderLayout.NORTH);**

**listPanel = new JPanel();**

**listPanel.setLayout(new BoxLayout(listPanel, BoxLayout.Y\_AXIS));**

**add(new JScrollPane(listPanel), BorderLayout.CENTER);**

**JPanel bottom = new JPanel(new FlowLayout(FlowLayout.RIGHT));**

**JButton addNew = new JButton("Add New");**

**JButton refresh = new JButton("Refresh");**

**style(addNew); style(refresh);**

**bottom.add(refresh); bottom.add(addNew);**

**add(bottom, BorderLayout.SOUTH);**

**addNew.addActionListener(e -> new Sell(seller)); // reuse Sell form**

**refresh.addActionListener(e -> loadAds());**

**loadAds();**

**setVisible(true);**

**}**

**private void style(JButton b) {**

**b.setBackground(themeGreen);**

**b.setForeground(Color.WHITE);**

**b.setFont(new Font("Segoe UI", Font.BOLD, 12));**

**b.setFocusPainted(false);**

**}**

**private void loadAds() {**

**listPanel.removeAll();**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**if (conn == null) {**

**JOptionPane.showMessageDialog(this, "DB not connected", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String sql = "SELECT id, title, location, price, description FROM properties WHERE owner\_email = ? ORDER BY created\_at DESC";**

**try (PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setString(1, seller.getEmail());**

**try (ResultSet rs = ps.executeQuery()) {**

**List<Property> list = new ArrayList<>();**

**while (rs.next()) {**

**Property p = new Property();**

**p.setId(rs.getInt("id"));**

**p.setTitle(rs.getString("title"));**

**p.setLocation(rs.getString("location"));**

**p.setPrice(rs.getDouble("price"));**

**p.setDescription(rs.getString("description"));**

**list.add(p);**

**}**

**if (list.isEmpty()) {**

**listPanel.add(new JLabel("No advertisements posted yet."));**

**} else {**

**for (Property p : list) {**

**JPanel card = new JPanel(new BorderLayout());**

**card.setBorder(BorderFactory.createLineBorder(Color.GRAY));**

**JLabel t = new JLabel(p.getTitle() + " — ₹" + p.getPrice());**

**t.setFont(new Font("Segoe UI", Font.BOLD, 14));**

**card.add(t, BorderLayout.CENTER);**

**JPanel actions = new JPanel();**

**JButton view = new JButton("View"); JButton edit = new JButton("Edit"); JButton del = new JButton("Delete");**

**actions.add(view); actions.add(edit); actions.add(del);**

**card.add(actions, BorderLayout.EAST);**

**view.addActionListener(e -> new PropertyDetailsPage((JFrame) this.getOwner(), p));**

**edit.addActionListener(e -> {**

**new PropertyForm(this, seller, p**

**SwingUtilities.invokeLater(this::loadAds);**

**});**

**del.addActionListener(e -> {**

**int ok = JOptionPane.showConfirmDialog(this, "Delete this property?", "Confirm", JOptionPane.YES\_NO\_OPTION);**

**if (ok == JOptionPane.YES\_OPTION) {**

**deleteProperty(p.getId());**

**}**

**});**

**listPanel.add(card);**

**listPanel.add(Box.createVerticalStrut(6));**

**}**

**}**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "Load failed: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**listPanel.revalidate();**

**listPanel.repaint();**

**}**

**private void deleteProperty(int id) {**

**try (Connection conn = DatabaseConnection.getConnection()) {**

**String del = "DELETE FROM properties WHERE id = ?";**

**try (PreparedStatement ps = conn.prepareStatement(del)) {**

**ps.setInt(1, id);**

**int r = ps.executeUpdate();**

**if (r == 1) {**

**JOptionPane.showMessageDialog(this, "Deleted.");**

**loadAds();**

**} else {**

**JOptionPane.showMessageDialog(this, "Delete failed.");**

**}**

**}**

**} catch (SQLException ex) {**

**JOptionPane.showMessageDialog(this, "DB: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);**

**ex.printStackTrace();**

**}**

**}**

**public static void main(String[] args) {**

**User demo = new User(); demo.setEmail("demo@landmark.com"); demo.setName("Demo Seller");**

**SwingUtilities.invokeLater(() -> new MyAdvertisement(demo));**

**}**

**}**

**5.9 About.java**

**package com.landmark.ui;**

**import javax.swing.\*;**

**import java.awt.\*;**

**public class About {**

**public static void show(Component parent) {**

**String html = "<html><h2>LandMark</h2><p>LandMark is a simple property marketplace desktop application.</p>"**

**+ "<p>Features include: list properties, search, cart, seller dashboard, and profile management.</p>"**

**+ "<p>Created as a conversion from a web app to Java Swing.</p></html>";**

**JOptionPane.showMessageDialog(parent, new JLabel(html), "About LandMark", JOptionPane.INFORMATION\_MESSAGE);**

**}**

**5.10** **Contact.java**

**package com.landmark.ui;**

**import javax.swing.\*;**

**import java.awt.\*;**

**public class Contact {**

**public static void show(Component parent) {**

**String html = "<html><h3>Contact Us</h3>"**

**+ "<p>Email: support@landmarkapp.com</p>"**

**+ "<p>Phone: +91 98765 43210</p>"**

**+ "<p>Address: Kerala, India</p></html>";**

**JOptionPane.showMessageDialog(parent, new JLabel(html), "Contact", JOptionPane.INFORMATION\_MESSAGE);**

**}**

**}**

**5.11 User.java**

**package com.landmark.model;**

**public class User {**

**private int id;**

**private String name;**

**private String email;**

**private String password;**

**private String phone;**

**private String role;**

**public User() {}**

**public User(int id, String name, String email, String password, String phone, String role) {**

**this.id = id;**

**this.name = name;**

**this.email = email;**

**this.password = password;**

**this.phone = phone;**

**this.role = role;**

**}**

**public int getId() { return id; }**

**public void setId(int id) { this.id = id; }**

**public String getName() { return name; }**

**public void setName(String name) { this.name = name; }**

**public String getEmail() { return email; }**

**public void setEmail(String email) { this.email = email; }**

**public String getPassword() { return password; }**

**public void setPassword(String password) { this.password = password; }**

**public String getPhone() { return phone; }**

**public void setPhone(String phone) { this.phone = phone; }**

**public String getRole() { return role; }**

**public void setRole(String role) { this.role = role; }**

**@Override**

**public String toString() {**

**return "User{" + "id=" + id + ", name='" + name + '\'' + ", email='" + email + '\'' + ", role='" + role + '\'' + '}';**

**}**

**}**

**5.12 CartService.java**

**package com.landmark.service;**

**import com.landmark.model.User;**

**import java.sql.\*;**

**public class UserService {**

**public User authenticate(String email, String password) throws SQLException {**

**String sql = "SELECT id, name, email, password, phone, role FROM users WHERE email = ? AND password = ?";**

**try (Connection conn = DatabaseConnection.getConnection();**

**PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setString(1, email);**

**ps.setString(2, password);**

**try (ResultSet rs = ps.executeQuery()) {**

**if (rs.next()) {**

**User u = new User();**

**u.setId(rs.getInt("id"));**

**u.setName(rs.getString("name"));**

**u.setEmail(rs.getString("email"));**

**u.setPassword(rs.getString("password"));**

**u.setPhone(rs.getString("phone"));**

**u.setRole(rs.getString("role"));**

**return u;**

**}**

**}**

**}**

**return null;**

**}**

**public boolean register(User user) throws SQLException {**

**String sql = "INSERT INTO users (name, email, password, phone, role) VALUES (?, ?, ?, ?, ?)";**

**try (Connection conn = DatabaseConnection.getConnection();**

**PreparedStatement ps = conn.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS)) {**

**ps.setString(1, user.getName());**

**ps.setString(2, user.getEmail());**

**ps.setString(3, user.getPassword());**

**ps.setString(4, user.getPhone());**

**ps.setString(5, user.getRole() == null ? "buyer" : user.getRole());**

**int affected = ps.executeUpdate();**

**if (affected == 1) {**

**try (ResultSet keys = ps.getGeneratedKeys()) {**

**if (keys.next()) user.setId(keys.getInt(1));**

**}**

**return true;**

**}**

**}**

**return false;**

**}**

**public boolean updateProfile(User user) throws SQLException {**

**String sql = "UPDATE users SET name = ?, phone = ? WHERE id = ?";**

**try (Connection conn = DatabaseConnection.getConnection();**

**PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setString(1, user.getName());**

**ps.setString(2, user.getPhone());**

**ps.setInt(3, user.getId());**

**return ps.executeUpdate() == 1;**

**}**

**}**

**public boolean changePassword(int userId, String oldPassword, String newPassword) throws SQLException {**

**String checkSql = "SELECT id FROM users WHERE id = ? AND password = ?";**

**try (Connection conn = DatabaseConnection.getConnection();**

**PreparedStatement check = conn.prepareStatement(checkSql)) {**

**check.setInt(1, userId);**

**check.setString(2, oldPassword);**

**try (ResultSet rs = check.executeQuery()) {**

**if (!rs.next()) return false; // old password mismatch**

**}**

**}**

**String updateSql = "UPDATE users SET password = ? WHERE id = ?";**

**try (Connection conn = DatabaseConnection.getConnection();**

**PreparedStatement ps = conn.prepareStatement(updateSql)) {**

**ps.setString(1, newPassword);**

**ps.setInt(2, userId);**

**return ps.executeUpdate() == 1;**

**}**

**}**

**public User findById(int id) throws SQLException {**

**String sql = "SELECT id, name, email, password, phone, role FROM users WHERE id = ?";**

**try (Connection conn = DatabaseConnection.getConnection();**

**PreparedStatement ps = conn.prepareStatement(sql)) {**

**ps.setInt(1, id);**

**try (ResultSet rs = ps.executeQuery()) {**

**if (rs.next()) {**

**User u = new User();**

**u.setId(rs.getInt("id"));**

**u.setName(rs.getString("name"));**

**u.setEmail(rs.getString("email"));**

**u.setPassword(rs.getString("password"));**

**u.setPhone(rs.getString("phone"));**

**u.setRole(rs.getString("role"));**

**return u;**

**}**

**}**

**}**

**return null;**

**}**

**}**

**5.** **13** **PropertyDetailsPage.java**

**package com.landmark.ui;**

**import com.landmark.model.Property;**

**import javax.swing.\*;**

**import java.awt.\*;**

**public class PropertyDetailsPage extends JDialog {**

**public PropertyDetailsPage(Frame owner, Property p) {**

**super(owner, "Property Details", true);**

**setSize(600, 420);**

**setLocationRelativeTo(owner);**

**setLayout(new BorderLayout(8, 8));**

**JLabel title = new JLabel(p.getTitle(), SwingConstants.CENTER);**

**title.setFont(new Font("Segoe UI", Font.BOLD, 20));**

**add(title, BorderLayout.NORTH);**

**JPanel center = new JPanel(new BorderLayout(6, 6));**

**JTextArea desc = new JTextArea(p.getDescription() == null ? "No description." : p.getDescription());**

**desc.setLineWrap(true);**

**desc.setWrapStyleWord(true);**

**desc.setEditable(false);**

**desc.setFont(new Font("Segoe UI", Font.PLAIN, 14));**

**center.add(new JScrollPane(desc), BorderLayout.CENTER);**

**JPanel info = new JPanel(new GridLayout(3, 1));**

**info.add(new JLabel("Location: " + (p.getLocation() == null ? "-" : p.getLocation())));**

**info.add(new JLabel(String.format("Price: ₹%.2f", p.getPrice())));**

**info.add(new JLabel("Owner: " + (p.getOwnerEmail() == null ? "-" : p.getOwnerEmail())));**

**center.add(info, BorderLayout.SOUTH);**

**add(center, BorderLayout.CENTER);**

**JPanel bottom = new JPanel(new FlowLayout(FlowLayout.RIGHT));**

**JButton close = new JButton("Close");**

**close.addActionListener(e -> dispose());**

**bottom.add(close);**

**add(bottom, BorderLayout.SOUTH);**

**setVisible(true);**

**}**

**}**

**5.14 Main.java**

**package com.landmark.main;**

**import com.landmark.ui.OpeningPage;**

**import javax.swing.SwingUtilities;**

**public class Main {**

**public static void main(String[] args) {**

**SwingUtilities.invokeLater(() -> {**

**finished)**

**OpeningPage splash = new OpeningPage(1000);**

**splash.showAndLaunch();**

**});**

**}**

**}**

**6. User Manual**

**Customer Flow**

1. **Launch:  
   Execute Main.java to start the application. The LoginPage appears as the entry screen.**
2. **Access:**
   * **Login: Enter registered email/username and password. Authentication is handled through UserDAO.**
   * **Sign Up: Click “Create Account” to register as a new user. Credentials are securely stored in the database.**
3. **HomePage (Landmark Dashboard):  
   After login, users are directed to the HomePage, which displays a list of nearby landmarks fetched from the database. Users can search, filter, and sort landmarks based on category, distance, or popularity.**
4. **Landmark Details:  
   Clicking a landmark card opens the LandmarkDetailPage, displaying its name, description, images, location map, and user reviews.**
5. **Navigation or Save:**
   * **Navigate: Clicking the “Get Directions” button redirects to the integrated map view, showing the optimal route to reach the landmark.**
   * **Save Landmark: Users can click “Add to Favourites” to store the landmark in their profile for future access.**
6. **Review and Rating:  
   Users can submit reviews or rate landmarks using the Feedback Section, which updates dynamically in the database.**
7. **Profile Management:  
   Access the Profile Page to view or edit user information, change passwords, or view saved landmarks.**
8. **Logout:  
   Clicking “Logout” safely ends the session and returns the user to the LoginPage.**

**Admin Flow**

1. **Login:  
   The Admin logs in using admin credentials through the AdminLoginPage.**
2. **Dashboard Access:  
   The Admin Dashboard provides options to manage landmarks, users, and feedback data.**
3. **Landmark Management:**
   * **Add Landmark: Insert new landmarks with details such as name, location, description, and category.**
   * **Update Landmark: Modify existing landmark information directly from the dashboard.**
   * **Delete Landmark: Remove outdated or incorrect landmark entries from the database.**
4. **User Management:  
   View all registered users and manage their access or accounts if necessary.**
5. **Feedback Monitoring:  
   Admin can review and moderate user feedback to maintain quality and authenticity.**
6. **Reports (Future Scope):  
   Generate analytics and reports for popular landmarks, user activity, and feedback trends.**

**7.Test Cases**

| **Test ID** | **Module** | **Action / Input** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- | --- |
| **T01** | **User Registration** | **Enter new user details with a unique email ID.** | **Account created successfully; entry added to the User table.** | **PASS** |
| **T02** | **User Login** | **Enter valid username and password.** | **Home page loads successfully.** | **PASS** |
| **T03** | **User Login (Failure)** | **Enter invalid password.** | **Error message displayed: “Invalid username or password.”** | **PASS** |
| **T04** | **Landmark Search** | **Enter “Taj Mahal” in the search bar.** | **Display details and image of “Taj Mahal.”** | **PASS** |
| **T05** | **Map Navigation** | **Click “View on Map” for a selected landmark.** | **Opens map view with accurate location marker.** | **PASS** |
| **T06** | **Feedback Submission** | **Submit feedback text: “Great experience.”** | **Message stored successfully; confirmation displayed.** | **PASS** |

**8.** **Conclusion**

**The Landmark App successfully meets its objectives by providing users with an efficient platform to explore and learn about various landmarks. Through its user-friendly interface, accurate location mapping, and reliable search and feedback modules, the application ensures a smooth and informative experience. The successful completion of all functional tests confirms its stability, performance, and usability across different environments. Overall, the Landmark App serves as a practical and innovative solution for users to discover and interact with historical and cultural landmarks digitally.**

**9. Future Scope**

**The Landmark App has significant potential for future development and enhancement. Upcoming versions can include advanced features such as:**

* **AI-Based Recommendations: Suggest nearby landmarks based on user location and interests.**
* **AR/VR Integration: Provide immersive virtual tours of landmarks for better user experience.**
* **Offline Access: Allow users to view landmark details and maps without an internet connection.**
* **User Accounts & Personalization: Enable users to create profiles, save favorite landmarks, and track visited places.**
* **Multilingual Support: Offer content in multiple languages to reach a wider audience.**
* **Real-Time Navigation: Integrate with GPS systems for live route guidance to landmarks.**
* **Review & Rating System: Let users share feedback and rate landmarks to enhance community engagement**

**10.Reference**

| **1.Reference Type** | **Title / Description** | **Author / Source** |
| --- | --- | --- |
| **Relational Database** | **Database Design Concepts (Normalization, Keys, Relationships, Transactions)** | **C.J. Date, *An Introduction to Database Systems* (or similar academic text).** |
| **Object-Oriented Design** | **UML (Unified Modeling Language) Standards for Class and Use Case Diagrams** | ***UML 2.5.1 Specification* (or equivalent modeling reference).** |
| **Design Patterns** | **Object-Oriented Patterns (MVC, Singleton, Observer)** | **Erich Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software.*** |