

Object-Oriented Programming

Assignment Report

BTech 4th Semester

2024 - 2025

Project Title: Smart City Application



Submitted To: Mr. Aryan Assistant Professor (SoCS)

Submitted By: Anurag Pandey 500120453

Smart City Application

Abstract

The rapid urbanization and technological advancements have led to the need for smart solutions to improve city infrastructure and enhance the overall quality of life. This project, **Smart City Application**, is designed to serve as an interactive guide for new visitors by providing crucial information about the city, including hotels, restaurants, shopping malls, tourist places, and transportation options. The application is developed using **Java Swing for the UI and MySQL for the database**, ensuring efficient data management and user-friendly navigation.

The system allows users to search for places based on location and displays relevant details such as contact information, pricing, opening hours, and ratings. The transportation module enhances user convenience by providing an interactive **cab-like booking system**. The project follows object-oriented programming (OOP) principles to ensure modularity and scalability.

By integrating technology with urban infrastructure, this application aims to streamline city exploration, making it easier for residents and visitors to access essential services. The future scope includes enhancing the application with real-time updates, GPS-based tracking, and additional smart city functionalities.

Acknowledgment

I would like to express my heartfelt gratitude to my **instructor**, whose invaluable guidance and constructive feedback have been instrumental in the successful completion of this project. Their insights into **Java programming, database management, and UI design** have significantly contributed to my learning experience.

I am also thankful to my **peers and classmates** for their continuous support and encouragement throughout the development process. Their suggestions helped in refining the application and making it more user-friendly.

Finally, I extend my appreciation to my **family and friends**, whose unwavering motivation kept me dedicated to completing this project efficiently. This journey has been a great learning experience, and I look forward to applying these skills in future smart city initiatives.

Table of Contents

Sr. No	Title
1	Introduction
2	Flow Chart
3	UML Diagram
4	Software Requirements
5	Project Directory Structure
6	Code Snippets
7	Database Schema and Tables
8	Output Screenshots
9	Project Summary
10	Important Learnings

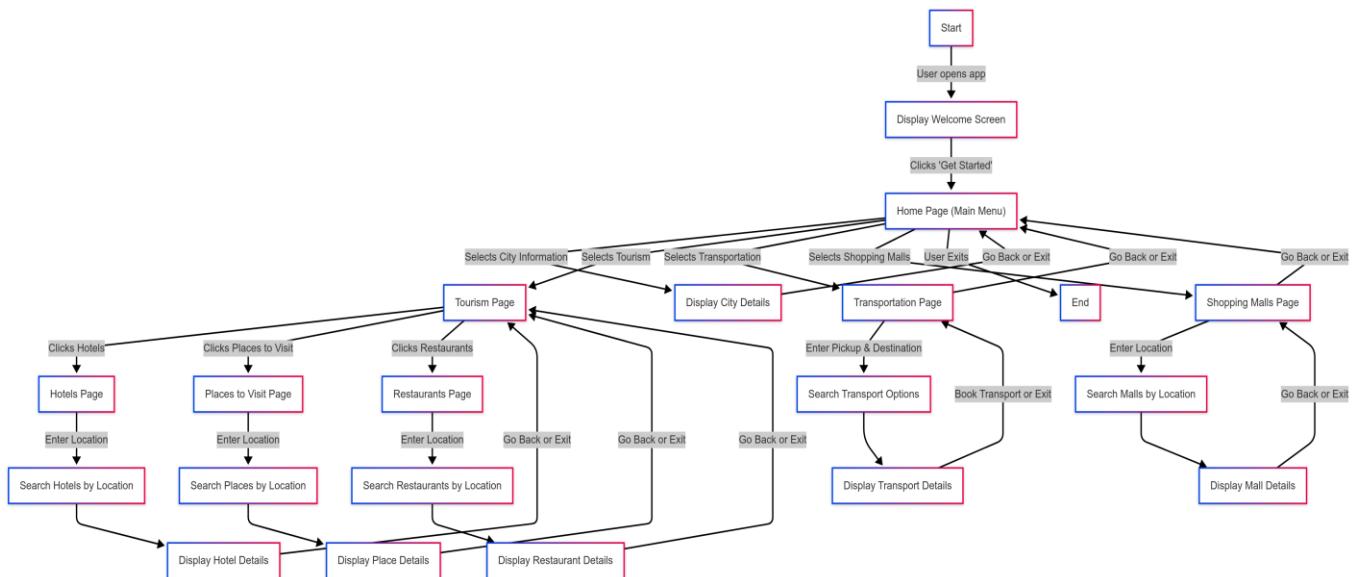
1. Introduction

The Smart City Application is a Java Swing-based desktop application that serves as a digital city guide. It provides users with easy access to essential city services, making their navigation seamless.

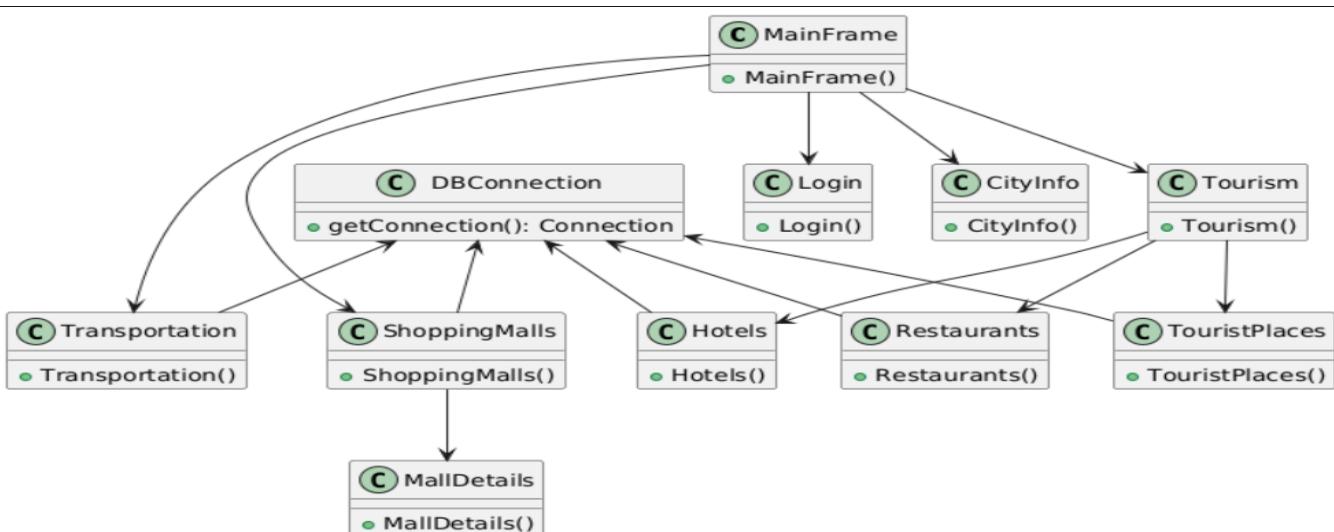
Key Features:

- City Information: Overview of major landmarks and facilities.
- Tourism Guide: Information about hotels, restaurants, and tourist attractions.
- Transportation System: Allows users to find and book rides from different locations.
- Shopping Malls: Provides details of malls, including their opening and closing times.
- User-Friendly Interface: Interactive UI designed to enhance the user experience.

2. Flow Chart



3. UML Diagram



4. Software Requirements

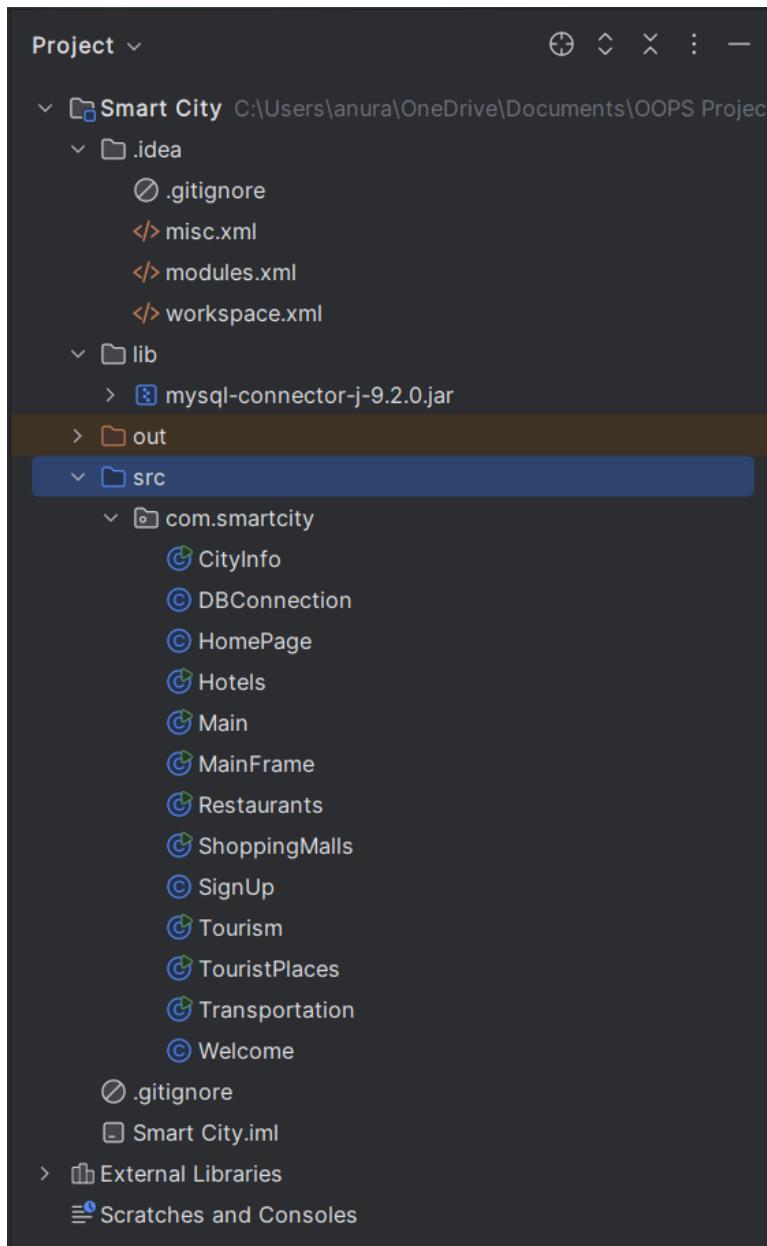
Hardware Requirements:

- Processor: Intel Core i3 or above
- RAM: Minimum 4GB
- Storage: Minimum 500MB

Software Requirements:

- Programming Language: Java (Swing for UI, JDBC for database connectivity)
- Database: MySQL
- IDE Used: IntelliJ IDEA Community Edition
- Libraries: Swing, AWT, MySQL Connector/J
- Operating System: Windows/Linux

5. Project Directory Structure



6. Database Schema and Tables

Query 1 ×

The screenshot shows the MySQL Workbench interface. The top part is a query editor titled "Query 1" containing a series of SQL commands. The bottom part is a results grid titled "Result Grid" showing the names of tables in the "smartcity" database.

```
1 • use smartcity;
2 • CREATE TABLE users (
3     id INT AUTO_INCREMENT PRIMARY KEY,
4     name VARCHAR(100),
5     username VARCHAR(50) UNIQUE NOT NULL,
6     password VARCHAR(50) NOT NULL,
7     dob VARCHAR(20),
8     country VARCHAR(50),
9     state VARCHAR(50),
10    pincode VARCHAR(10),
11    mobile VARCHAR(15) UNIQUE
12 );
13 • select * from users;
14 • show tables;
15 • select * from city_information;
16 • select * from tourism;
17 • select * from hotels;
18 • select * from restaurants;
19 • select * from transportation;
20 • select * from shopping_mall;
```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Co

Tables_in_smartcity
city_information
hotels
restaurants
shopping_mall
tourism
transportation
user_feedback
users

7. Code Snippets

A screenshot of a Java code editor showing the file `Main.java`. The code defines a `Main` class with a `main` method that creates a `MainFrame` instance and starts it with a `Welcome` page. The code editor has a dark theme with syntax highlighting. A status bar at the top right shows network connectivity with "1.0 Kbps".

```
1 package com.smartcity;
2
3 public class Main {
4     public static void main(String[] args) {
5         MainFrame mainFrame = new MainFrame(); // Create MainFrame instance
6         new Welcome(mainFrame); // Start with Welcome page
7     }
8 }
9
```

A screenshot of a Java code editor showing the file `MainFrame.java`. The code defines a `MainFrame` class that extends `JFrame`. It sets the title to "Expense Tracker", specifies a size of 500x400 pixels, and uses the `EXIT_ON_CLOSE` operation. The code editor has a dark theme with syntax highlighting. A tab bar at the top shows other files like `CheckBudget.java`, `HomePage.java`, `Signup.java`, `Update.java`, `ViewExpenses.java`, and `Welcome.java`. A status bar at the top right shows network connectivity with "0.1 Kbps" and "0.0 Kbps".

```
1 package com.expensetracker;
2
3 import javax.swing.*;
4
5 public class MainFrame extends JFrame { 4 usages
6     public MainFrame() { 2 usages
7         setTitle("Expense Tracker");
8         setSize( width: 500, height: 400);
9         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10        setLocationRelativeTo(null);
11    }
12 }
13
```

```
2.0 Kbps
0.4 Kbps
Main.java MainFrame.java Welcome.java
1 package com.smartcity;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.util.logging.Level;
7 import java.util.logging.Logger;
8
9 public class Welcome extends JFrame { 3 usages
10     private JPanel welcomePanel; 6 usages
11     private JPanel loginPanel; 10 usages
12     private final MainFrame mainFrame; 2 usages
13
14     public Welcome(MainFrame mainFrame) { 2 usages
15         this.mainFrame = mainFrame;
16         setTitle("Smart City");
17         setSize( width: 500, height: 300);
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         setLocationRelativeTo(null);
20         setLayout(new CardLayout());
21
22         // Create Welcome and Login Panels
23         createWelcomePanel();
24         createLoginPanel();
25
26         // Initially show the Welcome Panel
27         setContentPane(welcomePanel);
28         setVisible(true);
29     }
30
31     private void createWelcomePanel() { 1 usage
32         welcomePanel = new JPanel(new BorderLayout());
33         welcomePanel.setBackground(new Color( r: 220, g: 235, b: 255)); // Soft light blue
34
35         // **Title: "Welcome to Smart City"**
36         JLabel headline = new JLabel( text: "Welcome to Smart City", SwingConstants.CENTER);
37         headline.setFont(new Font( name: "Arial", Font.BOLD, size: 26));
38         headline.setForeground(new Color( r: 0, g: 76, b: 153)); // Dark blue for elegance
39
40         // **Subheading**
41         JLabel subheading = new JLabel( text: "Your Guide to Explore and Navigate", SwingConstants.CENTER);
42         subheading.setFont(new Font( name: "Arial", Font.PLAIN, size: 16));
43         subheading.setForeground(new Color( r: 80, g: 80, b: 80)); // Dark gray for readability
44
45         // **Button Panel for Centering**
46         JPanel buttonPanel = new JPanel();
47         buttonPanel.setBackground(new Color( r: 220, g: 235, b: 255)); // Same as main panel
48         JButton getStartedButton = new JButton( text: "Get Started");
49         getStartedButton.setFont(new Font( name: "Arial", Font.BOLD, size: 16));
50         getStartedButton.setBackground(new Color( r: 0, g: 153, b: 76)); // Green
51         getStartedButton.setForeground(Color.WHITE);
52         getStartedButton.setFocusPainted(false);
53         getStartedButton.setPreferredSize(new Dimension( width: 150, height: 40)); // Proper size
54
55         // **Hover Effect**
56         getStartedButton.addMouseListener(new java.awt.event.MouseAdapter() {
57             @Override
58             public void mouseEntered(java.awt.event.MouseEvent evt) {
59                 getStartedButton.setBackground(new Color( r: 0, g: 204, b: 102)); // Lighter green
60             }
61             public void mouseExited(java.awt.event.MouseEvent evt) {
62                 getStartedButton.setBackground(new Color( r: 0, g: 153, b: 76));
63             }
64         });
65     }
66 }
```

```
64         buttonPanel.add(getStartedButton);
65
66         // **Adding Components**
67         welcomePanel.add(subheading, BorderLayout.NORTH);
68         welcomePanel.add(headline, BorderLayout.CENTER);
69         welcomePanel.add(buttonPanel, BorderLayout.SOUTH);
70
71         // **Button Click Action: Switch to Login**
72         getStartedButton.addActionListener( ActionEvent e -> {
73             setContentPane(loginPanel);
74             revalidate();
75             repaint();
76         });
77     }
78 }
79
80 private void createLoginPanel() { 1 usage
81     loginPanel = new JPanel(new GridBagLayout());
82     loginPanel.setBackground(Color.WHITE); // White background for contrast
83
84     GridBagConstraints gbc = new GridBagConstraints();
85     gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10); // Padding
86
87     // Title
88     JLabel titleLabel = new JLabel( text: "Login to Smart City");
89     titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 20));
90     gbc.gridx = 0;
91     gbc.gridy = 0;
92     gbc.gridwidth = 2;
93     loginPanel.add(titleLabel, gbc);
94
95
96     // Username Label
97     gbc.gridwidth = 1;
98     gbc.gridx = 0;
99     gbc.gridy = 1;
100    loginPanel.add(new JLabel( text: "Username:"), gbc);
101
102    // Username Field
103    JTextField userField = new JTextField( columns: 15);
104    gbc.gridx = 1;
105    loginPanel.add(userField, gbc);
106
107    // Password Label
108    gbc.gridx = 0;
109    gbc.gridy = 2;
110    loginPanel.add(new JLabel( text: "Password:"), gbc);
111
112    // Password Field
113    JPasswordField passField = new JPasswordField( columns: 15);
114    gbc.gridx = 1;
115    loginPanel.add(passField, gbc);
116
117    // Sign In Button
118    JButton loginButton = new JButton( text: "Sign In");
119    loginButton.setBackground(new Color( r: 0, g: 102, b: 204)); // Blue color
120    loginButton.setForeground(Color.WHITE);
121    loginButton.setFocusPainted(false);
122    gbc.gridx = 0;
123    gbc.gridy = 3;
124    gbc.gridwidth = 2;
125    loginPanel.add(loginButton, gbc);
```

```
125
126     // Sign Up Button
127     JButton signupButton = new JButton( text: "Sign Up");
128     signupButton.setBackground(new Color( r: 0, g: 204, b: 102)); // Green color
129     signupButton.setForeground(Color.WHITE);
130     signupButton.setFocusPainted(false);
131     gbc.gridx = 4;
132     loginPanel.add(signupButton, gbc);
133
134     // Sign In Action
135     loginButton.addActionListener( ActionEvent e -> {
136         String username = userField.getText();
137         String password = new String(passField.getPassword());
138
139         if (validateUser(username, password)) {
140             JOptionPane.showMessageDialog( parentComponent: null, message: "Login Successful!");
141             dispose(); // Close Welcome Window
142             mainFrame.showMainUI(); // Open MainFrame
143         } else {
144             JOptionPane.showMessageDialog( parentComponent: null, message: "Incorrect Username or Password!", title: "Error");
145         }
146     });
147
148     // Open SignUp Window
149     signupButton.addActionListener( ActionEvent e -> new SignUp());
150 }
151
152
153
154     private boolean validateUser(String username, String password) { 1 usage
155
156         private boolean validateUser(String username, String password) { 1 usage
157             try (Connection con = DBConnection.getConnection()) {
158                 assert con != null;
159                 try (PreparedStatement pst = con.prepareStatement( sql: "SELECT * FROM users WHERE username=? AND password=?"));
160                     pst.setString( parameterIndex: 1, username);
161                     pst.setString( parameterIndex: 2, password);
162                     ResultSet rs = pst.executeQuery();
163                     return rs.next(); // Returns true if user exists
164                 }
165             } catch (SQLException ex) {
166                 Logger.getLogger(Welcome.class.getName()).log(Level.SEVERE, msg: "Database error", ex);
167                 return false;
168             }
169         }
170     }
```

```
1 package com.smartcity;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class HomePage extends JPanel { 2 usages
7     public HomePage(MainFrame mainFrame) { 1 usage
8         setLayout(new GridBagLayout());
9         setBackground(new Color( r: 230, g: 245, b: 255)); // Light background
10
11     // **Card Panel for Centered Content**
12     JPanel cardPanel = new JPanel() {
13         @Override
14     protected void paintComponent(Graphics g) {
15         super.paintComponent(g);
16         Graphics2D g2d = (Graphics2D) g;
17         int width = getWidth();
18         int height = getHeight();
19         Color color1 = new Color( r: 180, g: 220, b: 255);
20         Color color2 = new Color( r: 120, g: 180, b: 255);
21         GradientPaint gp = new GradientPaint( x1: 0, y1: 0, color1, width, height, color2);
22         g2d.setPaint(gp);
23         g2d.fillRoundRect( x: 0, y: 0, width, height, arcWidth: 30, arcHeight: 30); // Rounded Corners
24     }
25     cardPanel.setPreferredSize(new Dimension( width: 450, height: 300));
26     cardPanel.setLayout(new GridBagLayout());
27     cardPanel.setOpaque(false);
28
29     GridBagConstraints gbc = new GridBagConstraints();
30     gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
31     gbc.fill = GridBagConstraints.BOTH;
32
33
34     // **Title**
35     JLabel titleLabel = new JLabel( text: "Smart City Guide", SwingConstants.CENTER);
36     titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 26));
37     titleLabel.setForeground(new Color( r: 10, g: 10, b: 70));
38     gbc.gridx = 0;
39     gbc.gridy = 0;
40     gbc.gridwidth = 2;
41     cardPanel.add(titleLabel, gbc);
42
43     // **Buttons with Styling**
44     JButton cityInfoButton = createStyledButton( text: "City Information");
45     JButton tourismButton = createStyledButton( text: "Tourism");
46     JButton transportButton = createStyledButton( text: "Transportation");
47     JButton shoppingButton = createStyledButton( text: "Shopping Malls");
48
49     gbc.gridwidth = 1;
50     gbc.gridy = 1;
51     gbc.gridx = 0;
52     cardPanel.add(cityInfoButton, gbc);
53
54     gbc.gridx = 1;
55     cardPanel.add(tourismButton, gbc);
56
57     gbc.gridy = 2;
58     gbc.gridx = 0;
59     cardPanel.add(transportButton, gbc);
60
61     gbc.gridx = 1;
62     cardPanel.add(shoppingButton, gbc);
63
64     add(cardPanel);
```

```

66         // **Button Actions**
67         cityInfoButton.addActionListener( ActionEvent e -> new CityInfo().setVisible(true));
68         tourismButton.addActionListener( ActionEvent e -> new Tourism().setVisible(true));
69         transportButton.addActionListener( ActionEvent e -> new Transportation().setVisible(true));
70         shoppingButton.addActionListener( ActionEvent e -> new ShoppingMalls().setVisible(true));
71     }
72
73     // **Create a Styled Button**
74     @private JButton createStyledButton(String text) { 4 usages
75         JButton button = new JButton(text);
76         button.setFont(new Font( name: "Arial", Font.BOLD, size: 16));
77         button.setBackground(new Color( r: 0, g: 102, b: 204));
78         button.setForeground(Color.WHITE);
79         button.setFocusPainted(false);
80         button.setBorder(BorderFactory.createEmptyBorder( top: 10, left: 20, bottom: 10, right: 20));
81         button.setPreferredSize(new Dimension( width: 180, height: 45));
82         button.setCursor(new Cursor(Cursor.HAND_CURSOR));
83
84         // **3D Button Effect**
85         button.setBorder(BorderFactory.createRaisedSoftBevelBorder());
86
87         // **Hover Effect**
88         button.addMouseListener(new java.awt.event.MouseAdapter() {
89             @Override
90             public void mouseEntered(java.awt.event.MouseEvent evt) {
91                 button.setBackground(new Color( r: 0, g: 153, b: 255));
92                 button.setBorder(BorderFactory.createLoweredSoftBevelBorder());
93             }
94             public void mouseExited(java.awt.event.MouseEvent evt) {
95                 button.setBackground(new Color( r: 0, g: 102, b: 204));
96                 button.setBorder(BorderFactory.createRaisedSoftBevelBorder());
97             }
98         });
99
100        return button;
101    }
102}

```

CityInfo.java × 4.1 Kbps

```

1 package com.smartcity;
2
3 import javax.swing.*;
4 import javax.swing.border.EmptyBorder;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 public class CityInfo extends JFrame {
14     private final JTextField searchField; 5 usages
15     private final JLabel cityNameLabel; 5 usages
16     private final JTextPane infoPane; 8 usages
17
18     public void search() {
19         String query = searchField.getText();
20         Connection connection = null;
21         PreparedStatement statement = null;
22         ResultSet resultSet = null;
23
24         try {
25             connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/smartcity", "root", "password");
26             statement = connection.prepareStatement("SELECT * FROM cities WHERE name LIKE ?");
27             statement.setString(1, "%" + query + "%");
28             resultSet = statement.executeQuery();
29
30             while (resultSet.next()) {
31                 String name = resultSet.getString("name");
32                 String population = resultSet.getString("population");
33                 String area = resultSet.getString("area");
34
35                 cityNameLabel.setText(name);
36                 infoPane.setText("Population: " + population + "\nArea: " + area);
37             }
38
39         } catch (SQLException e) {
40             e.printStackTrace();
41         } finally {
42             if (resultSet != null) {
43                 try {
44                     resultSet.close();
45                 } catch (SQLException e) {
46                     e.printStackTrace();
47                 }
48             }
49             if (statement != null) {
50                 try {
51                     statement.close();
52                 } catch (SQLException e) {
53                     e.printStackTrace();
54                 }
55             }
56             if (connection != null) {
57                 try {
58                     connection.close();
59                 } catch (SQLException e) {
60                     e.printStackTrace();
61                 }
62             }
63         }
64     }
65
66     public void updateUI() {
67         // Update UI logic here
68     }
69
70     public void handleEvent(ActionEvent event) {
71         if (event.getSource() == searchField) {
72             search();
73         }
74     }
75
76     public void handleEvent(Event event) {
77         if (event instanceof MouseEvent) {
78             handleMouseEvent((MouseEvent) event);
79         }
80     }
81
82     private void handleMouseEvent(MouseEvent event) {
83         if (event.isPopupTrigger()) {
84             handlePopupMenu();
85         }
86     }
87
88     private void handlePopupMenu() {
89         // HandlePopupMenu logic here
90     }
91
92     public void handleEvent(Event event) {
93         if (event instanceof KeyEvent) {
94             handleKeyEvent((KeyEvent) event);
95         }
96     }
97
98     private void handleKeyEvent(KeyEvent event) {
99         if (event.getKeyCode() == KeyEvent.VK_ENTER) {
100            search();
101        }
102    }
103
104    public void handleEvent(Event event) {
105        if (event instanceof FocusEvent) {
106            handleFocusEvent((FocusEvent) event);
107        }
108    }
109
110    private void handleFocusEvent(FocusEvent event) {
111        if (event.getFocusOwner() == searchField) {
112            search();
113        }
114    }
115
116    public void handleEvent(Event event) {
117        if (event instanceof InputEvent) {
118            handleInputEvent((InputEvent) event);
119        }
120    }
121
122    private void handleInputEvent(InputEvent event) {
123        if (event instanceof KeyEvent) {
124            handleKeyEvent((KeyEvent) event);
125        }
126    }
127
128    public void handleEvent(Event event) {
129        if (event instanceof MouseEvent) {
130            handleMouseEvent((MouseEvent) event);
131        }
132    }
133
134    private void handleMouseEvent(MouseEvent event) {
135        if (event.isPopupTrigger()) {
136            handlePopupMenu();
137        }
138    }
139
140    private void handlePopupMenu() {
141        // HandlePopupMenu logic here
142    }
143
144    public void handleEvent(Event event) {
145        if (event instanceof KeyEvent) {
146            handleKeyEvent((KeyEvent) event);
147        }
148    }
149
150    private void handleKeyEvent(KeyEvent event) {
151        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
152            search();
153        }
154    }
155
156    public void handleEvent(Event event) {
157        if (event instanceof FocusEvent) {
158            handleFocusEvent((FocusEvent) event);
159        }
160    }
161
162    private void handleFocusEvent(FocusEvent event) {
163        if (event.getFocusOwner() == searchField) {
164            search();
165        }
166    }
167
168    public void handleEvent(Event event) {
169        if (event instanceof InputEvent) {
170            handleInputEvent((InputEvent) event);
171        }
172    }
173
174    private void handleInputEvent(InputEvent event) {
175        if (event instanceof KeyEvent) {
176            handleKeyEvent((KeyEvent) event);
177        }
178    }
179
180    public void handleEvent(Event event) {
181        if (event instanceof MouseEvent) {
182            handleMouseEvent((MouseEvent) event);
183        }
184    }
185
186    private void handleMouseEvent(MouseEvent event) {
187        if (event.isPopupTrigger()) {
188            handlePopupMenu();
189        }
190    }
191
192    private void handlePopupMenu() {
193        // HandlePopupMenu logic here
194    }
195
196    public void handleEvent(Event event) {
197        if (event instanceof KeyEvent) {
198            handleKeyEvent((KeyEvent) event);
199        }
200    }
201
202    private void handleKeyEvent(KeyEvent event) {
203        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
204            search();
205        }
206    }
207
208    public void handleEvent(Event event) {
209        if (event instanceof FocusEvent) {
210            handleFocusEvent((FocusEvent) event);
211        }
212    }
213
214    private void handleFocusEvent(FocusEvent event) {
215        if (event.getFocusOwner() == searchField) {
216            search();
217        }
218    }
219
220    public void handleEvent(Event event) {
221        if (event instanceof InputEvent) {
222            handleInputEvent((InputEvent) event);
223        }
224    }
225
226    private void handleInputEvent(InputEvent event) {
227        if (event instanceof KeyEvent) {
228            handleKeyEvent((KeyEvent) event);
229        }
230    }
231
232    public void handleEvent(Event event) {
233        if (event instanceof MouseEvent) {
234            handleMouseEvent((MouseEvent) event);
235        }
236    }
237
238    private void handleMouseEvent(MouseEvent event) {
239        if (event.isPopupTrigger()) {
240            handlePopupMenu();
241        }
242    }
243
244    private void handlePopupMenu() {
245        // HandlePopupMenu logic here
246    }
247
248    public void handleEvent(Event event) {
249        if (event instanceof KeyEvent) {
250            handleKeyEvent((KeyEvent) event);
251        }
252    }
253
254    private void handleKeyEvent(KeyEvent event) {
255        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
256            search();
257        }
258    }
259
260    public void handleEvent(Event event) {
261        if (event instanceof FocusEvent) {
262            handleFocusEvent((FocusEvent) event);
263        }
264    }
265
266    private void handleFocusEvent(FocusEvent event) {
267        if (event.getFocusOwner() == searchField) {
268            search();
269        }
270    }
271
272    public void handleEvent(Event event) {
273        if (event instanceof InputEvent) {
274            handleInputEvent((InputEvent) event);
275        }
276    }
277
278    private void handleInputEvent(InputEvent event) {
279        if (event instanceof KeyEvent) {
280            handleKeyEvent((KeyEvent) event);
281        }
282    }
283
284    public void handleEvent(Event event) {
285        if (event instanceof MouseEvent) {
286            handleMouseEvent((MouseEvent) event);
287        }
288    }
289
290    private void handleMouseEvent(MouseEvent event) {
291        if (event.isPopupTrigger()) {
292            handlePopupMenu();
293        }
294    }
295
296    private void handlePopupMenu() {
297        // HandlePopupMenu logic here
298    }
299
300    public void handleEvent(Event event) {
301        if (event instanceof KeyEvent) {
302            handleKeyEvent((KeyEvent) event);
303        }
304    }
305
306    private void handleKeyEvent(KeyEvent event) {
307        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
308            search();
309        }
310    }
311
312    public void handleEvent(Event event) {
313        if (event instanceof FocusEvent) {
314            handleFocusEvent((FocusEvent) event);
315        }
316    }
317
318    private void handleFocusEvent(FocusEvent event) {
319        if (event.getFocusOwner() == searchField) {
320            search();
321        }
322    }
323
324    public void handleEvent(Event event) {
325        if (event instanceof InputEvent) {
326            handleInputEvent((InputEvent) event);
327        }
328    }
329
330    private void handleInputEvent(InputEvent event) {
331        if (event instanceof KeyEvent) {
332            handleKeyEvent((KeyEvent) event);
333        }
334    }
335
336    public void handleEvent(Event event) {
337        if (event instanceof MouseEvent) {
338            handleMouseEvent((MouseEvent) event);
339        }
340    }
341
342    private void handleMouseEvent(MouseEvent event) {
343        if (event.isPopupTrigger()) {
344            handlePopupMenu();
345        }
346    }
347
348    private void handlePopupMenu() {
349        // HandlePopupMenu logic here
350    }
351
352    public void handleEvent(Event event) {
353        if (event instanceof KeyEvent) {
354            handleKeyEvent((KeyEvent) event);
355        }
356    }
357
358    private void handleKeyEvent(KeyEvent event) {
359        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
360            search();
361        }
362    }
363
364    public void handleEvent(Event event) {
365        if (event instanceof FocusEvent) {
366            handleFocusEvent((FocusEvent) event);
367        }
368    }
369
370    private void handleFocusEvent(FocusEvent event) {
371        if (event.getFocusOwner() == searchField) {
372            search();
373        }
374    }
375
376    public void handleEvent(Event event) {
377        if (event instanceof InputEvent) {
378            handleInputEvent((InputEvent) event);
379        }
380    }
381
382    private void handleInputEvent(InputEvent event) {
383        if (event instanceof KeyEvent) {
384            handleKeyEvent((KeyEvent) event);
385        }
386    }
387
388    public void handleEvent(Event event) {
389        if (event instanceof MouseEvent) {
390            handleMouseEvent((MouseEvent) event);
391        }
392    }
393
394    private void handleMouseEvent(MouseEvent event) {
395        if (event.isPopupTrigger()) {
396            handlePopupMenu();
397        }
398    }
399
400    private void handlePopupMenu() {
401        // HandlePopupMenu logic here
402    }
403
404    public void handleEvent(Event event) {
405        if (event instanceof KeyEvent) {
406            handleKeyEvent((KeyEvent) event);
407        }
408    }
409
410    private void handleKeyEvent(KeyEvent event) {
411        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
412            search();
413        }
414    }
415
416    public void handleEvent(Event event) {
417        if (event instanceof FocusEvent) {
418            handleFocusEvent((FocusEvent) event);
419        }
420    }
421
422    private void handleFocusEvent(FocusEvent event) {
423        if (event.getFocusOwner() == searchField) {
424            search();
425        }
426    }
427
428    public void handleEvent(Event event) {
429        if (event instanceof InputEvent) {
430            handleInputEvent((InputEvent) event);
431        }
432    }
433
434    private void handleInputEvent(InputEvent event) {
435        if (event instanceof KeyEvent) {
436            handleKeyEvent((KeyEvent) event);
437        }
438    }
439
440    public void handleEvent(Event event) {
441        if (event instanceof MouseEvent) {
442            handleMouseEvent((MouseEvent) event);
443        }
444    }
445
446    private void handleMouseEvent(MouseEvent event) {
447        if (event.isPopupTrigger()) {
448            handlePopupMenu();
449        }
450    }
451
452    private void handlePopupMenu() {
453        // HandlePopupMenu logic here
454    }
455
456    public void handleEvent(Event event) {
457        if (event instanceof KeyEvent) {
458            handleKeyEvent((KeyEvent) event);
459        }
460    }
461
462    private void handleKeyEvent(KeyEvent event) {
463        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
464            search();
465        }
466    }
467
468    public void handleEvent(Event event) {
469        if (event instanceof FocusEvent) {
470            handleFocusEvent((FocusEvent) event);
471        }
472    }
473
474    private void handleFocusEvent(FocusEvent event) {
475        if (event.getFocusOwner() == searchField) {
476            search();
477        }
478    }
479
480    public void handleEvent(Event event) {
481        if (event instanceof InputEvent) {
482            handleInputEvent((InputEvent) event);
483        }
484    }
485
486    private void handleInputEvent(InputEvent event) {
487        if (event instanceof KeyEvent) {
488            handleKeyEvent((KeyEvent) event);
489        }
490    }
491
492    public void handleEvent(Event event) {
493        if (event instanceof MouseEvent) {
494            handleMouseEvent((MouseEvent) event);
495        }
496    }
497
498    private void handleMouseEvent(MouseEvent event) {
499        if (event.isPopupTrigger()) {
500            handlePopupMenu();
501        }
502    }
503
504    private void handlePopupMenu() {
505        // HandlePopupMenu logic here
506    }
507
508    public void handleEvent(Event event) {
509        if (event instanceof KeyEvent) {
510            handleKeyEvent((KeyEvent) event);
511        }
512    }
513
514    private void handleKeyEvent(KeyEvent event) {
515        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
516            search();
517        }
518    }
519
520    public void handleEvent(Event event) {
521        if (event instanceof FocusEvent) {
522            handleFocusEvent((FocusEvent) event);
523        }
524    }
525
526    private void handleFocusEvent(FocusEvent event) {
527        if (event.getFocusOwner() == searchField) {
528            search();
529        }
530    }
531
532    public void handleEvent(Event event) {
533        if (event instanceof InputEvent) {
534            handleInputEvent((InputEvent) event);
535        }
536    }
537
538    private void handleInputEvent(InputEvent event) {
539        if (event instanceof KeyEvent) {
540            handleKeyEvent((KeyEvent) event);
541        }
542    }
543
544    public void handleEvent(Event event) {
545        if (event instanceof MouseEvent) {
546            handleMouseEvent((MouseEvent) event);
547        }
548    }
549
550    private void handleMouseEvent(MouseEvent event) {
551        if (event.isPopupTrigger()) {
552            handlePopupMenu();
553        }
554    }
555
556    private void handlePopupMenu() {
557        // HandlePopupMenu logic here
558    }
559
560    public void handleEvent(Event event) {
561        if (event instanceof KeyEvent) {
562            handleKeyEvent((KeyEvent) event);
563        }
564    }
565
566    private void handleKeyEvent(KeyEvent event) {
567        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
568            search();
569        }
570    }
571
572    public void handleEvent(Event event) {
573        if (event instanceof FocusEvent) {
574            handleFocusEvent((FocusEvent) event);
575        }
576    }
577
578    private void handleFocusEvent(FocusEvent event) {
579        if (event.getFocusOwner() == searchField) {
580            search();
581        }
582    }
583
584    public void handleEvent(Event event) {
585        if (event instanceof InputEvent) {
586            handleInputEvent((InputEvent) event);
587        }
588    }
589
590    private void handleInputEvent(InputEvent event) {
591        if (event instanceof KeyEvent) {
592            handleKeyEvent((KeyEvent) event);
593        }
594    }
595
596    public void handleEvent(Event event) {
597        if (event instanceof MouseEvent) {
598            handleMouseEvent((MouseEvent) event);
599        }
600    }
601
602    private void handleMouseEvent(MouseEvent event) {
603        if (event.isPopupTrigger()) {
604            handlePopupMenu();
605        }
606    }
607
608    private void handlePopupMenu() {
609        // HandlePopupMenu logic here
610    }
611
612    public void handleEvent(Event event) {
613        if (event instanceof KeyEvent) {
614            handleKeyEvent((KeyEvent) event);
615        }
616    }
617
618    private void handleKeyEvent(KeyEvent event) {
619        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
620            search();
621        }
622    }
623
624    public void handleEvent(Event event) {
625        if (event instanceof FocusEvent) {
626            handleFocusEvent((FocusEvent) event);
627        }
628    }
629
630    private void handleFocusEvent(FocusEvent event) {
631        if (event.getFocusOwner() == searchField) {
632            search();
633        }
634    }
635
636    public void handleEvent(Event event) {
637        if (event instanceof InputEvent) {
638            handleInputEvent((InputEvent) event);
639        }
640    }
641
642    private void handleInputEvent(InputEvent event) {
643        if (event instanceof KeyEvent) {
644            handleKeyEvent((KeyEvent) event);
645        }
646    }
647
648    public void handleEvent(Event event) {
649        if (event instanceof MouseEvent) {
650            handleMouseEvent((MouseEvent) event);
651        }
652    }
653
654    private void handleMouseEvent(MouseEvent event) {
655        if (event.isPopupTrigger()) {
656            handlePopupMenu();
657        }
658    }
659
660    private void handlePopupMenu() {
661        // HandlePopupMenu logic here
662    }
663
664    public void handleEvent(Event event) {
665        if (event instanceof KeyEvent) {
666            handleKeyEvent((KeyEvent) event);
667        }
668    }
669
670    private void handleKeyEvent(KeyEvent event) {
671        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
672            search();
673        }
674    }
675
676    public void handleEvent(Event event) {
677        if (event instanceof FocusEvent) {
678            handleFocusEvent((FocusEvent) event);
679        }
680    }
681
682    private void handleFocusEvent(FocusEvent event) {
683        if (event.getFocusOwner() == searchField) {
684            search();
685        }
686    }
687
688    public void handleEvent(Event event) {
689        if (event instanceof InputEvent) {
690            handleInputEvent((InputEvent) event);
691        }
692    }
693
694    private void handleInputEvent(InputEvent event) {
695        if (event instanceof KeyEvent) {
696            handleKeyEvent((KeyEvent) event);
697        }
698    }
699
700    public void handleEvent(Event event) {
701        if (event instanceof MouseEvent) {
702            handleMouseEvent((MouseEvent) event);
703        }
704    }
705
706    private void handleMouseEvent(MouseEvent event) {
707        if (event.isPopupTrigger()) {
708            handlePopupMenu();
709        }
710    }
711
712    private void handlePopupMenu() {
713        // HandlePopupMenu logic here
714    }
715
716    public void handleEvent(Event event) {
717        if (event instanceof KeyEvent) {
718            handleKeyEvent((KeyEvent) event);
719        }
720    }
721
722    private void handleKeyEvent(KeyEvent event) {
723        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
724            search();
725        }
726    }
727
728    public void handleEvent(Event event) {
729        if (event instanceof FocusEvent) {
730            handleFocusEvent((FocusEvent) event);
731        }
732    }
733
734    private void handleFocusEvent(FocusEvent event) {
735        if (event.getFocusOwner() == searchField) {
736            search();
737        }
738    }
739
740    public void handleEvent(Event event) {
741        if (event instanceof InputEvent) {
742            handleInputEvent((InputEvent) event);
743        }
744    }
745
746    private void handleInputEvent(InputEvent event) {
747        if (event instanceof KeyEvent) {
748            handleKeyEvent((KeyEvent) event);
749        }
750    }
751
752    public void handleEvent(Event event) {
753        if (event instanceof MouseEvent) {
754            handleMouseEvent((MouseEvent) event);
755        }
756    }
757
758    private void handleMouseEvent(MouseEvent event) {
759        if (event.isPopupTrigger()) {
760            handlePopupMenu();
761        }
762    }
763
764    private void handlePopupMenu() {
765        // HandlePopupMenu logic here
766    }
767
768    public void handleEvent(Event event) {
769        if (event instanceof KeyEvent) {
770            handleKeyEvent((KeyEvent) event);
771        }
772    }
773
774    private void handleKeyEvent(KeyEvent event) {
775        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
776            search();
777        }
778    }
779
780    public void handleEvent(Event event) {
781        if (event instanceof FocusEvent) {
782            handleFocusEvent((FocusEvent) event);
783        }
784    }
785
786    private void handleFocusEvent(FocusEvent event) {
787        if (event.getFocusOwner() == searchField) {
788            search();
789        }
790    }
791
792    public void handleEvent(Event event) {
793        if (event instanceof InputEvent) {
794            handleInputEvent((InputEvent) event);
795        }
796    }
797
798    private void handleInputEvent(InputEvent event) {
799        if (event instanceof KeyEvent) {
800            handleKeyEvent((KeyEvent) event);
801        }
802    }
803
804    public void handleEvent(Event event) {
805        if (event instanceof MouseEvent) {
806            handleMouseEvent((MouseEvent) event);
807        }
808    }
809
810    private void handleMouseEvent(MouseEvent event) {
811        if (event.isPopupTrigger()) {
812            handlePopupMenu();
813        }
814    }
815
816    private void handlePopupMenu() {
817        // HandlePopupMenu logic here
818    }
819
820    public void handleEvent(Event event) {
821        if (event instanceof KeyEvent) {
822            handleKeyEvent((KeyEvent) event);
823        }
824    }
825
826    private void handleKeyEvent(KeyEvent event) {
827        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
828            search();
829        }
830    }
831
832    public void handleEvent(Event event) {
833        if (event instanceof FocusEvent) {
834            handleFocusEvent((FocusEvent) event);
835        }
836    }
837
838    private void handleFocusEvent(FocusEvent event) {
839        if (event.getFocusOwner() == searchField) {
840            search();
841        }
842    }
843
844    public void handleEvent(Event event) {
845        if (event instanceof InputEvent) {
846            handleInputEvent((InputEvent) event);
847        }
848    }
849
850    private void handleInputEvent(InputEvent event) {
851        if (event instanceof KeyEvent) {
852            handleKeyEvent((KeyEvent) event);
853        }
854    }
855
856    public void handleEvent(Event event) {
857        if (event instanceof MouseEvent) {
858            handleMouseEvent((MouseEvent) event);
859        }
860    }
861
862    private void handleMouseEvent(MouseEvent event) {
863        if (event.isPopupTrigger()) {
864            handlePopupMenu();
865        }
866    }
867
868    private void handlePopupMenu() {
869        // HandlePopupMenu logic here
870    }
871
872    public void handleEvent(Event event) {
873        if (event instanceof KeyEvent) {
874            handleKeyEvent((KeyEvent) event);
875        }
876    }
877
878    private void handleKeyEvent(KeyEvent event) {
879        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
880            search();
881        }
882    }
883
884    public void handleEvent(Event event) {
885        if (event instanceof FocusEvent) {
886            handleFocusEvent((FocusEvent) event);
887        }
888    }
889
890    private void handleFocusEvent(FocusEvent event) {
891        if (event.getFocusOwner() == searchField) {
892            search();
893        }
894    }
895
896    public void handleEvent(Event event) {
897        if (event instanceof InputEvent) {
898            handleInputEvent((InputEvent) event);
899        }
900    }
901
902    private void handleInputEvent(InputEvent event) {
903        if (event instanceof KeyEvent) {
904            handleKeyEvent((KeyEvent) event);
905        }
906    }
907
908    public void handleEvent(Event event) {
909        if (event instanceof MouseEvent) {
910            handleMouseEvent((MouseEvent) event);
911        }
912    }
913
914    private void handleMouseEvent(MouseEvent event) {
915        if (event.isPopupTrigger()) {
916            handlePopupMenu();
917        }
918    }
919
920    private void handlePopupMenu() {
921        // HandlePopupMenu logic here
922    }
923
924    public void handleEvent(Event event) {
925        if (event instanceof KeyEvent) {
926            handleKeyEvent((KeyEvent) event);
927        }
928    }
929
930    private void handleKeyEvent(KeyEvent event) {
931        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
932            search();
933        }
934    }
935
936    public void handleEvent(Event event) {
937        if (event instanceof FocusEvent) {
938            handleFocusEvent((FocusEvent) event);
939        }
940    }
941
942    private void handleFocusEvent(FocusEvent event) {
943        if (event.getFocusOwner() == searchField) {
944            search();
945        }
946    }
947
948    public void handleEvent(Event event) {
949        if (event instanceof InputEvent) {
950            handleInputEvent((InputEvent) event);
951        }
952    }
953
954    private void handleInputEvent(InputEvent event) {
955        if (event instanceof KeyEvent) {
956            handleKeyEvent((KeyEvent) event);
957        }
958    }
959
960    public void handleEvent(Event event) {
961        if (event instanceof MouseEvent) {
962            handleMouseEvent((MouseEvent) event);
963        }
964    }
965
966    private void handleMouseEvent(MouseEvent event) {
967        if (event.isPopupTrigger()) {
968            handlePopupMenu();
969        }
970    }
971
972    private void handlePopupMenu() {
973        // HandlePopupMenu logic here
974    }
975
976    public void handleEvent(Event event) {
977        if (event instanceof KeyEvent) {
978            handleKeyEvent((KeyEvent) event);
979        }
980    }
981
982    private void handleKeyEvent(KeyEvent event) {
983        if (event.getKeyCode() == KeyEvent.VK_ENTER) {
984            search();
985        }
986    }
987
988    public void handleEvent(Event event) {
989        if (event instanceof FocusEvent) {
990            handleFocusEvent((FocusEvent) event);
991        }
992    }
993
994    private void handleFocusEvent(FocusEvent event) {
995        if (event.getFocusOwner() == searchField) {
996            search();
997        }
998    }
999
1000   public void handleEvent(Event event) {
1001      if (event instanceof InputEvent) {
1002          handleInputEvent((InputEvent) event);
1003      }
1004  }
1005
1006  private void handleInputEvent(InputEvent event) {
1007      if (event instanceof KeyEvent) {
1008          handleKeyEvent((KeyEvent) event);
1009      }
1010  }
1011
1012  public void handleEvent(Event event) {
1013      if (event instanceof MouseEvent) {
1014          handleMouseEvent((MouseEvent) event);
1015      }
1016  }
1017
1018  private void handleMouseEvent(MouseEvent event) {
1019      if (event.isPopupTrigger()) {
1020          handlePopupMenu();
1021      }
1022  }
1023
1024  private void handlePopupMenu() {
1025      // HandlePopupMenu logic here
1026  }
1027
1028  public void handleEvent(Event event) {
1029      if (event instanceof KeyEvent) {
1030          handleKeyEvent((KeyEvent) event);
1031      }
1032  }
1033
1034  private void handleKeyEvent(KeyEvent event) {
1035      if (event.getKeyCode() == KeyEvent.VK_ENTER) {
1036          search();
1037      }
1038  }
1039
1040  public void handleEvent(Event event) {
1041      if (event instanceof FocusEvent) {
1042          handleFocusEvent((FocusEvent) event);
1043      }
1044  }
1045
1046  private void handleFocusEvent(FocusEvent event) {
1047      if (event.getFocusOwner() == searchField) {
1048          search();
1049      }
1050  }
1051
1052  public void handleEvent(Event event) {
1053      if (event instanceof InputEvent) {
1054          handleInputEvent((InputEvent) event);
1055      }
1056  }
1057
1058  private void handleInputEvent(InputEvent event) {
1059      if (event instanceof KeyEvent) {
1060          handleKeyEvent((KeyEvent) event);
1061      }
1062  }
1063
1064  public void handleEvent(Event event) {
1065      if (event instanceof MouseEvent) {
1066          handleMouseEvent((MouseEvent) event);
1067      }
1068  }
1069
1070  private void handleMouseEvent(MouseEvent event) {
1071      if (event.isPopupTrigger()) {
1072          handlePopupMenu();
1073      }
1074  }
1075
1076  private void handlePopupMenu() {
1077      // HandlePopupMenu logic here
1078  }
1079
1080  public void handleEvent(Event event) {
1081      if (event instanceof KeyEvent) {
1082          handleKeyEvent((KeyEvent) event);
1083      }
1084  }
1085
1086  private void handleKeyEvent(KeyEvent event) {
1087      if (event.getKeyCode() == KeyEvent.VK_ENTER) {
1088          search();
1089      }
1090  }
1091
1092  public void handleEvent(Event event) {
1093      if (event instanceof FocusEvent) {
1094          handleFocusEvent((FocusEvent) event);
1095      }
1096  }
1097
1098  private void handleFocusEvent(FocusEvent event) {
1099      if (event.getFocusOwner() == searchField) {
1100          search();
1101      }
1102  }
1103
1104  public void handleEvent(Event event) {
1105      if (event instanceof InputEvent) {
1106          handleInputEvent((InputEvent) event);
1107      }
1108  }
1109
1110  private void handleInputEvent(InputEvent event) {
1111      if (event instanceof KeyEvent) {
1112          handleKeyEvent((KeyEvent) event);
1113      }
1114  }
1115
1116  public void handleEvent(Event event) {
1117      if (event instanceof MouseEvent) {
1118          handleMouseEvent((MouseEvent) event);
1119      }
1120  }
1121
1122  private void handleMouseEvent(MouseEvent event) {
1123      if (event.isPopupTrigger()) {
1124          handlePopupMenu();
1125      }
1126  }
1127
1128  private void handlePopupMenu() {
1129      // HandlePopupMenu logic here
1130  }
1131
1132  public void handleEvent(Event event) {
1133      if (event instanceof KeyEvent) {
1134          handleKeyEvent((KeyEvent) event);
1135      }
1136  }
1137
1138  private void handleKeyEvent(KeyEvent event) {
1139      if (event.getKeyCode() == KeyEvent.VK_ENTER) {
1140          search();
1141      }
1142  }
1143
1144  public void handleEvent(Event event) {
1145      if (event instanceof FocusEvent) {
1146          handleFocusEvent((FocusEvent) event);
1147      }
1148  }
1149
1150  private void handleFocusEvent(FocusEvent event) {
1151      if (event.getFocusOwner() == searchField) {
1152          search();
1153      }
1154  }
1155
1156  public void handleEvent(Event event) {
1157      if (event instanceof InputEvent) {
1158          handleInputEvent((InputEvent) event);
1159      }
1160  }
1161
1162  private void handleInputEvent(InputEvent event) {
1163      if (event instanceof KeyEvent) {
1164          handleKeyEvent((KeyEvent) event);
1165      }
1166  }
1167
1168  public void handleEvent(Event event) {
1169      if (event instanceof MouseEvent) {
1170          handleMouseEvent((MouseEvent) event);
1171      }
1172  }
1173
1174  private void handleMouseEvent(MouseEvent event) {
1175      if (event.isPopupTrigger()) {
1176          handlePopupMenu();
1177      }
1178  }
1179
1180  private void handlePopupMenu() {
1181      // HandlePopupMenu logic here
1182  }
1183
1184  public void handleEvent(Event event) {
1185      if (event instanceof KeyEvent) {
1186          handleKeyEvent((KeyEvent) event);
1187      }
1188  }
1189
1190  private void handleKeyEvent(KeyEvent event) {
1191      if (event.getKeyCode() == KeyEvent.VK_ENTER) {
1192          search();
1193      }
1194  }
1195
1196  public void handleEvent(Event event) {
1197      if (event instanceof FocusEvent) {
1198          handleFocusEvent((FocusEvent) event);
1199      }
1200  }
1201
1202  private void handleFocusEvent(FocusEvent event) {
1203      if (event.getFocusOwner() == searchField) {
1204          search();
1205      }
1206  }
1207
1208  public void handleEvent(Event event) {
1209      if (event instanceof InputEvent) {
1210          handleInputEvent((InputEvent) event);
1211      }
1212  }
1213
1214  private void handleInputEvent(InputEvent event) {
1215      if (event instanceof KeyEvent) {
1216          handleKeyEvent((KeyEvent) event);
1217      }
1218  }
1219
1220  public void handleEvent(Event event) {
1221      if (event instanceof MouseEvent) {
1222          handleMouseEvent((MouseEvent) event);
1223      }
1224  }
1225
1226  private void handleMouseEvent(MouseEvent event) {
1227      if (event.isPopupTrigger()) {
1228          handlePopupMenu();
1229      }
1230  }
1231
1232  private void handlePopupMenu() {
1233      // HandlePopupMenu logic here
1234  }
1235
1236  public void handleEvent(Event event) {
1237      if (event instanceof KeyEvent) {
1238          handleKeyEvent((KeyEvent) event);
1239      }
1240  }
1241
1242  private void handleKeyEvent(KeyEvent event) {
1243      if (event.getKeyCode() == KeyEvent.VK_ENTER) {
1244          search();
1245      }
1246  }
1247
1248  public void handleEvent(Event event) {
1249      if (event instanceof FocusEvent) {
1250          handleFocusEvent((FocusEvent) event);
1251      }
1252  }
1253
1254  private void handleFocusEvent(FocusEvent event) {
1255      if (event.getFocusOwner() == searchField) {
1256          search();
1257      }
1258  }
1259
1260  public void handleEvent(Event event) {
1261      if (event instanceof InputEvent) {
1262          handleInputEvent((InputEvent) event);
1263      }
1264  }
1265
1266  private void handleInputEvent(InputEvent event) {
1267      if (event instanceof KeyEvent) {
1268          handleKeyEvent((KeyEvent) event);
1269      }
1270  }
1271
1272  public void handleEvent(Event event) {
1273      if (event instanceof MouseEvent) {
1274          handleMouseEvent((MouseEvent) event);
1275      }
1276  }
1277
1278  private void handleMouseEvent(MouseEvent event) {
1279      if (event.isPopupTrigger()) {
1280          handlePopupMenu();
1281      }
1282  }
1283
1284  private void handlePopupMenu() {
1285      // HandlePopupMenu logic here
1286  }
1287
1288  public void handleEvent(Event event) {
1289      if (event instanceof KeyEvent) {
1290          handleKeyEvent((KeyEvent) event);
1291      }
1292  }
1293
1294  private void handleKeyEvent(KeyEvent event) {
1295      if (event.getKeyCode() == KeyEvent.VK_ENTER) {
1296          search();
1297      }
1298  }
1299
1300  public void handleEvent(Event event) {
1301      if (event instanceof FocusEvent) {
1302          handleFocusEvent((FocusEvent) event);
1303      }
1304  }
1305
1306  private void handleFocusEvent(FocusEvent event) {
1307      if (event.getFocusOwner() == searchField) {
1308          search();
1309      }
1310  }
1311
1
```

```
17
18     public CityInfo() { 2 usages
19         setTitle("City Information");
20         setSize( width: 850, height: 550);
21         setLocationRelativeTo(null);
22         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
23         setLayout(new BorderLayout());
24
25         // Header Panel (Title)
26         JPanel headerPanel = new JPanel();
27         headerPanel.setBackground(new Color(r: 30, g: 144, b: 255)); // Dodger Blue
28         headerPanel.setPreferredSize(new Dimension(getWidth(), height: 60));
29
30         cityNameLabel = new JLabel(text: "CITY INFORMATION", SwingConstants.CENTER);
31         cityNameLabel.setFont(new Font(name: "Arial", Font.BOLD, size: 26));
32         cityNameLabel.setForeground(Color.WHITE);
33         headerPanel.add(cityNameLabel);
34         add(headerPanel, BorderLayout.NORTH);
35
36         // Main Content Panel
37         JPanel mainPanel = new JPanel(new BorderLayout());
38         mainPanel.setBorder(new EmptyBorder(top: 20, left: 20, bottom: 20, right: 20));
39         add(mainPanel);
40
41         // Info Pane (City Details)
42         infoPane = new JTextPane();
43         infoPane.setContentType("text/html");
44         infoPane.setEditable(false);
45         infoPane.setFont(new Font(name: "Arial", Font.PLAIN, size: 16));
46         infoPane.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY, thickness: 1));
47         JScrollPane scrollPane = new JScrollPane(infoPane);
48         mainPanel.add(scrollPane, BorderLayout.CENTER);
49
50         // Search Panel
51         JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, hgap: 10, vgap: 10));
52         searchPanel.setBackground(new Color(r: 245, g: 245, b: 245)); // Light Gray Background
53
54         searchField = new JTextField(columns: 20);
55         searchField.setFont(new Font(name: "Arial", Font.PLAIN, size: 14));
56         searchField.setBorder(BorderFactory.createCompoundBorder(
57             BorderFactory.createLineBorder(new Color(r: 100, g: 149, b: 237), thickness: 2), // Cornflower Blue Border
58             BorderFactory.createEmptyBorder(top: 5, left: 10, bottom: 5, right: 10)
59         ));
60
61         JButton searchButton = new JButton(text: "Search");
62         searchButton.setFont(new Font(name: "Arial", Font.BOLD, size: 14));
63         searchButton.setBackground(new Color(r: 0, g: 123, b: 255));
64         searchButton.setForeground(Color.WHITE);
65         searchButton.setFocusPainted(false);
66         searchButton.setBorder(BorderFactory.createEmptyBorder(top: 8, left: 15, bottom: 8, right: 15));
67
68         searchPanel.add(new JLabel(text: "Enter City: "));
69         searchPanel.add(searchField);
70         searchPanel.add(searchButton);
71         mainPanel.add(searchPanel, BorderLayout.SOUTH);
72
73         // Search Action
74         searchButton.addActionListener(new ActionListener() {
75             @Override
76             public void actionPerformed(ActionEvent e) {
77                 searchCity();
78             }
79         });

```

```

80 }
81
82     private void searchCity() { 1 usage
83         String cityName = searchField.getText().trim();
84         if (cityName.isEmpty()) {
85             JOptionPane.showMessageDialog( parentComponent: this, message: "Please enter a city name.", title: "Error", JOptionPane.ERROR_MESSAGE);
86             return;
87         }
88
89         try (Connection conn = DBConnection.getConnection()) {
90             assert conn != null;
91             try (PreparedStatement stmt = conn.prepareStatement( sql: "SELECT description, population, emergency_contacts FROM cities WHERE name = ?")) {
92
93                 stmt.setString( parameterIndex: 1, cityName);
94                 ResultSet rs = stmt.executeQuery();
95
96                 if (rs.next()) {
97                     String description = rs.getString( columnLabel: "description");
98                     int population = rs.getInt( columnLabel: "population");
99                     String emergencyContacts = rs.getString( columnLabel: "emergency_contacts");
100
101                     // Update Title & Text
102                     cityNameLabel.setText(cityName.toUpperCase());
103                     String displayText = "<html><div style='text-align:left;'>" +
104                         + "<h2 style='color:#1E90FF;'>Population: " + population + "</h2>" +
105                         + "<p style='font-size:16px;'>" + description + "</p>" +
106                         + "<div style='background-color:#FFD700; padding:10px; border-radius:5px;'>" +
107                         + "<b>Emergency Contacts:</b> " + emergencyContacts + "</div>" +
108                         + "</div></html>";
109
110                     infoPane.setText(displayText);
111                 } else {
112                     infoPane.setText("<html><h3 style='color:red;'>No information found for: " + cityName + "</h3></html>");
113                 }
114             }
115         } catch (SQLException ex) {
116             JOptionPane.showMessageDialog( parentComponent: this, message: "Database error occurred. Please try again.", JOptionPane.ERROR_MESSAGE);
117         }
118     }
119 }
120
121 ▶ public static void main(String[] args) {
122     SwingUtilities.invokeLater(() -> new CityInfo().setVisible(true));
123 }
124 }
125

```

Tourism.java x 0.1 Kbps

```

1 package com.smartcity;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7
8 ▶ public class Tourism extends JFrame {
9     public Tourism() { 2 usages
10         setTitle("Tourism - Smart City");
11         setSize( width: 600, height: 400);
12         setLocationRelativeTo(null);
13         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
14         setLayout(new BorderLayout());
15

```

```
16     // Header Panel
17     JPanel headerPanel = new JPanel();
18     headerPanel.setBackground(new Color(r: 0, g: 102, b: 204));
19     JLabel titleLabel = new JLabel(text: "Explore Tourism");
20     titleLabel.setFont(new Font(name: "Arial", Font.BOLD, size: 26));
21     titleLabel.setForeground(Color.WHITE);
22     headerPanel.add(titleLabel);
23     add(headerPanel, BorderLayout.NORTH);
24
25     // Center Panel with Buttons
26     JPanel centerPanel = new JPanel();
27     centerPanel.setLayout(new GridLayout(rows: 3, cols: 1, hgap: 10, vgap: 10));
28     centerPanel.setBorder(BorderFactory.createEmptyBorder(top: 30, left: 50, bottom: 30, right: 50));
29
30     JButton btnHotels = createStyledButton(text: "Hotels");
31     JButton btnPlaces = createStyledButton(text: "Places to Visit");
32     JButton btnRestaurants = createStyledButton(text: "Restaurants");
33
34     centerPanel.add(btnHotels);
35     centerPanel.add(btnPlaces);
36     centerPanel.add(btnRestaurants);
37
38     add(centerPanel, BorderLayout.CENTER);
39
40     // Button Actions
41     btnHotels.addActionListener(ActionEvent e -> new Hotels().setVisible(true));
42     btnPlaces.addActionListener(ActionEvent e -> new TouristPlaces().setVisible(true));
43     btnRestaurants.addActionListener(ActionEvent e -> new Restaurants().setVisible(true));
44 }
45
46 // Method to create styled buttons
47 @
48 private JButton createStyledButton(String text) { 3 usages
49     JButton button = new JButton(text);
50     button.setFont(new Font(name: "Arial", Font.BOLD, size: 16));
51     button.setBackground(new Color(r: 0, g: 153, b: 255));
52     button.setForeground(Color.WHITE);
53     button.setFocusPainted(false);
54     button.setBorder(BorderFactory.createEmptyBorder(top: 10, left: 20, bottom: 10, right: 20));
55
56     // Hover Effect
57     button.addMouseListener(new java.awt.event.MouseAdapter() {
58         public void mouseEntered(java.awt.event.MouseEvent evt) {
59             button.setBackground(new Color(r: 0, g: 102, b: 204));
60         }
61         public void mouseExited(java.awt.event.MouseEvent evt) {
62             button.setBackground(new Color(r: 0, g: 153, b: 255));
63         }
64     });
65
66     return button;
67 }
68 ▶ public static void main(String[] args) {
69     SwingUtilities.invokeLater(() -> new Tourism().setVisible(true));
70 }
71 }
72 }
```

```
Hotels.java × 83.7 Kbps
```

```
1 package com.smartcity;
2
3 import javax.swing.*;
4 import javax.swing.table.DefaultTableModel;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11
12 public class Hotels extends JFrame {
13     private final JTextField searchField; 3 usages
14     private final DefaultTableModel model; 5 usages
15
16     public Hotels() { 2 usages
17         setTitle("Hotel Search");
18         setSize( width: 700, height: 400);
19         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
20         setLocationRelativeTo(null);
21
22         // Panel for Search
23         JPanel searchPanel = new JPanel();
24         searchPanel.add(new JLabel( text: "Enter Location: "));
25         searchField = new JTextField( columns: 15);
26         searchPanel.add(searchField);
27         JButton searchButton = new JButton( text: "Search");
28         searchPanel.add(searchButton);
29
30         // Table Model Setup
31         model = new DefaultTableModel();
32         model.setColumnIdentifiers(new String[]{"Hotel Name", "Location", "Contact", "Price Range", "Rating"});
33         JTable table = new JTable(model);
34         JScrollPane scrollPane = new JScrollPane(table);
35
36         // Layout
37         setLayout(new BorderLayout());
38         add(searchPanel, BorderLayout.NORTH);
39         add(scrollPane, BorderLayout.CENTER);
40
41         // Action Listener for Search Button
42         searchButton.addActionListener(new ActionListener() {
43             @Override
44             public void actionPerformed(ActionEvent e) {
45                 searchHotels();
46             }
47         });
48     }
49
50     private void searchHotels() { 1 usage
51         String location = searchField.getText().trim();
52
53         if (location.isEmpty()) {
54             JOptionPane.showMessageDialog( parentComponent: this, message: "Please enter a location!", title: "Warning", JOptionPane.ERROR_MESSAGE);
55             return;
56         }
57
58         try {
59             Connection con = DBConnection.getConnection();
60             String sql = "SELECT hotel_name, location, contact, price_range, rating FROM hotels WHERE location LIKE ?";
61             assert con != null;
62             PreparedStatement pst = con.prepareStatement(sql);
63             pst.setString( parameterIndex: 1, x: "%" + location + "%");
64
65             ResultSet rs = pst.executeQuery();
66             model.setRowCount(0); // Clear previous results
67         } catch (SQLException ex) {
68             ex.printStackTrace();
69         }
70     }
71 }
```

```

67
68     boolean found = false;
69     while (rs.next()) {
70         model.addRow(new Object[]{
71             rs.getString( columnLabel: "hotel_name"),
72             rs.getString( columnLabel: "location"),
73             rs.getString( columnLabel: "contact"),
74             rs.getString( columnLabel: "price_range"),
75             rs.getFloat( columnLabel: "rating")
76         });
77         found = true;
78     }
79
80     if (!found) {
81         JOptionPane.showMessageDialog( parentComponent: this, message: "No hotels found in this location.", title: "Search Results")
82     }
83
84     rs.close();
85     pst.close();
86     con.close();
87 } catch (Exception ex) {
88     JOptionPane.showMessageDialog( parentComponent: this, message: "Error fetching data: " + ex.getMessage(), title: "Search Results")
89 }
90 }
91
92 public static void main(String[] args) {
93     SwingUtilities.invokeLater(() -> new Hotels().setVisible(true));
94 }
95 }
96

```

TouristPlaces.java

```

1 package com.smartcity;
2
3 import javax.swing.*;
4 import javax.swing.table.DefaultTableModel;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11
12 public class TouristPlaces extends JFrame {
13     private final JTextField searchField; 3 usages
14     private final DefaultTableModel model; 5 usages
15
16     public TouristPlaces() { 2 usages
17         setTitle("Tourist Places");
18         setSize( width: 800, height: 400);
19         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
20         setLocationRelativeTo(null);
21
22         // Search Panel
23         JPanel searchPanel = new JPanel();
24         searchPanel.add(new JLabel( text: "Enter Location: "));
25         searchField = new JTextField( columns: 15);
26         searchPanel.add(searchField);
27         JButton searchButton = new JButton( text: "Search");
28         searchPanel.add(searchButton);
29
30         // Table Setup
31         model = new DefaultTableModel();
32         model.setColumnIdentifiers(new String[]{"Place Name", "Location", "Description", "Entry Fee", "Opening Hours"});
33         JTable table = new JTable(model);
34         JScrollPane scrollPane = new JScrollPane(table);

```

```
35
36     // Layout
37     setLayout(new BorderLayout());
38     add(searchPanel, BorderLayout.NORTH);
39     add(scrollPane, BorderLayout.CENTER);
40
41     // Search Button Action
42     searchButton.addActionListener(new ActionListener() {
43         @Override
44         public void actionPerformed(ActionEvent e) {
45             searchTouristPlaces();
46         }
47     });
48 }
49
50 private void searchTouristPlaces() { 1 usage
51     String location = searchField.getText().trim();
52
53     if (location.isEmpty()) {
54         JOptionPane.showMessageDialog( parentComponent: this, message: "Please enter a location!", title: "Warning", JO
55         return;
56     }
57
58     try {
59         Connection con = DBConnection.getConnection();
60         String sql = "SELECT place_name, location, description, entry_fee, opening_hours FROM tourism WHERE locat:
61         assert con != null;
62         PreparedStatement pst = con.prepareStatement(sql);
63         pst.setString( parameterIndex: 1, x: "%" + location + "%");
64
65         ResultSet rs = pst.executeQuery();
66         model.setRowCount(0); // Clear previous results
67
68         boolean found = false;
69         while (rs.next()) {
70             model.addRow(new Object[]{
71                 rs.getString( columnLabel: "place_name"),
72                 rs.getString( columnLabel: "location"),
73                 rs.getString( columnLabel: "description"),
74                 rs.getDouble( columnLabel: "entry_fee"),
75                 rs.getString( columnLabel: "opening_hours")
76             });
77             found = true;
78         }
79
80         if (!found) {
81             JOptionPane.showMessageDialog( parentComponent: this, message: "No tourist places found in this location.
82         }
83
84         rs.close();
85         pst.close();
86         con.close();
87     } catch (Exception ex) {
88         JOptionPane.showMessageDialog( parentComponent: this, message: "Error fetching data: " + ex.getMessage(), titl
89     }
90 }
91
92 D public static void main(String[] args) {
93     SwingUtilities.invokeLater(() -> new TouristPlaces().setVisible(true));
94 }
95 }
96 }
```

Restaurants.java 58.4 KBp

```
1 package com.smartcity;
2
3 import javax.swing.*;
4 import javax.swing.table.DefaultTableModel;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11
12 public class Restaurants extends JFrame {
13     private final JTextField searchField; 3 usages
14     private final DefaultTableModel model; 5 usages
15
16     public Restaurants() { 2 usages
17         setTitle("Restaurants");
18         setSize( width: 800, height: 400);
19         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
20         setLocationRelativeTo(null);
21
22         // Search Panel
23         JPanel searchPanel = new JPanel();
24         searchPanel.add(new JLabel( text: "Enter Location: "));
25         searchField = new JTextField( columns: 15);
26         searchPanel.add(searchField);
27         JButton searchButton = new JButton( text: "Search");
28         searchPanel.add(searchButton);
29
30         // Table Setup
31         model = new DefaultTableModel();
32         model.setColumnIdentifiers(new String[]{"Restaurant Name", "Location", "Cuisine", "Price Range", "Contact"});
33         JTable table = new JTable(model);
34         JScrollPane scrollPane = new JScrollPane(table);
35
36         // Layout
37         setLayout(new BorderLayout());
38         add(searchPanel, BorderLayout.NORTH);
39         add(scrollPane, BorderLayout.CENTER);
40
41         // Search Button Action
42         searchButton.addActionListener(new ActionListener() {
43             @Override
44             public void actionPerformed(ActionEvent e) {
45                 searchRestaurants();
46             }
47         });
48     }
49
50     private void searchRestaurants() { 1 usage
51         String location = searchField.getText().trim();
52
53         if (location.isEmpty()) {
54             JOptionPane.showMessageDialog( parentComponent: this, message: "Please enter a location!", title: "Warning", J
55             return;
56         }
57
58         try {
59             Connection con = DBConnection.getConnection();
60             String sql = "SELECT restaurant_name, location, cuisine, price_range, contact FROM restaurants WHERE loca
61             PreparedStatement pst = con.prepareStatement(sql);
62             pst.setString( parameterIndex: 1, x: "%" + location + "%");
63
64             ResultSet rs = pst.executeQuery();
65             model.setRowCount(0); // Clear previous results
66         }
67     }
68 }
```

```

67         boolean found = false;
68         while (rs.next()) {
69             model.addRow(new Object[]{
70                 rs.getString( columnLabel: "restaurant_name"),
71                 rs.getString( columnLabel: "location"),
72                 rs.getString( columnLabel: "cuisine"),
73                 rs.getString( columnLabel: "price_range"),
74                 rs.getString( columnLabel: "contact")
75             });
76             found = true;
77         }
78
79         if (!found) {
80             JOptionPane.showMessageDialog( parentComponent: this, message: "No restaurants found in this location.", title: "Information")
81         }
82
83         rs.close();
84         pst.close();
85         con.close();
86     } catch (Exception ex) {
87         JOptionPane.showMessageDialog( parentComponent: this, message: "Error fetching data: " + ex.getMessage(), title: "Error")
88     }
89 }
90
91 ▶ public static void main(String[] args) {
92     SwingUtilities.invokeLater(() -> new Restaurants().setVisible(true));
93 }
94 }
95

```

Transportation.java × 0.5 Kbps

```

1 package com.smartcity;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7 import java.sql.Connection;
8 import java.sql.PreparedStatement;
9 import java.sql.ResultSet;
10 import java.util.Objects;
11
12 ▶ public class Transportation extends JFrame {
13     private final JComboBox<String> pickupField; 4 usages
14     private final JComboBox<String> destinationField; 4 usages
15     private final JPanel transportPanel; 8 usages
16
17     public Transportation() { 2 usages
18         setTitle("Find Your Ride");
19         setSize( width: 600, height: 500);
20         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
21         setLocationRelativeTo(null);
22         setLayout(new BorderLayout());
23
24         // Title Label
25         JLabel titleLabel = new JLabel( text: "Find Your Transportation", SwingConstants.CENTER);
26         titleLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 24));
27         titleLabel.setBorder(BorderFactory.createEmptyBorder( top: 10, left: 0, bottom: 10, right: 0));
28         add(titleLabel, BorderLayout.NORTH);
29
30         // Input Panel
31         JPanel inputPanel = new JPanel(new GridBagLayout());
32         inputPanel.setBorder(BorderFactory.createEmptyBorder( top: 10, left: 20, bottom: 10, right: 20));
33         GridBagConstraints gbc = new GridBagConstraints();
34         gbc.gridx = 0;

```

```

35         gbc.gridx = 0;
36         gbc.anchor = GridBagConstraints.WEST;
37         gbc.insets = new Insets( top: 5, left: 5, bottom: 5, right: 5);
38
39         inputPanel.add(new JLabel( text: "📍 Pickup Location:"), gbc);
40         gbc.gridx = 1;
41         pickupField = new JComboBox<>(new String[]{"Select", "Central Station", "Airport", "Metro Station", "Railway"});
42         pickupField.setPreferredSize(new Dimension( width: 200, height: 30));
43         inputPanel.add(pickupField, gbc);
44
45         gbc.gridx = 0;
46         gbc.gridy = 1;
47         inputPanel.add(new JLabel( text: "📍 Destination:"), gbc);
48         gbc.gridx = 1;
49         destinationField = new JComboBox<>(new String[]{"Select", "Downtown", "City Center", "TechVille", "Springfield"});
50         destinationField.setPreferredSize(new Dimension( width: 200, height: 30));
51         inputPanel.add(destinationField, gbc);
52
53         // Find Ride Button
54         JButton findRideBtn = new JButton( text: "Find Ride");
55         findRideBtn.setFont(new Font( name: "Arial", Font.BOLD, size: 16));
56         findRideBtn.setBackground(new Color( r: 30, g: 144, b: 255)); // Blue color
57         findRideBtn.setForeground(Color.WHITE);
58         findRideBtn.setFocusPainted(false);
59         findRideBtn.setPreferredSize(new Dimension( width: 200, height: 40));
60         findRideBtn.setBorder(BorderFactory.createLineBorder(Color.BLACK, thickness: 2));
61
62         gbc.gridx = 1;
63         gbc.gridy = 2;
64         inputPanel.add(findRideBtn, gbc);
65
66         add(inputPanel, BorderLayout.CENTER);
67
68         // Transport Panel
69         transportPanel = new JPanel();
70         transportPanel.setLayout(new GridLayout( rows: 0, cols: 1, hgap: 10, vgap: 10));
71         transportPanel.setBorder(BorderFactory.createTitledBorder("🚗 Available Transport Options"));
72         JScrollPane scrollPane = new JScrollPane(transportPanel);
73         scrollPane.setPreferredSize(new Dimension( width: 580, height: 200));
74         add(scrollPane, BorderLayout.SOUTH);
75
76         // Button Action
77         findRideBtn.addActionListener(new ActionListener() {
78             @Override
79             public void actionPerformed(ActionEvent e) {
80                 String pickup = (String) pickupField.getSelectedItem();
81                 String destination = (String) destinationField.getSelectedItem();
82                 assert pickup != null;
83                 if (pickup.equals("Select") || Objects.equals(destination, b: "Select")) {
84                     JOptionPane.showMessageDialog( parentComponent: null, message: "Please select both pickup and destination");
85                 } else {
86                     fetchTransportOptions(pickup, destination);
87                 }
88             }
89         });
90
91         setVisible(true);
92     }
93
94     private void fetchTransportOptions(String pickup, String destination) { 1 usage
95         transportPanel.removeAll();
96         try {
97             Connection conn = DBConnection.getConnection();
98             String query = "SELECT * FROM transportation WHERE pickup_point = ? AND destination = ?";

```

```
12  public class Transportation extends JFrame {
13
14      private void fetchTransportOptions(String pickup, String destination) { 1 usage
15
16          assert conn != null;
17          PreparedStatement pstmt = conn.prepareStatement(query);
18          pstmt.setString( parameterIndex: 1, pickup);
19          pstmt.setString( parameterIndex: 2, destination);
20          ResultSet rs = pstmt.executeQuery();
21
22
23          boolean dataFound = false;
24
25
26          while (rs.next()) {
27              dataFound = true;
28              String transportType = rs.getString( columnLabel: "transport_type");
29              double fare = rs.getDouble( columnLabel: "fare");
30              String bookingMethod = rs.getString( columnLabel: "booking_method");
31
32
33              // Transport Option Card
34              JPanel transportCard = new JPanel(new BorderLayout());
35              transportCard.setBorder(BorderFactory.createLineBorder(Color.GRAY, thickness: 1));
36              transportCard.setBackground(new Color(r: 245, g: 245, b: 245));
37              transportCard.setPreferredSize(new Dimension(width: 550, height: 50));
38
39
40              JLabel transportLabel = new JLabel(text: "🚗 " + transportType, SwingConstants.LEFT);
41              transportLabel.setFont(new Font(name: "Arial", Font.BOLD, size: 14));
42
43              JLabel fareLabel = new JLabel(text: "📍 Fare: $" + fare, SwingConstants.CENTER);
44              JLabel bookingLabel = new JLabel(text: "🕒 Booking: " + bookingMethod, SwingConstants.RIGHT);
45
46
47              transportCard.add(transportLabel, BorderLayout.WEST);
48              transportCard.add(fareLabel, BorderLayout.CENTER);
49              transportCard.add(bookingLabel, BorderLayout.EAST);
50
51
52              transportPanel.add(transportCard);
53          }
54
55
56          if (!dataFound) {
57              JOptionPane.showMessageDialog(parentComponent: null, message: "No transport options available for the s");
58          }
59
60
61          transportPanel.revalidate();
62          transportPanel.repaint();
63
64
65          conn.close();
66      } catch (Exception ex) {
67          JOptionPane.showMessageDialog(parentComponent: null, message: "Error fetching data: " + ex.getMessage());
68      }
69
70  }
71
72
73  public static void main(String[] args) {
74      new Transportation();
75  }
76
77  }
```

```
1 package com.smartcity;
2
3 import javax.swing.*;
4 import javax.swing.table.DefaultTableModel;
5 import java.awt.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.sql.*;
9
10 public class ShoppingMalls extends JFrame {
11     private final JTextField searchField; 5 usages
12     private final DefaultTableModel tableModel; 4 usages
13
14     public ShoppingMalls() { 2 usages
15         setTitle("Shopping Malls");
16         setSize( width: 700, height: 450);
17         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
18         setLayout(new BorderLayout());
19         getContentPane().setBackground(new Color( r: 240, g: 248, b: 255)); // Light blue background
20
21         // Search Panel
22         JPanel searchPanel = new JPanel();
23         searchPanel.setBackground(new Color( r: 200, g: 220, b: 240));
24         searchPanel.setBorder(BorderFactory.createEmptyBorder( top: 10, left: 10, bottom: 10, right: 10));
25         searchPanel.setLayout(new FlowLayout());
26
27         JLabel searchLabel = new JLabel( text: "Search by Location:");
28         searchLabel.setFont(new Font( name: "Arial", Font.BOLD, size: 14));
29         searchField = new JTextField( columns: 15);
30         searchField.setFont(new Font( name: "Arial", Font.PLAIN, size: 14));
31         searchField.setBorder(BorderFactory.createLineBorder(Color.GRAY, thickness: 1));
32
33         JButton searchButton = new JButton( text: "Search");
34         searchButton.setFont(new Font( name: "Arial", Font.BOLD, size: 14));
35         searchButton.setBackground(new Color( r: 70, g: 130, b: 180)); // Steel Blue
36         searchButton.setForeground(Color.WHITE);
37         searchButton.setFocusPainted(false);
38         searchButton.setBorder(BorderFactory.createEmptyBorder( top: 5, left: 10, bottom: 5, right: 10));
39
40         searchPanel.add(searchLabel);
41         searchPanel.add(searchField);
42         searchPanel.add(searchButton);
43         add(searchPanel, BorderLayout.NORTH);
44
45         // Table Panel
46         tableModel = new DefaultTableModel(new String[]{"Mall Name", "Location", "Opening Time", "Closing Time"}, 1);
47         JTable table = new JTable(tableModel);
48         table.setRowHeight(25);
49         table.getTableHeader().setFont(new Font( name: "Arial", Font.BOLD, size: 14));
50         table.getTableHeader().setBackground(new Color( r: 70, g: 130, b: 180));
51         table.getTableHeader().setForeground(Color.WHITE);
52         table.setFont(new Font( name: "Arial", Font.PLAIN, size: 14));
53         table.setSelectionBackground(new Color( r: 173, g: 216, b: 230));
54
55         add(new JScrollPane(table), BorderLayout.CENTER);
56
57         // Search action
58         searchButton.addActionListener(new ActionListener() {
59             @Override
60             public void actionPerformed(ActionEvent e) {
61                 String location = searchField.getText().trim();
62                 searchMalls(location);
63             }
64         });
65
66         setVisible(true);
67 }
```

```

66     setVisible(true);
67 }
68
69     private void searchMalls(String location) { 1 usage
70         tableModel.setRowCount(0); // Clear previous results
71
72         try (Connection conn = DBConnection.getConnection()) {
73             assert conn != null;
74             try (PreparedStatement stmt = conn.prepareStatement( sql: "SELECT mall_name, location, opening_time, closing_time FROM malls WHERE location LIKE ?")) {
75
76                 stmt.setString( parameterIndex: 1, x: "%" + location + "%");
77                 ResultSet rs = stmt.executeQuery();
78
79                 while (rs.next()) {
80                     tableModel.addRow(new Object[]{rs.getString( columnLabel: "mall_name"), rs.getString( columnLabel: "location"), rs.getString( columnLabel: "opening_time"), rs.getString( columnLabel: "closing_time")});
81                 }
82             } catch (SQLException ex) {
83                 JOptionPane.showMessageDialog( parentComponent: this, message: "Database Error: " + ex.getMessage(), title: "Error");
84             }
85         }
86     }
87
88 D     public static void main(String[] args) {
89         new ShoppingMalls();
90     }
91 }
92

```

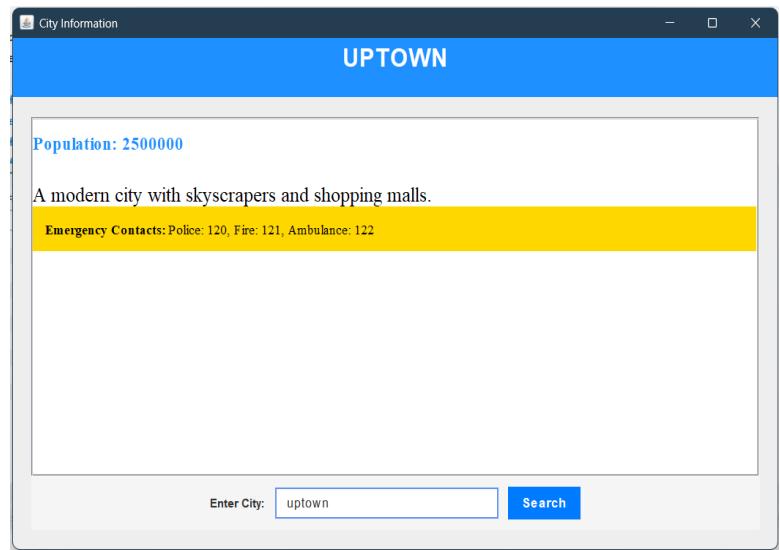
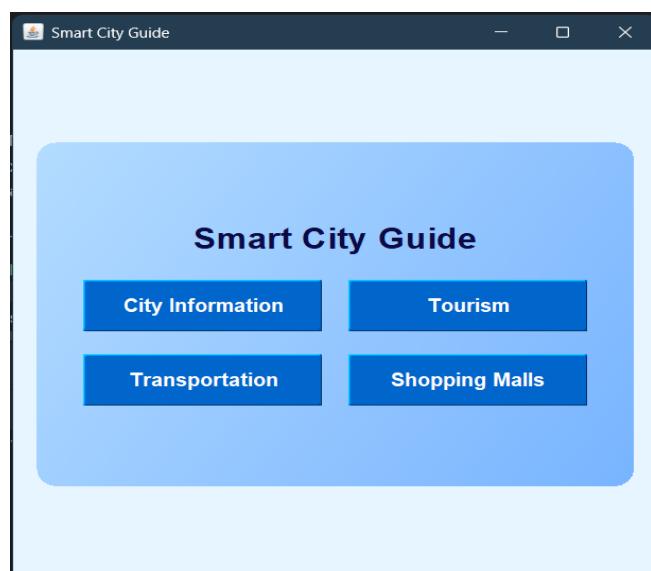
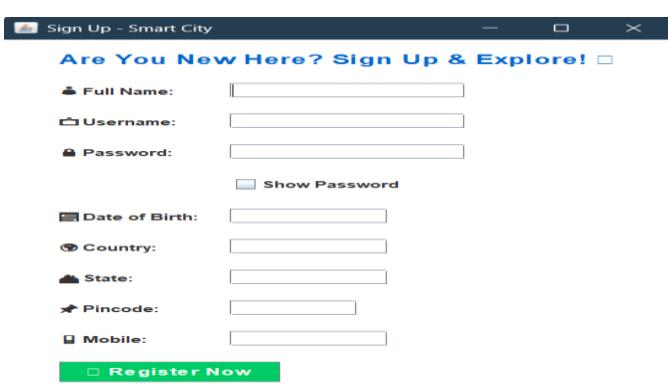
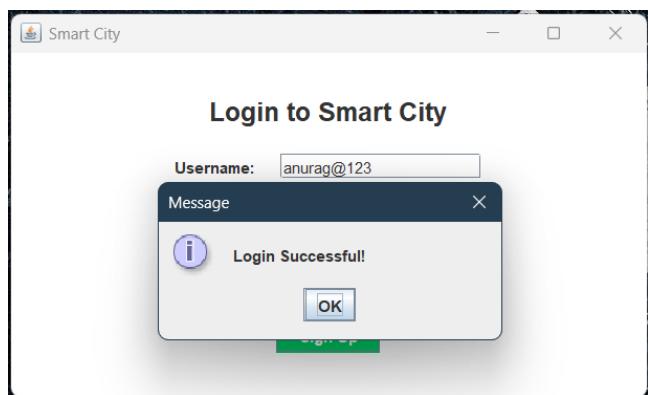
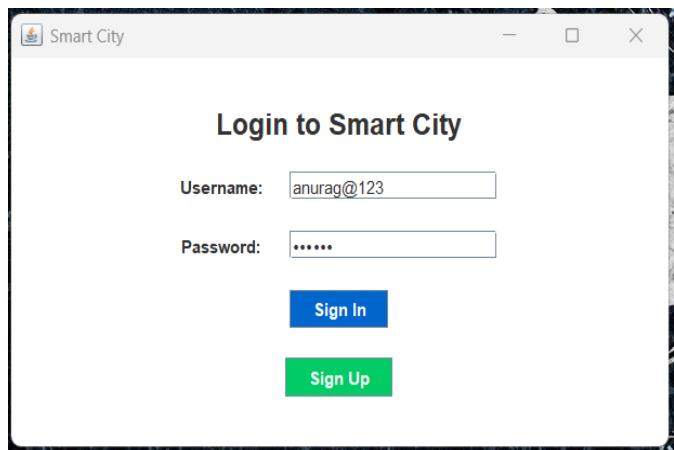
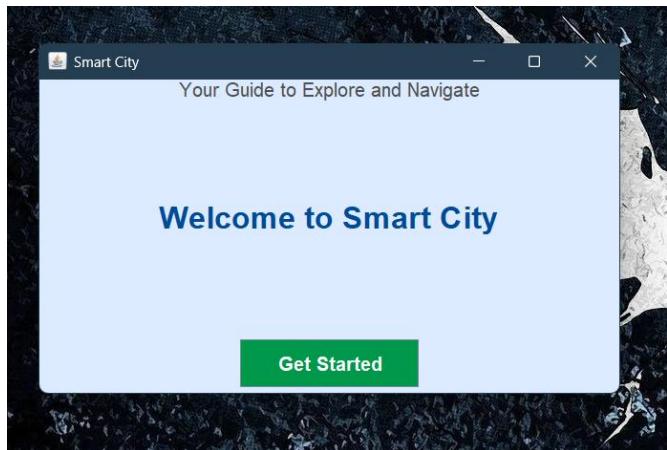
DBConnection.java × 1.24 KB ↑ 0.5 KByte

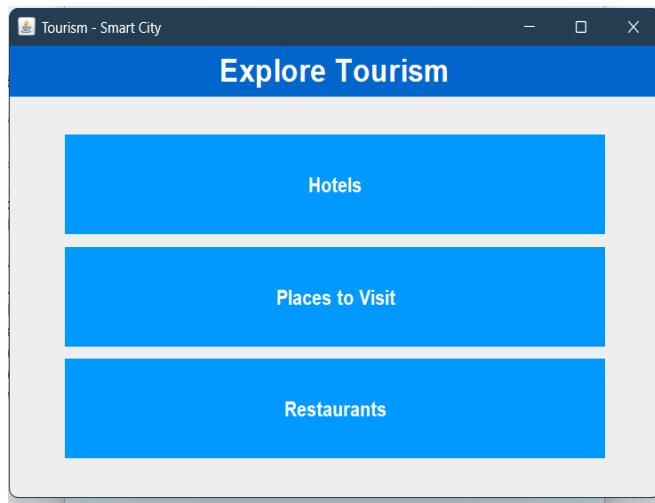
```

1 package com.smartcity;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import java.util.logging.Level;
7 import java.util.logging.Logger;
8
9 public class DBConnection { 9 usages
10     private static final String URL = "jdbc:mysql://localhost:3306/smartzity"; 1 usage
11     private static final String USER = "root"; 1 usage
12     private static final String PASSWORD = "614599"; 1 usage
13
14     private static final Logger LOGGER = Logger.getLogger(DBConnection.class.getName()); 1 usage
15
16 @
17     public static Connection getConnection() { 8 usages
18         try {
19             // Load MySQL JDBC Driver
20             Class.forName( className: "com.mysql.cj.jdbc.Driver");
21
22             // Establish connection
23             return DriverManager.getConnection(URL, USER, PASSWORD);
24         } catch (ClassNotFoundException | SQLException e) {
25             LOGGER.log(Level.SEVERE, msg: "Database connection failed!", e);
26             return null; // Return null if connection fails
27         }
28     }
29

```

8. Output Screenshots





Hotel Search				
Enter Location: <input type="text" value="Downtown"/> <input type="button" value="Search"/>				
Hotel Name	Location	Contact	Price Range	Rating
Grand Palace Hotel	Downtown	1234567890	\$100-\$300	4.5

Enter Location: <input type="text" value="downtown"/> <input type="button" value="Search"/>				
Place Name	Location	Description	Entry Fee	Opening Hours
Central Park	Downtown Metropolis	A large green park with lakes	\$5.0	6:00 AM - 8:00 PM

Enter Location: <input type="text" value="city center"/> <input type="button" value="Search"/>				
Restaurant Name	Location	Cuisine	Price Range	Contact
The Royal Dine	City Center	Indian	\$20-\$100	9879879876
Luxury Dining	City Center	Gourmet	\$50-\$200	8908908901

Search by Location: <input type="text" value="uptown"/> <input type="button" value="Search"/>			
Mall Name	Location	Opening Time	Closing Time
Uptown Shopping Hub	Uptown	09:30:00	23:00:00

Find Your Transportation

Pickup Location:

Destination:

Available Transport Options

<input type="checkbox"/> Bus	Fare: \$2.5	Booking: Cash
------------------------------	-------------	---------------

9. Project Summary

The Smart City Application successfully integrates city information, tourism, transportation, and shopping services into a single platform. The application is built using Java Swing and MySQL, allowing users to search for places dynamically. The enhanced UI design, especially in the Transportation section, provides an experience similar to modern ride-booking apps.

10. Important Learnings

During the development of this project, I gained valuable insights into:

- Java Swing UI Development
- MySQL Database Integration
- JDBC for Data Fetching
- Object-Oriented Programming (OOP) Concepts
- UI/UX Enhancement Techniques
- Designing an Interactive Transportation System

Conclusion

The **Smart City Guide Application** successfully provides an interactive platform for users to explore essential city services, including **tourism, transportation, hotels, restaurants, and shopping malls**. Through an intuitive Java Swing interface and a well-structured MySQL database, the application ensures smooth navigation and efficient data retrieval. Users can search for locations, view important details, and make informed decisions based on real-time information.

This project not only enhanced our understanding of **Object-Oriented Programming (OOP) in Java** but also provided hands-on experience in **database integration, UI design, and application development**. Future improvements may include **real-time booking options, user authentication, and advanced search features** to further enhance user engagement. Overall, this project serves as a solid foundation for building smart city solutions and improving urban navigation experiences.