

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.Scanner;
7
8 public class DatabaseManager {
9     private static final String DB_URL = "jdbc:mysql://localhost:3306/mydatabase";
10    private static final String USER = "user";
11    private static final String PASS = "password";
12
13    public static void main(String[] args) {
14        Scanner scanner = new Scanner(System.in);
15        int option;
16        do {
17            System.out.println("\n--- Database Management System ---");
18            System.out.println("1. Add User");
19            System.out.println("2. Update User");
20            System.out.println("3. Delete User");
21            System.out.println("4. View All Users");
22            System.out.println("5. Add Role");
23            System.out.println("6. Assign Role to User");
24            System.out.println("7. Remove Role from User");
25            System.out.println("8. List All Roles");
26            System.out.println("9. Search User by Name");
27            System.out.println("10. Search User by Email");
28            System.out.println("11. Exit");
29            System.out.print("Choose an option: ");
30            option = scanner.nextInt();
31            switch (option) {
32                case 1:
33                    addUser(scanner);
34                    break;
35                case 2:
36                    updateUser(scanner);
37                    break;
38                case 3:
39                    deleteUser(scanner);
40                    break;
41                case 4:
42                    viewUsers();
43                    break;
44                case 5:
45                    addRole(scanner);
46                    break;
47                case 6:
48                    assignRole(scanner);
49                    break;
50                case 7:
51                    removeRole(scanner);
52                    break;
53                case 8:
54                    listRoles();
55                    break;
56                case 9:
```

```

57         searchUserByName(scanner);
58         break;
59     case 10:
60         searchUserByEmail(scanner);
61         break;
62     case 11:
63         System.out.println("Exiting the program...");
64         break;
65     default:
66         System.out.println("Invalid option. Please enter a valid
number.");
67     }
68     } while (option != 11);
69     scanner.close();
70 }
71
72 private static Connection getConnection() throws SQLException {
73     return DriverManager.getConnection(DB_URL, USER, PASS);
74 }
75
76 private static void addUser(Scanner scanner) {
77     System.out.print("Enter user name: ");
78     scanner.nextLine(); // clear buffer
79     String name = scanner.nextLine();
80     System.out.print("Enter email: ");
81     String email = scanner.next();
82     String sql = "INSERT INTO users (name, email) VALUES (?, ?)";
83     try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
84         conn.setAutoCommit(false);
85         pstmt.setString(1, name);
86         pstmt.setString(2, email);
87         executeUpdate(pstmt, "User added successfully!");
88         conn.commit();
89     } catch (SQLException e) {
90         System.out.println("Error adding user: " + e.getMessage());
91     }
92 }
93
94 private static void updateUser(Scanner scanner) {
95     System.out.print("Enter user id to update: ");
96     int id = scanner.nextInt();
97     scanner.nextLine(); // clear buffer
98     System.out.print("Enter new email: ");
99     String email = scanner.nextLine();
100    String sql = "UPDATE users SET email = ? WHERE id = ?";
101    try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
102        conn.setAutoCommit(false);
103        pstmt.setString(1, email);
104        pstmt.setInt(2, id);
105        executeUpdate(pstmt, "User updated successfully!");
106        conn.commit();
107    } catch (SQLException e) {
108        System.out.println("Error updating user: " + e.getMessage());
109    }

```

```

110     }
111
112     private static void deleteUser(Scanner scanner) {
113         System.out.print("Enter user id to delete: ");
114         int id = scanner.nextInt();
115         String sql = "DELETE FROM users WHERE id = ?";
116         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
117             conn.setAutoCommit(false);
118             pstmt.setInt(1, id);
119             executeUpdate(pstmt, "User deleted successfully!");
120             conn.commit();
121         } catch (SQLException e) {
122             System.out.println("Error deleting user: " + e.getMessage());
123         }
124     }
125
126     private static void viewUsers() {
127         String sql = "SELECT * FROM users";
128         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql); ResultSet rs = pstmt.executeQuery()) {
129             System.out.println("List of all users:");
130             while (rs.next()) {
131                 int id = rs.getInt("id");
132                 String name = rs.getString("name");
133                 String email = rs.getString("email");
134                 System.out.printf("ID: %d, Name: %s, Email: %s\n", id, name, email);
135             }
136         } catch (SQLException e) {
137             System.out.println("Error retrieving users: " + e.getMessage());
138         }
139     }
140
141     private static void addRole(Scanner scanner) {
142         System.out.print("Enter role name: ");
143         scanner.nextLine(); // clear buffer
144         String roleName = scanner.nextLine();
145         String sql = "INSERT INTO roles (role_name) VALUES (?)";
146         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
147             conn.setAutoCommit(false);
148             pstmt.setString(1, roleName);
149             executeUpdate(pstmt, "Role added successfully!");
150             conn.commit();
151         } catch (SQLException e) {
152             System.out.println("Error adding role: " + e.getMessage());
153         }
154     }
155
156     private static void assignRole(Scanner scanner) {
157         System.out.print("Enter user id for role assignment: ");
158         int userId = scanner.nextInt();
159         System.out.print("Enter role id to assign: ");
160         int roleId = scanner.nextInt();
161         String sql = "INSERT INTO user_roles (user_id, role_id) VALUES (?, ?)";

```

```

162         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
163             conn.setAutoCommit(false);
164             pstmt.setInt(1, userId);
165             pstmt.setInt(2, roleId);
166             executeUpdate(pstmt, "Role assigned to user successfully!");
167             conn.commit();
168         } catch (SQLException e) {
169             System.out.println("Error assigning role: " + e.getMessage());
170         }
171     }
172
173     private static void removeRole(Scanner scanner) {
174         System.out.print("Enter user id for role removal: ");
175         int userId = scanner.nextInt();
176         System.out.print("Enter role id to remove: ");
177         int roleId = scanner.nextInt();
178         String sql = "DELETE FROM user_roles WHERE user_id = ? AND role_id = ?";
179         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
180             conn.setAutoCommit(false);
181             pstmt.setInt(1, userId);
182             pstmt.setInt(2, roleId);
183             executeUpdate(pstmt, "Role removed from user successfully!");
184             conn.commit();
185         } catch (SQLException e) {
186             System.out.println("Error removing role: " + e.getMessage());
187         }
188     }
189
190     private static void listRoles() {
191         String sql = "SELECT * FROM roles";
192         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql); ResultSet rs = pstmt.executeQuery()) {
193             System.out.println("List of all roles:");
194             while (rs.next()) {
195                 int roleId = rs.getInt("role_id");
196                 String roleName = rs.getString("role_name");
197                 System.out.printf("Role ID: %d, Role Name: %s\n", roleId, roleName);
198             }
199         } catch (SQLException e) {
200             System.out.println("Error listing roles: " + e.getMessage());
201         }
202     }
203
204     private static void searchUserByName(Scanner scanner) {
205         System.out.print("Enter user name to search: ");
206         scanner.nextLine(); // clear buffer
207         String name = scanner.nextLine();
208         String sql = "SELECT * FROM users WHERE name LIKE ?";
209         try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
210             pstmt.setString(1, "%" + name + "%");
211             try (ResultSet rs = pstmt.executeQuery()) {
212                 System.out.println("Search results:");
213                 while (rs.next()) {

```

```

214         int id = rs.getInt("id");
215         String email = rs.getString("email");
216         System.out.printf("ID: %d, Name: %s, Email: %s\n", id, name,
email);
217     }
218 }
219 } catch (SQLException e) {
220     System.out.println("Error searching for user: " + e.getMessage());
221 }
222 }
223
224 private static void searchUserByEmail(Scanner scanner) {
225     System.out.print("Enter email to search for: ");
226     scanner.nextLine(); // clear buffer
227     String email = scanner.nextLine();
228     String sql = "SELECT * FROM users WHERE email LIKE ?";
229     try (Connection conn = getConnection(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
230         pstmt.setString(1, "%" + email + "%");
231         try (ResultSet rs = pstmt.executeQuery()) {
232             System.out.println("Search results:");
233             while (rs.next()) {
234                 int id = rs.getInt("id");
235                 String name = rs.getString("name");
236                 System.out.printf("ID: %d, Name: %s, Email: %s\n", id, name,
email);
237             }
238         }
239     } catch (SQLException e) {
240         System.out.println("Error searching for email: " + e.getMessage());
241     }
242 }
243
244 private static void executeUpdate(PreparedStatement pstmt, String
successMessage) throws SQLException {
245     int affectedRows = pstmt.executeUpdate();
246     if (affectedRows > 0) {
247         System.out.println(successMessage);
248         System.out.println("Operation was successful.");
249     } else {
250         System.out.println("Operation failed, no changes were made.");
251     }
252 }
253 }
254

```