```
package sql2;

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

import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.time.Instant;

public class hii {

    private static Connection conn;

    public static void main(String[] args) {
            try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/dhwani", "root",
"Pallu@vaishu572");

            // Show the login page first
            showLoginPage();

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Database connection error: " +
e.getMessage());
        }
    }

    private static void showLoginPage() {
        // Create a login dialog
        JPanel loginPanel = new JPanel();
        loginPanel.setLayout(new GridLayout(3, 2, 10, 10));
        loginPanel.setBorder(new EmptyBorder(100,100,100,100));
        loginPanel.setBackground(new Color(25, 20, 20));

        JLabel lblUserId = new JLabel("User ID:");
        lblUserId.setForeground(Color.WHITE);
        JTextField txtUserId = new JTextField();

        JLabel lblPassword = new JLabel("Password:");
        lblPassword.setForeground(Color.WHITE);
        JPasswordField txtPassword = new JPasswordField();
```

```
    JButton btnLogin = new JButton("Login");
    btnLogin.setBackground(new Color(30, 215, 96));
    btnLogin.setForeground(Color.BLACK);
    btnLogin.setFont(new Font("SansSerif", Font.BOLD, 40));
    btnLogin.setFocusPainted(false);

    // Add action listener for login button
    btnLogin.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String userId = txtUserId.getText();
            String password = new String(txtPassword.getPassword());

            if (validateLogin(userId, password)) {
                // If login is successful, open the main application
                showMainApp();
            } else {
                JOptionPane.showMessageDialog(null, "Invalid credentials, please try again.");
            }
        }
    });

    // Add components to the login panel
    loginPanel.add(lblUserId);
    loginPanel.add(txtUserId);
    loginPanel.add(lblPassword);
    loginPanel.add(txtPassword);
    loginPanel.add(new JLabel()); // Empty label for spacing
    loginPanel.add(btnLogin);

    // Create the login frame
    JFrame loginFrame = new JFrame("Login");
    loginFrame.setSize(400, 300);
    loginFrame.setLocationRelativeTo(null);
    loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    loginFrame.add(loginPanel);
    loginFrame.setVisible(true);
}

private static boolean validateLogin(String userId, String password) {
    String query = "SELECT * FROM users WHERE user_id = ? AND password = ?";
    try (PreparedStatement pstmt = conn.prepareStatement(query)) {
        pstmt.setInt(1, Integer.parseInt(userId)); // Set User ID
        pstmt.setString(2, password); // Set Password
```

```
            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    return true; // User found, login successful
                }
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error validating login: " + e.getMessage());
        }
        return false; // User not found or invalid credentials
    }

    private static void showMainApp() {
        // Create the main frame
        JFrame frame = new JFrame("Dhwani Music Management System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 500);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.getContentPane().setBackground(new Color(25, 20, 20)); // Dark background color

            // Create a header panel for branding
            JPanel headerPanel = new JPanel();
            headerPanel.setBackground(new Color(23,104,79)); // Spotify-like green color
            headerPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

            JLabel headerLabel = new JLabel("Welcome to Dhwani Music App!");
            headerLabel.setForeground(Color.WHITE);
            headerLabel.setFont(new Font("SansSerif", Font.BOLD, 40));
            headerPanel.add(headerLabel);

            // Create the main button panel
            JPanel buttonPanel = new JPanel();
            buttonPanel.setBackground(new Color(25, 20, 20));
            buttonPanel.setBorder(new EmptyBorder(70, 70, 70, 70)); // Added more padding to the
borders
            buttonPanel.setLayout(new GridLayout(3, 2, 20, 20)); // 3 rows and 2 columns with gaps

            // Create buttons with custom styling
            JButton btnInsertUser = createStyledButton("Insert User");
            JButton btnInsertSong = createStyledButton("Insert Song");
            JButton btnInsertPlaylist = createStyledButton("Insert Playlist");
            JButton btnViewSongs = createStyledButton("View Songs");
            JButton btnViewConcerts = createStyledButton("View Concerts");
            JButton btnExit = createStyledButton("Exit");
```

```
        // Add action listeners
        btnInsertUser.addActionListener(new InsertUserListener());
        btnInsertSong.addActionListener(new InsertSongListener());
        btnInsertPlaylist.addActionListener(new InsertPlaylistListener());
        btnViewSongs.addActionListener(new ViewSongsListener());
        btnViewConcerts.addActionListener(new ViewConcertsListener());
        btnExit.addActionListener(e -> System.exit(0));

        // Add buttons to the button panel
        buttonPanel.add(btnInsertUser);
        buttonPanel.add(btnInsertSong);
        buttonPanel.add(btnInsertPlaylist);
        buttonPanel.add(btnViewSongs);
        buttonPanel.add(btnViewConcerts);
        buttonPanel.add(btnExit);

        // Add components to the frame
        frame.setLayout(new BorderLayout());
        frame.add(headerPanel, BorderLayout.NORTH);
        frame.add(buttonPanel, BorderLayout.CENTER);
        frame.setVisible(true);
    }

   private static JButton createStyledButton(String text) {
      JButton button = new JButton(text);
      button.setBackground(new Color(30, 215, 96)); // Spotify-like green color
      button.setForeground(Color.BLACK); // Black text for contrast
      button.setFont(new Font("SansSerif", Font.BOLD, 30)); // Reduced font size
      button.setFocusPainted(false);
      button.setPreferredSize(new Dimension(150, 40)); // Reduced button size
      button.setBorder(BorderFactory.createLineBorder(new Color(20, 200, 90), 2, true)); //
Rounded border

      // Add hover effect
      button.addMouseListener(new java.awt.event.MouseAdapter() {
         public void mouseEntered(java.awt.event.MouseEvent evt) {
            button.setBackground(new Color(20, 180, 80)); // Slightly darker green on hover
         }

         public void mouseExited(java.awt.event.MouseEvent evt) {
            button.setBackground(new Color(30, 215, 96));
         }
      });
```

```
        return button;
    }

    // Original logic and database-related code
    private static void displayTableData(String query, String title) {
        try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
            ResultSetMetaData metaData = rs.getMetaData();
            int columnCount = metaData.getColumnCount();

            // Prepare table data
            String[] columnNames = new String[columnCount];
            for (int i = 0; i < columnCount; i++) {
                columnNames[i] = metaData.getColumnLabel(i + 1);
            }

            // Fill data
            DefaultTableModel model = new DefaultTableModel(columnNames, 0);
            while (rs.next()) {
                Object[] row = new Object[columnCount];
                for (int i = 0; i < columnCount; i++) {
                    row[i] = rs.getObject(i + 1);
                }
                model.addRow(row);
            }

            // Display data in JTable
            JTable table = new JTable(model);
            JScrollPane scrollPane = new JScrollPane(table);
            JOptionPane.showMessageDialog(null, scrollPane, title,
JOptionPane.PLAIN_MESSAGE);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error displaying data: " + e.getMessage());
        }
    }

    static class InsertUserListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String idStr = JOptionPane.showInputDialog("Enter User ID:");
            String username = JOptionPane.showInputDialog("Enter Username:");
            String password = JOptionPane.showInputDialog("Enter password:");

            String email = JOptionPane.showInputDialog("Enter Email:");

            if (idStr != null && username != null && email != null && password!=null) {
```

```
            try {
                int userId = Integer.parseInt(idStr);  // Convert ID input to integer

                // Prepare SQL statement with user_id and created_at
                String sql = "INSERT INTO users (user_id, username, email, created_at,password)
VALUES (?, ?, ?, ?,?)";
                try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                    pstmt.setInt(1, userId);         // Set user ID
                    pstmt.setString(2, username);    // Set username
                    pstmt.setString(3, email);       // Set email
                    pstmt.setTimestamp(4, Timestamp.from(Instant.now())); // Set created_at to
current timestamp
                    pstmt.setString(5, password);
                    pstmt.executeUpdate();
                    JOptionPane.showMessageDialog(null, "User inserted successfully.");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "User ID must be a valid number.");
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(null, "Error inserting user: " + ex.getMessage());
            }
        } else {
            JOptionPane.showMessageDialog(null, "Please fill in all fields.");
        }
    }
  }
    // Method to generate a unique history ID
    private int generateHistoryId() {
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT MAX(history_id) FROM user_history")) {
            if (rs.next()) {
                return rs.getInt(1) + 1;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return (int) (Math.random() * 100000);
    }

   // Insert Song
    static class InsertSongListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String songIdStr = JOptionPane.showInputDialog("Enter Song ID:");
            String title = JOptionPane.showInputDialog("Enter Song Title:");
```

```
            String durationStr = JOptionPane.showInputDialog("Enter Duration (in seconds):");
            String albumIdStr = JOptionPane.showInputDialog("Enter Album ID:");
            String artistIdStr = JOptionPane.showInputDialog("Enter Artist ID:");

            // Check if any input field is empty
            if (songIdStr == null || title == null || durationStr == null || albumIdStr == null ||
artistIdStr == null ||
                songIdStr.isEmpty() || title.isEmpty() || durationStr.isEmpty() || albumIdStr.isEmpty()
|| artistIdStr.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Please fill in all fields.");
                return;
            }

            try {
                int songId = Integer.parseInt(songIdStr);
                int duration = Integer.parseInt(durationStr);
                int albumId = Integer.parseInt(albumIdStr);
                int artistId = Integer.parseInt(artistIdStr);

                // Establish a connection if it's not already connected
                if (conn == null || conn.isClosed()) {
                    conn = hii.getConnection();
                }

                // SQL query to insert song details
                String sql = "INSERT INTO songs (song_id, title, duration, album_id, artist_id)
VALUES (?, ?, ?, ?, ?)";

                // Use try-with-resources to manage PreparedStatement
                try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                    pstmt.setInt(1, songId);
                    pstmt.setString(2, title);
                    pstmt.setInt(3, duration);
                    pstmt.setInt(4, albumId);
                    pstmt.setInt(5, artistId);

                    pstmt.executeUpdate();

                    JOptionPane.showMessageDialog(null, "Song inserted successfully.");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "ID and duration fields must be valid
numbers.");
            } catch (SQLException ex) {
```

```
                JOptionPane.showMessageDialog(null, "Error inserting song: " + ex.getMessage());
            }
        }
    }


    // Insert Playlist
    static class InsertPlaylistListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                String userIdStr = JOptionPane.showInputDialog("Enter User ID:");
                String playlistIdStr = JOptionPane.showInputDialog("Enter Playlist ID:");
                        String playlistName = JOptionPane.showInputDialog("Enter Playlist
Name:");

                        try {
                            int userId = Integer.parseInt(userIdStr);
                            int playlistID= Integer.parseInt(playlistIdStr);
                            String sql = "INSERT INTO playlists (playlist_id, name, created_at,
user_id) VALUES (?, ?, ?, ?)";
                            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                                pstmt.setInt(1, playlistID);
                                pstmt.setString(2, playlistName);
                                pstmt.setTimestamp(3, Timestamp.from(Instant.now()));
                                pstmt.setInt(4, userId);
                                pstmt.executeUpdate();


                JOptionPane.showMessageDialog(null, "Playlist inserted successfully.");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "User ID and Playlist ID must be valid
numbers.");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Error inserting playlist: " +
ex.getMessage());
        }
    }
}


// View Songs
static class ViewSongsListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        displayTableData("SELECT * FROM songs", "Songs");
```

```
        }
    }

    // View Concerts
    static class ViewConcertsListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            displayTableData("SELECT * FROM concerts", "Concerts");
        }
    }
            public static Connection getConnection() {
                    // TODO Auto-generated method stub
                    return null;
            }
}
```