HOTEL MANAGEMENT AND BG

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.io.File;
import java.sql.*;

public class Image extends JFrame {

    private static final String url = "jdbc:mysql://localhost:3306/
hotel_db";
    private static final String username = "root";
    private static final String password = "Vigshan@2116";

    private JTextField guestNameField, roomNumberField, contactNumberField,
reservationIdField;
    private JTextArea outputArea;
    private Connection connection;

    // ImagePanel to handle the background image
    class ImagePanel extends JPanel {
        private java.awt.Image backgroundImage;

        public ImagePanel(String imagePath) {
            try {
                backgroundImage = ImageIO.read(new File(imagePath)); // Load
the image
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            // Draw the image to fit the panel
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(),
this);
        }
    }

    public Image() {
        try {
            connection = DriverManager.getConnection(url, username,
password);
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Database connection
failed", "Error", JOptionPane.ERROR_MESSAGE);
        }

        setTitle("Hotel Reservation System");
        setSize(470, 430);
        setLocationRelativeTo(null);  // Center the frame on the screen
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        initUI();
    }

    private void initUI() {
        // Create a custom panel with background image
        ImagePanel panel = new ImagePanel("/Users/shangamithra/Downloads/
hotel.jpeg");
        panel.setLayout(new BorderLayout());

        // Create a tabbed pane to organize operations
        JTabbedPane tabbedPane = new JTabbedPane();

        // Create tabs for different operations
        tabbedPane.addTab("Reserve Room", createReserveRoomTab());
        tabbedPane.addTab("View Reservations", createViewReservationsTab());
        tabbedPane.addTab("Update Reservation",
createUpdateReservationTab());
        tabbedPane.addTab("Delete Reservation",
createDeleteReservationTab());

        // Add the tabbed pane to the main panel
        panel.add(tabbedPane, BorderLayout.CENTER);

        // Output Area to show results
        outputArea = new JTextArea(8, 7);
```

```java
        outputArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(outputArea);
        panel.add(scrollPane, BorderLayout.SOUTH);

        // Add panel to the frame
        add(panel);
    }

    // Create Reserve Room Tab
    private JPanel createReserveRoomTab() {
        JPanel reservePanel = new JPanel();
        reservePanel.setLayout(new GridLayout(4, 2));

        JLabel guestNameLabel = new JLabel("Guest Name:");
        guestNameField = new JTextField();
        JLabel roomNumberLabel = new JLabel("Room Number:");
        roomNumberField = new JTextField();
        JLabel contactNumberLabel = new JLabel("Contact Number:");
        contactNumberField = new JTextField();

        JButton reserveButton = new JButton("Reserve Room");
        reserveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                reserveRoom();
            }
        });

        reservePanel.add(guestNameLabel);
        reservePanel.add(guestNameField);
        reservePanel.add(roomNumberLabel);
        reservePanel.add(roomNumberField);
        reservePanel.add(contactNumberLabel);
        reservePanel.add(contactNumberField);
        reservePanel.add(reserveButton);

        return reservePanel;
    }

    // Create View Reservations Tab
    private JPanel createViewReservationsTab() {
        JPanel viewPanel = new JPanel();
        viewPanel.setLayout(new BorderLayout());

        JButton viewButton = new JButton("View Reservations");
        viewButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                viewReservations();
            }
        });

        viewPanel.add(viewButton, BorderLayout.CENTER);
        return viewPanel;
    }

    // Create Update Reservation Tab
    private JPanel createUpdateReservationTab() {
        JPanel updatePanel = new JPanel();
        updatePanel.setLayout(new GridLayout(5, 2));

        JLabel reservationIdLabel = new JLabel("Reservation ID:");
        reservationIdField = new JTextField();
        JLabel newGuestNameLabel = new JLabel("New Guest Name:");
        JTextField newGuestNameField = new JTextField();
        JLabel newRoomNumberLabel = new JLabel("New Room Number:");
        JTextField newRoomNumberField = new JTextField();
        JLabel newContactNumberLabel = new JLabel("New Contact Number:");
        JTextField newContactNumberField = new JTextField();

        JButton updateButton = new JButton("Update Reservation");
        updateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Get the input data from the fields
                String reservationId = reservationIdField.getText().trim();
                String newGuestName = newGuestNameField.getText().trim();
                String newRoomNumber = newRoomNumberField.getText().trim();
                String newContactNumber = 
newContactNumberField.getText().trim();

                if (reservationId.isEmpty() || newGuestName.isEmpty() || 
newRoomNumber.isEmpty() || newContactNumber.isEmpty()) {
                    outputArea.setText("Please fill all fields");
                } else {
                    // Call the method to update the reservation with the
provided data
                    updateReservation(reservationId, newGuestName,
newRoomNumber, newContactNumber);
                }
            }
        });

        updatePanel.add(reservationIdLabel);
        updatePanel.add(reservationIdField);
        updatePanel.add(newGuestNameLabel);
        updatePanel.add(newGuestNameField);
        updatePanel.add(newRoomNumberLabel);
        updatePanel.add(newRoomNumberField);
        updatePanel.add(newContactNumberLabel);
        updatePanel.add(newContactNumberField);
        updatePanel.add(updateButton);

        return updatePanel;
    }

    // Create Delete Reservation Tab
    private JPanel createDeleteReservationTab() {
        JPanel deletePanel = new JPanel();
        deletePanel.setLayout(new BorderLayout());

        JLabel reservationIdLabel = new JLabel("Enter Reservation ID to
delete:");
        JTextField reservationIdToDeleteField = new JTextField();
        JButton deleteButton = new JButton("Delete Reservation");

        deleteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                deleteReservation(reservationIdToDeleteField.getText());
            }
        });

        deletePanel.add(reservationIdLabel, BorderLayout.NORTH);
        deletePanel.add(reservationIdToDeleteField, BorderLayout.CENTER);
        deletePanel.add(deleteButton, BorderLayout.SOUTH);
        return deletePanel;
    }

    // Reserve Room Method
    private void reserveRoom() {
        String guestName = guestNameField.getText();
        int roomNumber = Integer.parseInt(roomNumberField.getText());
        String contactNumber = contactNumberField.getText();

        String sql = "INSERT INTO reservations (guest_name, room_number, 
comtact_number) " +
                     "VALUES (?, ?, ?)";

        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, guestName);
            stmt.setInt(2, roomNumber);
            stmt.setString(3, contactNumber);

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                outputArea.setText("Reservation successful!");
            } else {
                outputArea.setText("Reservation failed.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            outputArea.setText("Error occurred while making the 
reservation.");
        }
    }

    // View Reservations Method
    private void viewReservations() {
        String sql = "SELECT reservation_id, guest_name, room_number, 
comtact_number, reservation_date FROM reservations";

        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {

            StringBuilder result = new StringBuilder();
            result.append("Reservation ID | Guest Name | Room Number |
Contact Number | Reservation Date\n");

result.append("-------------------------------------------------------------
\n");

            while (rs.next()) {
                int reservationId = rs.getInt("reservation_id");
                String guestName = rs.getString("guest_name");
                int roomNumber = rs.getInt("room_number");
                String contactNumber = rs.getString("comtact_number");
                String reservationDate =
rs.getTimestamp("reservation_date").toString();

                result.append(reservationId).append(" | ")
                        .append(guestName).append(" | ")
                        .append(roomNumber).append(" | ")
                        .append(contactNumber).append(" | ")
                        .append(reservationDate).append("\n");
            }

            outputArea.setText(result.toString());
        } catch (SQLException e) {
            e.printStackTrace();
            outputArea.setText("Error occurred while fetching
reservations.");
        }
    }

    // Update Reservation Method
    private void updateReservation(String reservationId, String
newGuestName, String newRoomNumber, String newContactNumber) {
        String sql = "UPDATE reservations SET guest_name = ?, room_number
= ?, comtact_number = ? WHERE reservation_id = ?";

        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, newGuestName);
            stmt.setInt(2, Integer.parseInt(newRoomNumber));
            stmt.setString(3, newContactNumber);
            stmt.setInt(4, Integer.parseInt(reservationId));  // Assuming
reservation ID is an integer

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                outputArea.setText("Reservation updated successfully!");
            } else {
                outputArea.setText("Failed to update the reservation.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            outputArea.setText("Error occurred while updating the
reservation.");
        }
    }

    // Delete Reservation Method
    private void deleteReservation(String reservationId) {
        String sql = "DELETE FROM reservations WHERE reservation_id = ?";

        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, Integer.parseInt(reservationId)); // Convert
reservationId to integer

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                outputArea.setText("Reservation deleted successfully!");
            } else {
                outputArea.setText("Failed to delete the reservation.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            outputArea.setText("Error occurred while deleting the
reservation.");
        }
    }

    public static void main(String[] args) {
        // Run the application
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Image().setVisible(true);  // Changed class name to
Image
            }
        });
        }
}
```