CODE FOR CONTACT APP

```
package contactbook;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
class Contact {
   String name, phone, email;
   Contact(String n, String p, String e) { name = n; phone = p; email = e; }
   @Override
   public String toString() { return name + "," + phone + "," + email; }
}
public class ContactBookApp extends JFrame {
   private JTextField nameField, phoneField, emailField, searchField;
   private JTable contactTable;
   private DefaultTableModel tableModel;
   private java.util.List<Contact> contacts;
   private File file;
   public ContactBookApp() {
      setTitle("Contact Book");
      setSize(650, 500);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLocationRelativeTo(null);
      setLayout(new BorderLayout(10, 10));
      contacts = new ArrayList<>();
      file = new File("contacts.txt");
      loadContacts();
      // Top Panel - Inputs + Search
      JPanel topPanel = new JPanel(new GridBagLayout());
      topPanel.setBorder(new TitledBorder(BorderFactory.createLineBorder(Color.GRAY),
"Add / Search Contact", TitledBorder.LEADING, TitledBorder.TOP, new Font("Arial",
Font.BOLD, 14), Color.DARK_GRAY));
      topPanel.setBackground(new Color(245, 245, 245));
      GridBagConstraints c = new GridBagConstraints();
      c.insets = new Insets(8, 8, 8, 8);
      Font labelFont = new Font("Arial", Font.BOLD, 14);
      Font fieldFont = new Font("Arial", Font.PLAIN, 14);
      c.gridx = 0; c.gridy = 0; topPanel.add(new JLabel("Name:"), c);
      c.gridx = 1; nameField = new JTextField(15); nameField.setFont(fieldFont);
topPanel.add(nameField, c);
      c.gridx = 0; c.gridy = 1; topPanel.add(new JLabel("Phone:"), c);
      c.gridx = 1; phoneField = new JTextField(15); phoneField.setFont(fieldFont);
topPanel.add(phoneField, c);
      c.gridx = 0; c.gridy = 2; topPanel.add(new JLabel("Email:"), c);
      c.gridx = 1; emailField = new JTextField(15); emailField.setFont(fieldFont);
topPanel.add(emailField, c);
      c.gridx = 2; c.gridy = 0; topPanel.add(new JLabel("Search:"), c);
```

```
      c.gridx = 3; searchField = new JTextField(15); searchField.setFont(fieldFont);
topPanel.add(searchField, c);
      add(topPanel, BorderLayout.NORTH);
      // Center Panel - Table
      tableModel = new DefaultTableModel(new Object[]{"Name", "Phone", "Email"}, 0);
      contactTable = new JTable(tableModel) {
          public Component prepareRenderer(TableCellRenderer renderer, int row, int
column) {
              Component c = super.prepareRenderer(renderer, row, column);
              if (!isRowSelected(row)) {
                  c.setBackground(row % 2 == 0 ? Color.WHITE : new Color(230, 230, 230));
              }
              return c;
          }
      };
      contactTable.setRowHeight(25);
      JTableHeader header = contactTable.getTableHeader();
      header.setFont(new Font("Arial", Font.BOLD, 14));
      contactTable.setFont(new Font("Arial", Font.PLAIN, 14));
      contactTable.setEnabled(false);
      JScrollPane tableScroll = new JScrollPane(contactTable);
      tableScroll.setBorder(new TitledBorder(BorderFactory.createLineBorder(Color.GRAY),
"Contacts", TitledBorder.LEADING, TitledBorder.TOP, new Font("Arial", Font.BOLD, 14),
Color.DARK_GRAY));
      add(tableScroll, BorderLayout.CENTER);
      // Bottom Panel - Buttons
      JPanel buttonPanel = new JPanel(new FlowLayout());
      buttonPanel.setBackground(new Color(245, 245, 245));
      JButton addButton = createButton("Add", new Color(34, 139, 34));
      JButton deleteButton = createButton("Delete", new Color(178, 34, 34));
      JButton refreshButton = createButton("Refresh", new Color(30, 144, 255));
      JButton searchButton = createButton("Search", new Color(255, 140, 0));
      buttonPanel.add(addButton); buttonPanel.add(deleteButton);
buttonPanel.add(refreshButton); buttonPanel.add(searchButton);
      add(buttonPanel, BorderLayout.SOUTH);
      // Button Actions
      addButton.addActionListener(e -> addContact());
      deleteButton.addActionListener(e -> deleteContact());
      refreshButton.addActionListener(e -> displayContacts());
      searchButton.addActionListener(e -> searchContacts());
      displayContacts();
      setVisible(true);
  }
  // Create styled buttons
  private JButton createButton(String text, Color bg) {
      JButton button = new JButton(text);
      button.setBackground(bg); button.setForeground(Color.WHITE);
      button.setFont(new Font("Arial", Font.BOLD, 14));
      button.setFocusPainted(false);
      button.setPreferredSize(new Dimension(100, 30));
      return button;
  }
  private void addContact() {
      String name = nameField.getText().trim();
```

```
        String phone = phoneField.getText().trim();
        String email = emailField.getText().trim();
        if (name.isEmpty() || phone.isEmpty() || email.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please fill all fields!"); return;
        }
        contacts.add(new Contact(name, phone, email));
        saveContacts(); displayContacts();
        nameField.setText(""); phoneField.setText(""); emailField.setText("");
    }
    private void deleteContact() {
        String nameToDelete = JOptionPane.showInputDialog(this, "Enter name to delete:");
        if (nameToDelete == null || nameToDelete.trim().isEmpty()) return;
        boolean found = false;
        Iterator<Contact> it = contacts.iterator();
        while (it.hasNext()) {
            Contact c = it.next();
            if (c.name.equalsIgnoreCase(nameToDelete.trim())) { it.remove(); found = true;
break; }
        }
        if (found) { saveContacts(); displayContacts(); JOptionPane.showMessageDialog(this,
"Deleted successfully!"); }
        else JOptionPane.showMessageDialog(this, "Contact not found!");
    }
    private void searchContacts() {
        String query = searchField.getText().trim().toLowerCase();
        tableModel.setRowCount(0);
        for (Contact c : contacts) {
            if (c.name.toLowerCase().contains(query)) {
                tableModel.addRow(new Object[]{c.name, c.phone, c.email});
            }
        }
    }
    private void displayContacts() {
        tableModel.setRowCount(0);
        for (Contact c : contacts) tableModel.addRow(new Object[]{c.name, c.phone,
c.email});
    }
    private void loadContacts() {
        if (!file.exists()) return;
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(",");
                if (parts.length == 3) contacts.add(new Contact(parts[0], parts[1], parts[2]));
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
    private void saveContacts() {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
            for (Contact c : contacts) { bw.write(c.toString()); bw.newLine(); }
        } catch (IOException e) { e.printStackTrace(); }
    }
    public static void main(String[] args) { new ContactBookApp(); }
}
```