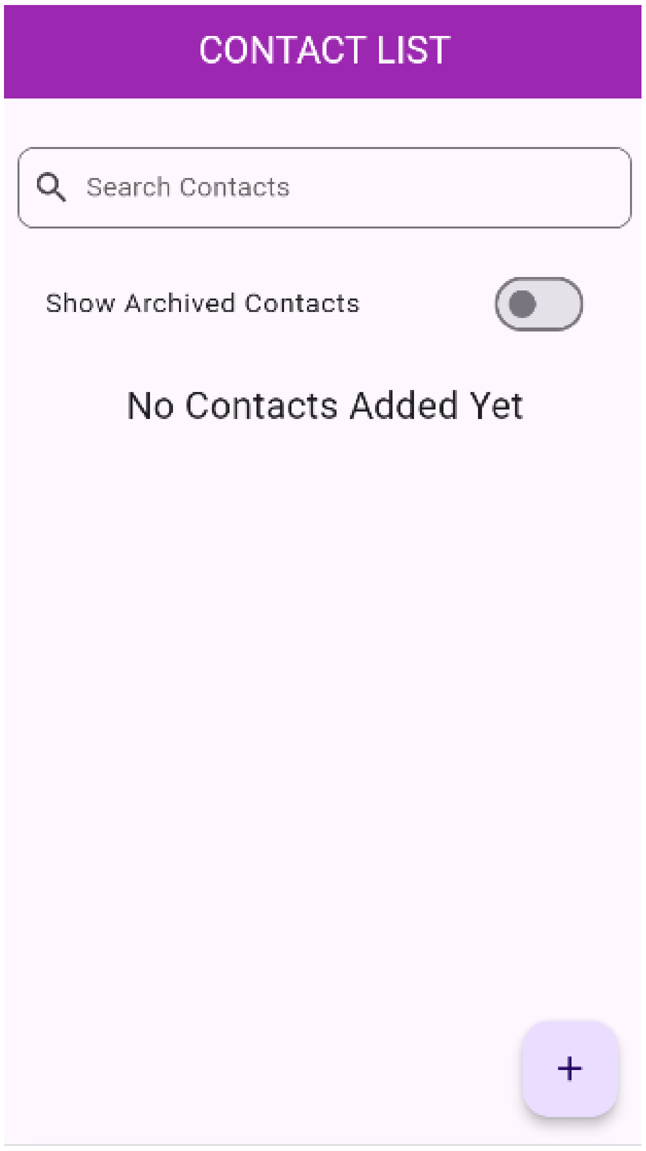
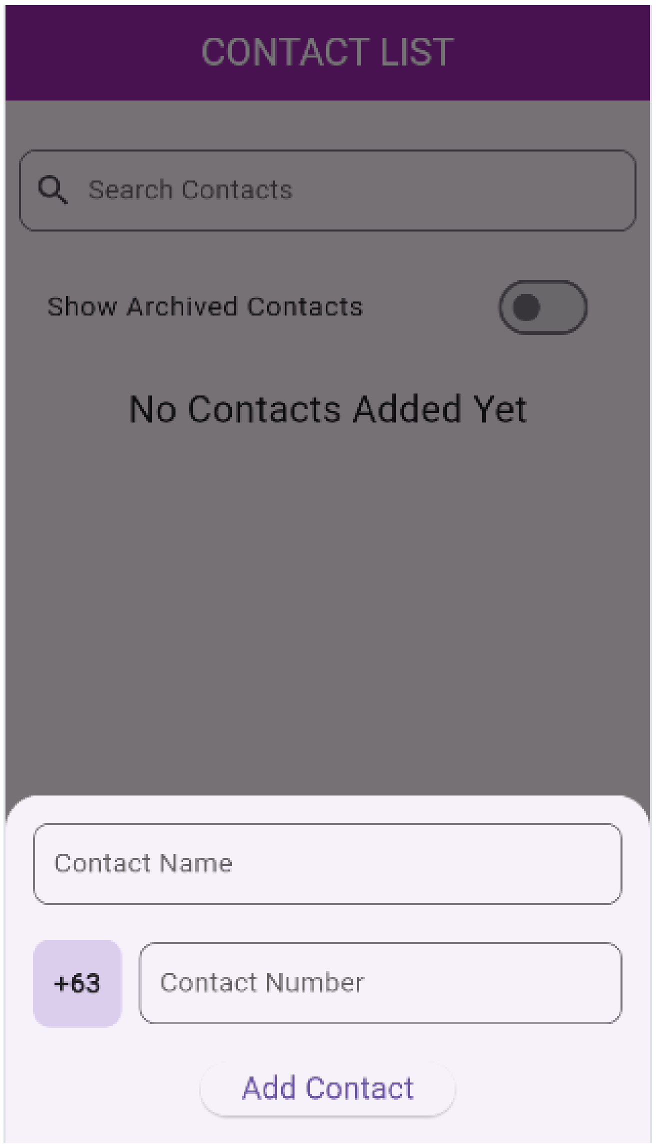
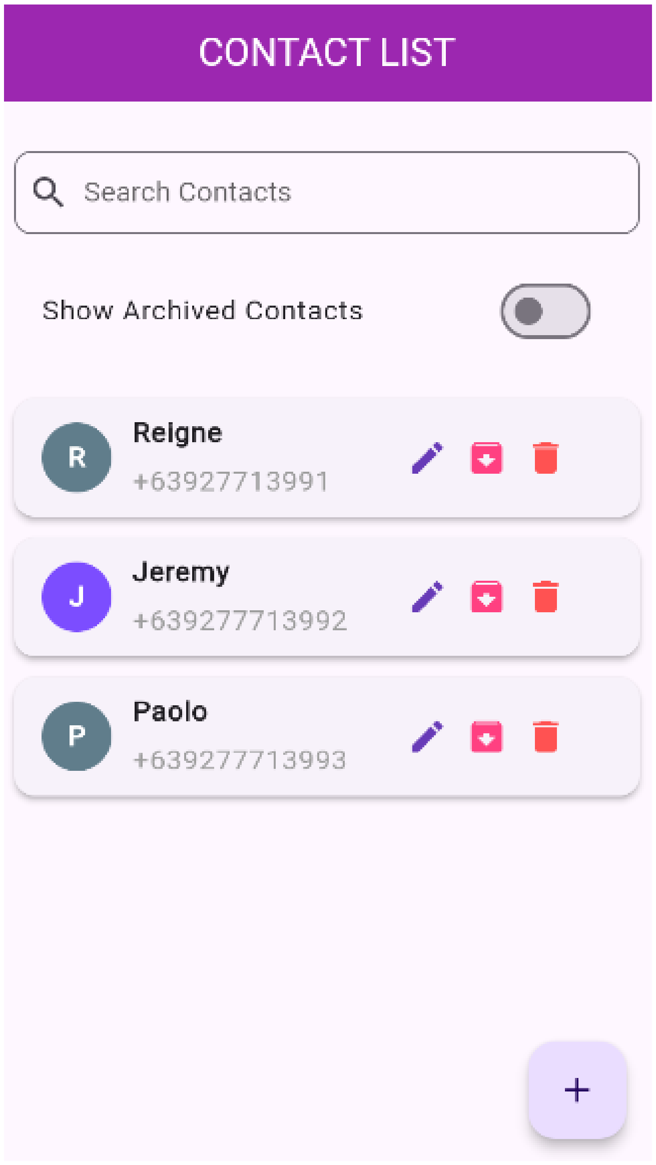
Screenshot of the Mobile Application





Without Added Contacts

Add Contact

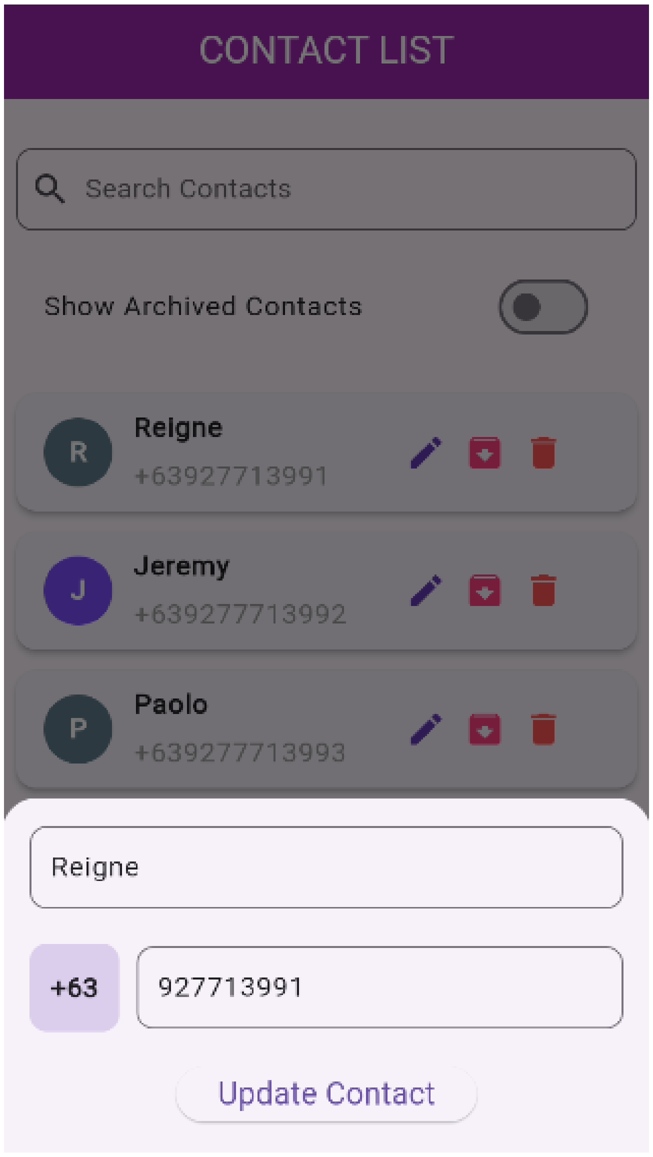

CONTACT LIST

Search Contacts

Show Archived Contacts

Reigne	+63927713991
Jeremy	+639277713992
Paolo	+639277713993

With Added Contacts


CONTACT LIST

Search Contacts

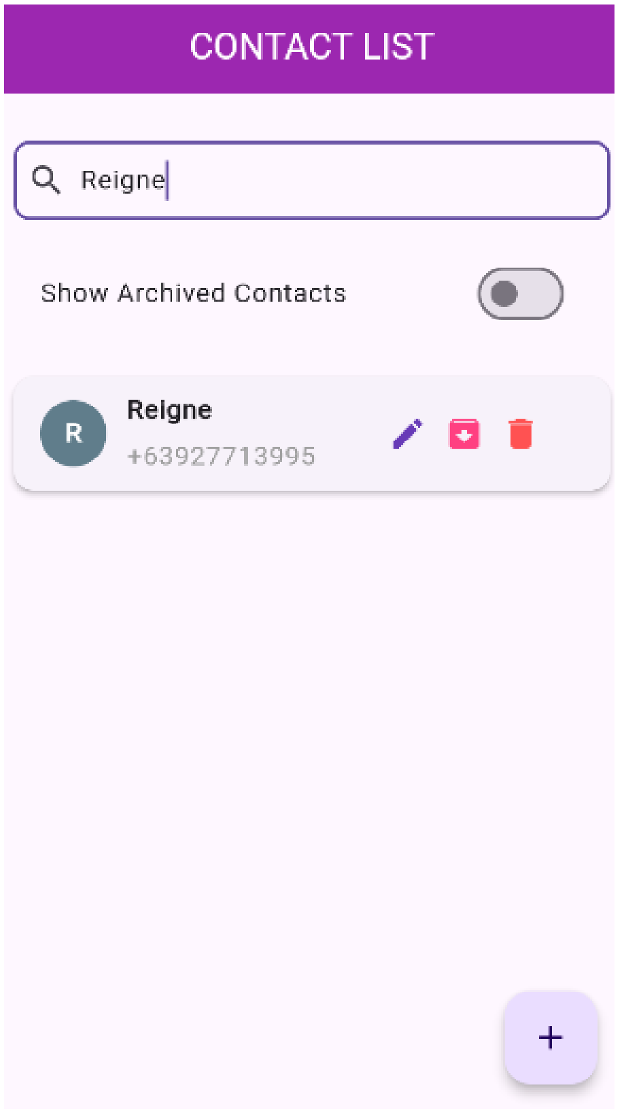
Show Archived Contacts

Reigne	+63927713991
Jeremy	+639277713992
Paolo	+639277713993

Reigne

+63 927713991

Update Contact

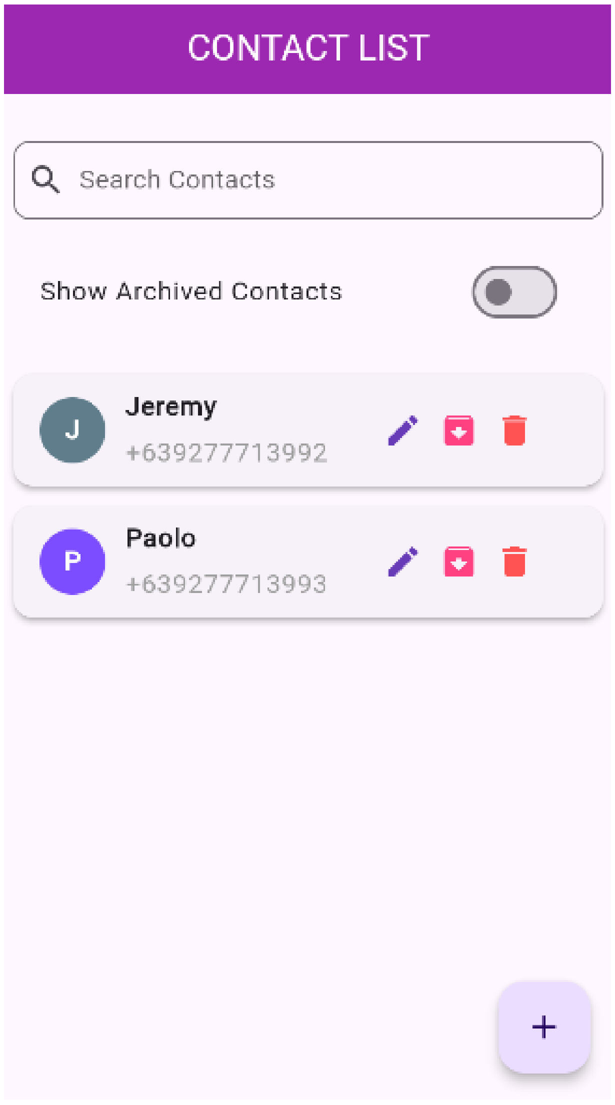

CONTACT LIST

Reigne

Show Archived Contacts

Reigne
+63927713995

Search Function


CONTACT LIST

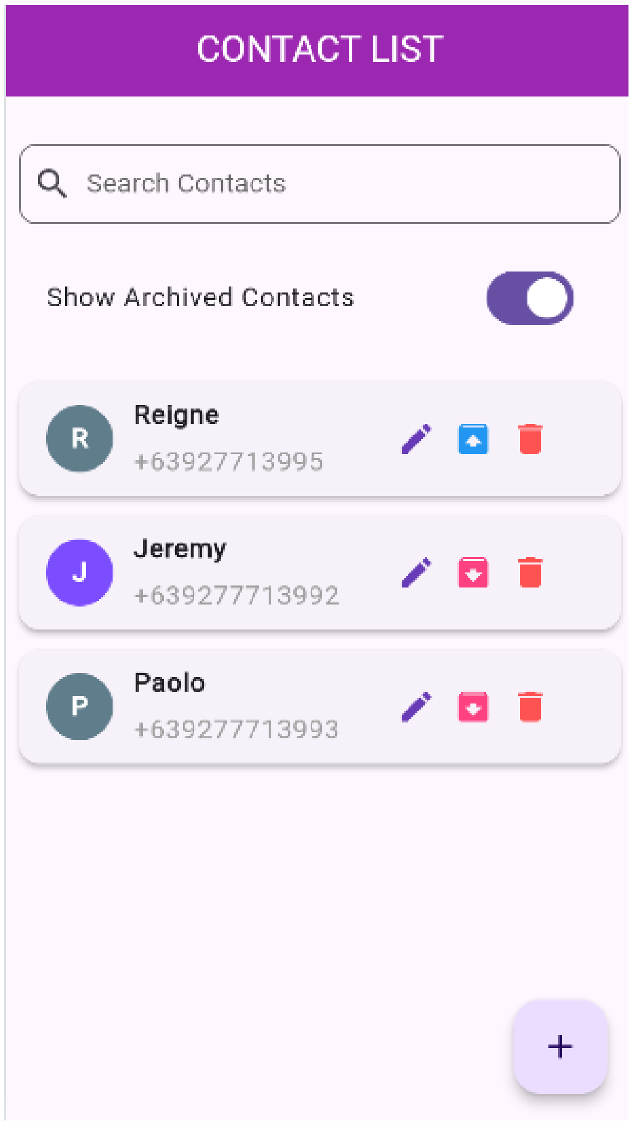
Search Contacts

Show Archived Contacts

Jeremy
+639277713992

Paolo
+639277713993

Archived not Shown

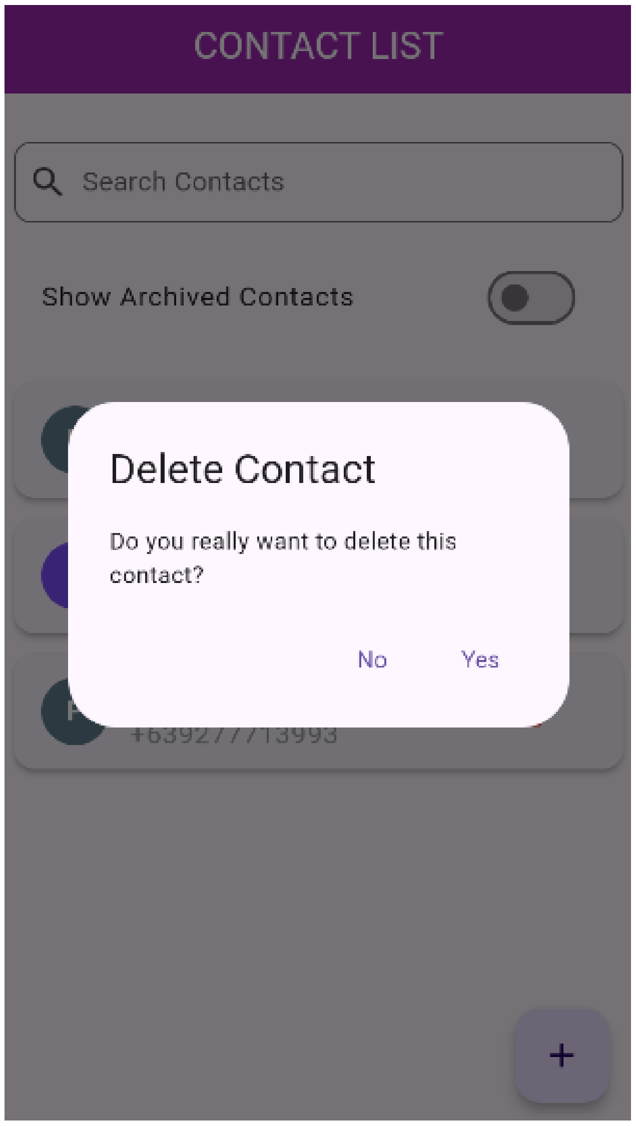


CONTACT LIST

Search Contacts

Show Archived Contacts

Reigne	+63927713995
Jeremy	+639277713992
Paolo	+639277713993



CONTACT LIST

Search Contacts

Show Archived Contacts

Delete Contact

Do you really want to delete this contact?

Show Archived Contacts

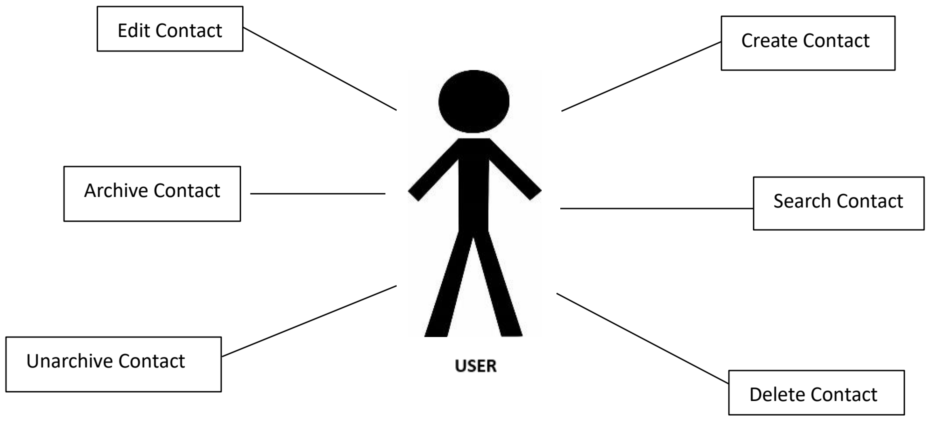
Delete Contact

User can Unarchive through the blue button

Summary of the Functions of Mobile App

In the mobile contact list app, users can easily manage their contacts by creating, editing, and deleting them. They can also archive or unarchive contacts as needed. A search function was also included to help the user quickly find specific contacts in the list.

Use-Case Diagram



Main.dart

```
import 'package:new_contact_list/home_page.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "CONTACT LIST",
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.purple,
      ),
      home: const HomePage(),
    );
  }
}
```

Contact.Dart

```
import 'package:flutter/material.dart';
import 'contact.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  final TextEditingController nameController = TextEditingController();
  final TextEditingController contactController = TextEditingController();
  final TextEditingController searchController = TextEditingController();

  List<Contact> contacts = List.empty(growable: true);
  List<Contact> filteredContacts = List.empty(growable: true);
  int? selectedContactIndex;
  bool showArchived = false;

  @override
  void initState() {
    super.initState();
    searchController.addListener(() {
      filterContacts();
    });
  }

  @override
  void dispose() {
    nameController.dispose();
    contactController.dispose();
    searchController.dispose();
    super.dispose();
  }

  void addOrUpdateContact() {
    String name = nameController.text.trim();
    String contact = contactController.text.trim();

    if (name.isEmpty || contact.isEmpty) {
      return;
    }

    contact = '+63' + contact;

    setState(() {
      if (selectedContactIndex == null) {
        contacts.add(Contact(name: name, contact: contact));
      } else {
        contacts[selectedContactIndex!] = Contact(name: name, contact: contact);
        selectedContactIndex = null;
      }
      nameController.clear();
      contactController.clear();
      filterContacts();
    });
    Navigator.pop(context);
  }

  void editContact(int index) {
    setState(() {
      selectedContactIndex = index;
      nameController.text = filteredContacts[index].name;
      contactController.text = filteredContacts[index].contact.replaceFirst('+63', '');
    });
    showContactModal();
  }

  Future<void> archiveContact(int index) async {
    setState(() {
      filteredContacts[index].isArchived = true;
      filterContacts(); // Update the filtered list
    });
  }
```

```
Future<void> unarchiveContact(int index) async {
  setState(() {
    filteredContacts[index].isArchived = false;
    filterContacts(); // Update the filtered list
  });
}

Future<void> deleteContact(int index) async {
  bool? confirm = await showDeleteConfirmationDialog();

  if (confirm == true) {
    setState(() {
      contacts.removeAt(contacts.indexOf(filteredContacts[index]));
      filterContacts(); // Update the filtered list
      if (selectedContactIndex == index) {
        selectedContactIndex = null;
        nameController.clear();
        contactController.clear();
      }
    });
  }
}

Future<bool?> showDeleteConfirmationDialog() {
  return showDialog<bool>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Delete Contact'),
        content: const Text('Do you really want to delete this contact?'),
        actions: <Widget>[
          TextButton(
            onPressed: () => Navigator.of(context).pop(false),
            child: const Text('No'),
          ),
          TextButton(
            onPressed: () => Navigator.of(context).pop(true),
            child: const Text('Yes'),
          ),
        ],
      );
    },
  );
}

void filterContacts() {
  String query = searchController.text.toLowerCase().trim();
  setState(() {
    filteredContacts = contacts.where((contact) {
      if (!showArchived && contact.isArchived) return false;
      return (contact.name.toLowerCase().contains(query) ||
          contact.contact.contains(query));
    }).toList();
  });
}

void showContactModal() {
  showModalBottomSheet(
    context: context,
    isScrollControlled: true,
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.vertical(top: Radius.circular(20)),
    ),
    builder: (context) {
      return Padding(
        padding: EdgeInsets.only(
          bottom: MediaQuery.of(context).viewInsets.bottom,
          left: 16,
          right: 16,
          top: 16,
        ),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextField(
              controller: nameController,
              decoration: const InputDecoration(
                hintText: "Contact Name",
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
              ),
            ),
            const SizedBox(height: 20),
            Row(
              children: [
                Container(
                  padding: const EdgeInsets.symmetric(horizontal: 12, vertical: 14),
                  decoration: BoxDecoration(
                    color: Colors.deepPurple.withOpacity(0.2),
                    borderRadius: BorderRadius.circular(10),
                  ),
                  child: const Text(
                    "+63",
                    style: TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
                const SizedBox(width: 10),
                Expanded(
                  child: TextField(
                    controller: contactController,
                    keyboardType: TextInputType.number,
                    maxLength: 10,
                    decoration: const InputDecoration(
                      counterText: "",
                      hintText: "Contact Number",
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                    ),
                  ),
                ),
              ],
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: addOrUpdateContact,
              child: Text(selectedContactIndex == null ? "Add Contact" : "Update Contact",style:
TextStyle(fontSize: 18),),
            ),
            const SizedBox(height: 20),
          ],
        ),
      );
    },
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      centerTitle: true,
      backgroundColor: Colors.purple,
      title: const Text("CONTACT LIST", style: TextStyle(color: Colors.white),),
    ),
    body: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        children: [
          const SizedBox(height: 20),
          TextField(
            controller: searchController,
            decoration: const InputDecoration(
              hintText: "Search Contacts",
              border: OutlineInputBorder(
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
              prefixIcon: Icon(Icons.search),
          ),
        ),
        const SizedBox(height: 20),
        SwitchListTile(
          title: const Text("Show Archived Contacts"),
          value: showArchived,
          onChanged: (value) {
            setState(() {
              showArchived = value;
              filterContacts();
            });
          },
        ),
        const SizedBox(height: 20),
        filteredContacts.isEmpty
            ? const Text(
          "No Contacts Added Yet",
          style: TextStyle(fontSize: 22),
        )
            : Expanded(
          child: ListView.builder(
            itemCount: filteredContacts.length,
            itemBuilder: (context, index) => getRow(index),
          ),
        ),
      ],
    ),
  ),
  floatingActionButton: FloatingActionButton(
    onPressed: () {
      selectedContactIndex = null;
      nameController.clear();
      contactController.clear();
      showContactModal();
    },
    child: const Icon(Icons.add),
  ),
);
}

Widget getRow(int index) {
  return Card(
    elevation: 3,
    margin: const EdgeInsets.symmetric(vertical: 6),
    child: ListTile(
      leading: CircleAvatar(
        backgroundColor: index % 2 == 0 ? Colors.blueGrey : Colors.deepPurpleAccent,
        foregroundColor: Colors.white,
        child: Text(
          filteredContacts[index].name[0].toUpperCase(),
          style: const TextStyle(fontWeight: FontWeight.bold),
          ),
        ),
        title: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              filteredContacts[index].name,
              style: const TextStyle(fontWeight: FontWeight.bold),
            ),
            const SizedBox(height: 4),
            Text(
              filteredContacts[index].contact,
              style: const TextStyle(color: Colors.grey),
            ),
          ],
        ),
        trailing: SizedBox(
          width: 120,
          child: Row(
            children: [
              const SizedBox(width: 10),
              InkWell(
                onTap: () => editContact(index),
                child: const Icon(Icons.edit, color: Colors.deepPurple),
              ),
              const SizedBox(width: 10),
              if (filteredContacts[index].isArchived)
                InkWell(
                  onTap: () => unarchiveContact(index),
                  child: const Icon(Icons.unarchive, color: Colors.blue),
                )
              else
                InkWell(
                  onTap: () => archiveContact(index),
                  child: const Icon(Icons.archive, color: Colors.pinkAccent),
                ),

              const SizedBox(width: 10),
              InkWell(
                onTap: () => deleteContact(index),
                child: const Icon(Icons.delete, color: Colors.redAccent),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Contact.dart

```
class Contact {
  String name;
  String contact;
  bool isArchived;

  Contact({required this.name, required this.contact, this.isArchived = false});
}
```