Software Tools 4

Assignment 9

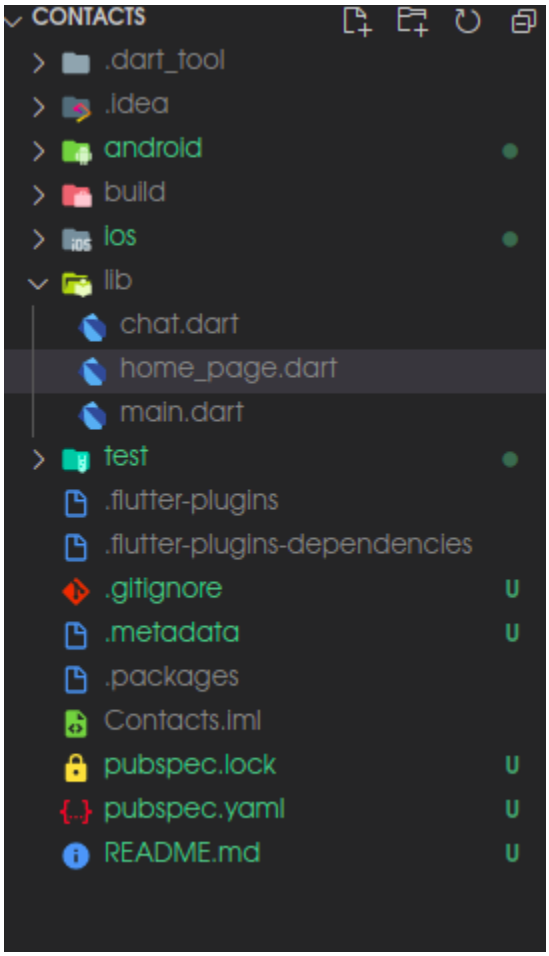
Krunal Rank
U18CO081

Extend your lab assignment 8 by inserting a new table which stores messages related to the persons available in your contact list.

Answer:

Tech Stack used :
Dart
Flutter SDK

Project Directory Structure:



Code:

./lib/main.dart:

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

import 'chat.dart';
import 'home_page.dart';

void main() {
 runApp(MyApp());
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Contacts',
     theme: ThemeData(
       primaryColor: Colors.purple,
       accentColor: Colors.purpleAccent,
     ),
     home: HomePage(),
     routes: {
       HomePage.routeName: (ctx) => HomePage(),
       ChatScreen.routeName: (ctx) => ChatScreen(
             contact: ModalRoute.of(ctx).settings.arguments,
           ),
     },
   );
 }
}
```

./lib/home_page.dart:

```
import 'dart:math';

import 'package:Contacts/chat.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:sliding_up_panel/sliding_up_panel.dart';
import 'package:sqflite/sqflite.dart';

class HomePage extends StatefulWidget {
 static const routeName = '/home';

 @override
 _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<HomePage> {
 bool isLoading = true, selectionMode = false, updateMode = false;
 List<Map> contacts = List<Map>();
 List selections = List();
 Database db;

 final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
 var nameTextController = new TextEditingController();
 var contactTextController = new TextEditingController();
 var emailTextController = new TextEditingController();
 var addressTextController = new TextEditingController();

 String name = "", contact = "", email = "", address = "";

 fetchData() async {
   db = await openDatabase('contacts.db');
   await db.execute(
       'CREATE TABLE IF NOT EXISTS Contacts(Name TEXT, Contact TEXT PRIMARY KEY, Email
TEXT, Address TEXT)');
   await db.execute(
       'CREATE TABLE IF NOT EXISTS Messages(Contact TEXT NOT NULL, MessageID TEXT
PRIMARY KEY, Timestamp TEXT NOT NULL, Content TEXT NOT NULL, FOREIGN KEY(Contact)
REFERENCES Contacts(Contact))');

   List<Map> list = await db.rawQuery('SELECT * FROM Contacts ORDER BY Name');
   this.setState(() {
```

```
    contacts = list;
    isLoading = false;
  });
}

String validateName(String value) {
  if (value.length <= 0 || value.length >= 30)
    return 'Please enter a non empty Name with less than 30 characters!';
  final alpha = RegExp(r'^[a-zA-Z ]+$');
  if (alpha.hasMatch(value)) return null;
  return 'Please enter a valid Name!';
}

String validateContact(String value) {
  if (value.length <= 9 || value.length >= 13)
    return 'Please enter valid Phone Number!';
  final numeric = RegExp(r'^[0-9+]+$');
  if (numeric.hasMatch(value)) return null;
  return 'Please enter valid Phone Number!';
}

String validateEmail(String value) {
  final emailRegEx = RegExp(
      r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+");
  if (emailRegEx.hasMatch(value)) return null;
  return 'Please enter valid Email!';
}

String validateAddress(String value) {
  if (value.length <= 0 || value.length > 70)
    return 'Enter non empty valid Address with at most 70 characters';
  final alphanumeric = RegExp(r'^[a-zA-Z 0-9,:-]+$');
  if (alphanumeric.hasMatch(value)) return null;
  return 'Please enter a valid Address!';
}

showToast(msg) {
  Fluttertoast.showToast(
      msg: msg,
      toastLength: Toast.LENGTH_SHORT,
      gravity: ToastGravity.BOTTOM,
      backgroundColor: Colors.purple,
      timeInSecForIosWeb: 1,
      fontSize: 16.0);
```

```
 }

 addEntry() async {
   if (validateName(name) != null ||
       validateEmail(email) != null ||
       validateContact(contact) != null ||
       validateAddress(address) != null) {
     showToast('Please enter valid Details!');
     return;
   }
   await db.transaction((txn) async {
     try {
       String query =
           "INSERT INTO Contacts(Name,Contact,Email,Address)
VALUES('$name','$contact','$email','$address');";
       await txn.rawInsert(query);
       showToast('Inserted Details in Database!');
       setState(() {
         contacts = List.from(contacts)
           ..add({
             'Name': name,
             'Contact': contact,
             'Address': address,
             'Email': email
           })
           ..sort((a, b) =>
               a['Name'].toLowerCase().compareTo(b['Name'].toLowerCase()));
         name = "";
         contact = "";
         address = "";
         email = "";
       });
       nameTextController.text = name;
       contactTextController.text = contact;
       emailTextController.text = email;
       addressTextController.text = address;
     } catch (e) {
       print(e);
       showToast(
           'Failed to Insert Details in Database! Make sure the Contact is unique!');
     }
   });
 }
```

```
updateEntry() async {
  if (validateName(name) != null ||
      validateEmail(email) != null ||
      validateContact(contact) != null ||
      validateAddress(address) != null) {
    showToast('Please enter valid Details to be updated!');
    return;
  }
  setState(() {
    isLoading = true;
  });

  var whereQuery = "Contact='${selections[0]}'";
  var setQuery =
      "NAME = '$name',CONTACT='$contact',EMAIL='$email',ADDRESS='$address' ";

  final count =
      await db.rawUpdate('UPDATE Contacts SET $setQuery WHERE $whereQuery');
  if (count == 0) {
    showToast('Failed to update Record! Please modify at least one value!');
    setState(() {
      isLoading = false;
    });
    return;
  }
  setState(() {
    selections = [];
    selectionMode = false;
    updateMode = false;
    name = '';
    address = '';
    email = '';
    contact = '';
  });
  nameTextController.text = name;
  contactTextController.text = contact;
  emailTextController.text = email;
  addressTextController.text = address;
  List<Map> list = await db.rawQuery('SELECT * FROM Contacts ORDER BY Name');
  setState(() {
    contacts = list;
    isLoading = false;
  });
  showToast('Record Updated Successfully!');
```

```
  return;
}

deleteEntries() async {
  setState(() {
    isLoading = true;
  });
  try {
    var inQuery = '';
    selections.forEach((contact) {
      if (inQuery.length == 0) {
        inQuery = "'$contact'";
      } else {
        inQuery += ",'$contact'";
      }
    });
    await db.rawDelete('DELETE FROM Contacts WHERE Contact IN ($inQuery)');
    List<Map> list =
        await db.rawQuery('SELECT * FROM Contacts ORDER BY Name');
    setState(() {
      contacts = list;
      selections = [];
      selectionMode = false;
      isLoading = false;
    });
    showToast('Deleted Contacts successfully!');
  } catch (e) {
    showToast('Failed to Delete Contacts!');
  }
}

deleteEntry(idx, contact) async {
  try {
    await db.rawDelete('DELETE FROM Contacts WHERE Contact="$contact"');
    setState(() {
      contacts = List.from(contacts)..removeAt(idx);
    });
    showToast('Deleted Contact successfully!');
  } catch (e) {
    showToast('Failed to Delete Contacts!');
  }
}

toggleSelection(idx) {
```

```
  if (!selectionMode) return;
  if (selections.contains(contacts[idx]['Contact']) == true) {
    setState(() {
      selections = List.from(selections)..remove(contacts[idx]['Contact']);
    });
    if (selections.length == 0) {
      setState(() {
        selectionMode = false;
      });
    }
  } else {
    setState(() {
      selections = List.from(selections)..add(contacts[idx]['Contact']);
    });
    if (selections.length == 1) {
      setState(() {
        updateMode = true;
        name = contacts[idx]['Name'];
        address = contacts[idx]['Address'];
        email = contacts[idx]['Email'];
        contact = contacts[idx]['Contact'];
      });
    } else {
      setState(() {
        updateMode = false;
        name = '';
        address = '';
        email = '';
        contact = '';
      });
    }
    nameTextController.text = name;
    contactTextController.text = contact;
    emailTextController.text = email;
    addressTextController.text = address;
  }
}

@override
void initState() {
  super.initState();
  fetchData();
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: Text('Contacts'),
        backgroundColor:
            selectionMode ? Colors.purple.shade700 : Colors.purple,
        actions: selectionMode
            ? [IconButton(icon: Icon(Icons.delete), onPressed: deleteEntries)]
            : null,
      ),
      body: SafeArea(
          child: isLoading
              ? Center(
                  child: CircularProgressIndicator(),
                )
              : SlidingUpPanel(
                  panel: Container(
                    color: Colors.transparent,
                    child: Column(children: [
                      Padding(
                        padding: EdgeInsets.all(10),
                        child: Center(
                            child: Text(
                                updateMode
                                    ? 'Update Contact'
                                    : 'Add new Contact',
                                style: GoogleFonts.lato(fontSize: 28))),
                      ),
                      Padding(
                          padding: EdgeInsets.only(
                              left: 150, right: 150, bottom: 20, top: 15),
                          child: Divider(
                            thickness: 3,
                          )),
                      Form(
                          key: _formKey,
                          autovalidate: true,
                          child: ListView(
                            shrinkWrap: true,
                            children: [
                              Padding(
                                padding: EdgeInsets.all(10),
                                child: TextFormField(
```

```
                controller: nameTextController,
                keyboardType: TextInputType.name,
                decoration: const InputDecoration(
                  icon: Icon(Icons.person),
                  border: OutlineInputBorder(),
                  hintText: 'What do people call you?',
                  labelText: 'Name *',
                ),
                validator: validateName,
                onChanged: (value) {
                  setState(() {
                    name = value;
                  });
                },
              ),
            ),
            Padding(
              padding: EdgeInsets.all(10),
              child: TextFormField(
                controller: contactTextController,
                keyboardType: TextInputType.number,
                decoration: const InputDecoration(
                  icon: Icon(Icons.phone),
                  border: OutlineInputBorder(),
                  hintText: 'Your Phone Number',
                  labelText: 'Contact Number *',
                ),
                validator: validateContact,
                onChanged: (value) {
                  setState(() {
                    contact = value;
                  });
                },
              ),
            ),
            Padding(
              padding: EdgeInsets.all(10),
              child: TextFormField(
                controller: emailTextController,
                keyboardType: TextInputType.emailAddress,
                decoration: const InputDecoration(
                  icon: Icon(Icons.email),
                  border: OutlineInputBorder(),
                  hintText: 'Your Email Address',
```

```
            labelText: 'Email Address *',
          ),
          validator: validateEmail,
          onChanged: (value) {
            setState(() {
              email = value;
            });
          },
        ),
      ),
      Padding(
        padding: EdgeInsets.all(10),
        child: TextFormField(
          controller: addressTextController,
          keyboardType: TextInputType.streetAddress,
          decoration: const InputDecoration(
            icon: Icon(Icons.pin_drop),
            border: OutlineInputBorder(),
            hintText: 'Your Address',
            labelText: 'Address *',
          ),
          validator: validateAddress,
          maxLines: 4,
          onChanged: (value) {
            setState(() {
              address = value;
            });
          },
        ),
      ),
      Padding(
          padding: EdgeInsets.all(10),
          child: FlatButton(
              color: Colors.purple,
              minWidth: 700,
              onPressed:
                  updateMode ? updateEntry : addEntry,
              child: Padding(
                  padding: EdgeInsets.all(5),
                  child: Text(
                      updateMode
                          ? 'Update Contact'
                          : 'Add Contact',
                      style: GoogleFonts.openSans(
```

```
                                    color: Colors.white,
                                    fontWeight: FontWeight.w700,
                                    fontSize: 16))))),
                          ],
                        ))
                  ]),
              ),
              maxHeight: 700,
              borderRadius: BorderRadius.only(
                  topLeft: Radius.circular(40.0),
                  topRight: Radius.circular(40.0)),
              body: contacts.length == 0
                  ? Center(
                      child: Text(
                          'No Contacts. Please create some Contacts.',
                          style: GoogleFonts.lato(fontSize: 16)))
                  : ListView.builder(
                      itemCount: contacts.length,
                      itemBuilder: (ctx, i) {
                        return GestureDetector(
                            onLongPress: () {
                              setState(() {
                                selectionMode = true;
                              });
                              toggleSelection(i);
                            },
                            onTap: () {
                              if (selectionMode) {
                                toggleSelection(i);
                                return;
                              }
                              Navigator.of(context).pushNamed(
                                  ChatScreen.routeName,
                                  arguments: {
                                    'Contact': contacts[i]['Contact']
                                  });
                            },
                            child: Dismissible(
                                direction: DismissDirection.startToEnd,
                                onDismissed: (direction) async {
                                  await deleteEntry(
                                      i, contacts[i]['Contact']);
                                },
                                background: Container(
```

```
                                    color: Colors.red.shade800,
                                    child: Row(children: [
                                      Padding(
                                          padding:
                                              EdgeInsets.only(left: 20),
                                          child: Icon(Icons.delete,
                                              color: Colors.white))
                                    ])),
                                key: Key(contacts[i]['Contact']),
                                child: Container(
                                    color: selectionMode &&
                                            selections.contains(
                                                contacts[i]['Contact'])
                                        ? Color(0xffDA70D6)
                                        : i % 2 == 0
                                            ? Color(0xffEFDFEF)
                                            : Colors.transparent,
                                    child: ListTile(
                                        title: Text(contacts[i]['Name'],
                                            style: GoogleFonts.lato(
                                                fontSize: 18)),
                                        subtitle:
                                            Text(contacts[i]['Contact']),
                                        leading: Container(
                                            decoration: BoxDecoration(
                                              color: i % 2 == 0
                                                  ? Colors.purple
                                                  : Colors.purpleAccent,
                                              shape: BoxShape.circle,
                                            ),
                                            child: CircleAvatar(
                                                child: Text(
                                                    contacts[i]['Name']
                                                        [0],
                                                    style: TextStyle(
                                                        color: Colors
                                                            .white)),
                                                backgroundColor: Colors
                                                    .transparent))))));
                        }),
                ));
 }
}
```

./lib/chat.dart:

```
import 'dart:math';

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:sqflite/sqflite.dart';

class ChatScreen extends StatefulWidget {
 static const routeName = '/chat';
 var contact;
 ChatScreen({@required this.contact});
 @override
 _ChatScreenState createState() => _ChatScreenState();
}

class _ChatScreenState extends State<ChatScreen> {
 bool isLoading = true, selectionMode = false;
 String name = "", contact = "", email = "", address = "";
 List<Map> messages = List<Map>();
 List selections = List();

 String messageContent = "";
 var messageTextController = new TextEditingController();
 final FocusNode focusNode = FocusNode();

 Database db;
 fetchData() async {
   setState(() {
     contact = widget.contact['Contact'];
   });
   db = await openDatabase('contacts.db');
   List<Map> list =
       await db.rawQuery('SELECT * FROM Contacts WHERE Contact="$contact"');
   List<Map> message = await db.rawQuery(
       'SELECT * FROM Messages WHERE Contact="$contact" ORDER BY Timestamp');
   this.setState(() {
     messages = message;
     name = list[0]['Name'];
     contact = list[0]['Contact'];
     address = list[0]['Address'];
     email = list[0]['Email'];
```

```
    isLoading = false;
  });
}

showToast(msg) {
  Fluttertoast.showToast(
      msg: msg,
      toastLength: Toast.LENGTH_SHORT,
      gravity: ToastGravity.BOTTOM,
      backgroundColor: Colors.purple,
      timeInSecForIosWeb: 1,
      fontSize: 16.0);
}

toggleSelection(idx) {
  if (!selectionMode) return;
  if (selections.contains(idx) == true) {
    setState(() {
      selections = List.from(selections)..remove(idx);
    });
    if (selections.length == 0) {
      setState(() {
        selectionMode = false;
      });
    }
  } else {
    setState(() {
      selections = List.from(selections)..add(idx);
    });
  }
}

addEntry() async {
  if (messageContent.length == 0) {
    showToast('Please enter a valid message!');
    return;
  }
  if (messageContent.length > 200) {
    showToast('Please enter a message that is at most 200 characters long!');
    return;
  }
  try {
    String timestamp = DateTime.now().millisecondsSinceEpoch.toString();
    String messageID = contact + timestamp;
```

```
    await db.rawInsert(
        "INSERT INTO Messages(Contact,MessageID,Timestamp,Content) VALUES(?,?,?,?)",
        [contact, messageID, timestamp, messageContent]);
    setState(() {
      messages = List.from(messages)
        ..add({
          'MessageID': messageID,
          'Timestamp': timestamp,
          'Content': messageContent,
          'Contact': contact
        });
      messageContent = '';
    });
    messageTextController.text = messageContent;
    showToast('Sent Message!');
  } catch (e) {
    print(e);
    showToast('Failed to send Message!');
  }
}

deleteEntries() async {
  if (!selectionMode) return;

  setState(() {
    isLoading = true;
  });
  try {
    String inQuery = '';
    selections.forEach((id) {
      var messageID = messages[id]['MessageID'];
      if (inQuery == '') {
        inQuery = "'$messageID'";
      } else {
        inQuery += ",'$messageID'";
      }
    });
    var count = await db
        .rawDelete('DELETE FROM Messages WHERE MessageID IN ($inQuery)');
    List<Map> message = await db.rawQuery(
        'SELECT * FROM Messages WHERE Contact="$contact" ORDER BY Timestamp');
    this.setState(() {
      messages = message;
      isLoading = false;
```

```
      selections = [];
      selectionMode = false;
    });
    showToast('Deleted $count Messages Successfully!');
  } catch (e) {
    print(e);
    this.setState(() {
      isLoading = false;
      selections = [];
      selectionMode = false;
    });
    showToast('Failed to Delete Messages!');
  }
}

@override
void initState() {
  super.initState();
  fetchData();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: isLoading
            ? Text('Chat')
            : selectionMode
                ? Text('Delete Messages')
                : Text(name),
        backgroundColor:
            selectionMode ? Colors.purple.shade700 : Colors.purple,
        actions: selectionMode
            ? [IconButton(icon: Icon(Icons.delete), onPressed: deleteEntries)]
            : null,
      ),
      body: SafeArea(
          child: isLoading
              ? Center(
                  child: CircularProgressIndicator(),
                )
              : Column(children: [
                  messages.length == 0
                      ? Flexible(
```

```
                    child: Center(
                        child: Text('No Messages.',
                            style: GoogleFonts.lato(fontSize: 16))))
                : Flexible(
                    child: ListView.builder(
                        itemCount: messages.length,
                        itemBuilder: (ctx, i) {
                          return GestureDetector(
                              onLongPress: () {
                                setState(() {
                                  selectionMode = true;
                                });
                                toggleSelection(i);
                              },
                              onTap: () {
                                if (selectionMode) toggleSelection(i);
                              },
                              child: Container(
                                  decoration: BoxDecoration(
                                    color: selectionMode &&
                                            selections.contains(i)
                                        ? Color(0xffDA70D6)
                                        : Colors.transparent,
                                  ),
                                  child: Container(
                                      padding: EdgeInsets.fromLTRB(
                                          10.0, 5.0, 10.0, 5.0),
                                      decoration: BoxDecoration(
                                          color: selectionMode &&
                                                  selections.contains(i)
                                              ? Colors.purple.shade700
                                              : Colors.purple,
                                          borderRadius:
                                              BorderRadius.circular(
                                                  10.0)),
                                      margin: EdgeInsets.only(
                                          top: 10,
                                          left: 100,
                                          bottom: 10.0,
                                          right: 10.0),
                                      child: ListTile(
                                        title: Text(
                                            messages[i]['Content'],
                                            textAlign: TextAlign.end,
```

```
                                                    style: GoogleFonts.lato(
                                                        fontSize: 14,
                                                        color: Colors.white)),
                                                subtitle: Text(
                                                    DateFormat('dd MMM HH:mm')
                                                        .format(DateTime

.fromMillisecondsSinceEpoch(
                                                                int.parse(messages[
                                                                        i][
                                                                    'Timestamp'])))),
                                                    style: GoogleFonts.lato(
                                                        fontSize: 14,
                                                        color: Colors.white70)),
                                            ))));
                                },
                            ),
                        Container(
                          child: Row(
                            children: <Widget>[
                              Flexible(
                                child: Container(
                                  child: TextField(
                                    style: TextStyle(fontSize: 15.0),
                                    controller: messageTextController,
                                    decoration: InputDecoration.collapsed(
                                      hintText: 'Type your message...',
                                      hintStyle:
                                          TextStyle(color: Color(0xffaeaeae)),
                                    ),
                                    onChanged: (val) {
                                      setState(() {
                                        messageContent = val;
                                      });
                                    },
                                    focusNode: focusNode,
                                  ),
                                ),
                              ),
                              Material(
                                child: Container(
                                  margin: EdgeInsets.symmetric(horizontal: 8.0),
                                  child: IconButton(
                                    icon: Icon(Icons.send),
                                    onPressed: addEntry,
```

```
                              color: Colors.purple,
                            ),
                          ),
                        ),
                      ],
                    ),
                    width: double.infinity,
                    height: 50.0,
                    decoration: BoxDecoration(
                      border: Border(top: BorderSide(width: 0.5)),
                    ),
                  )
                ])));
  }
}
```

Screenshots: