
PRAKTIKUM_8

Rizal Fadlullah - A11.2019.12070

Contents

1	Tugas_1	2
1.1	lib/repositories/repositories.dart	2
1.2	lib/repositories/user_repository.dart	2
1.3	lib/models/user.dart	3
1.4	lib/models/models.dart	4
1.5	lib/main.dart	4
1.6	lib/views/register_page.dart	4
1.7	lib/views/main_page.dart	6
1.8	lib/views/login_page.dart	7
1.9	lib/views/views.dart	9
1.10	lib/services/db_service.dart	9
1.11	lib/services/auth_service.dart	10
1.12	lib/services/services.dart	11
1.13	OUTPUT	13
2	Latihan_SQLite	14
2.1	lib/database/db_helper.dart	14
2.2	lib/models/contact.dart	15
2.3	lib/main.dart	16
2.4	lib/add_contact_page.dart	16
2.5	lib/contact_page.dart	19
2.6	OUTPUT	24
3	Tugas_2	25
3.1	lib/repositories/repositories.dart	25
3.2	lib/repositories/product_repository.dart	25
3.3	lib/models/product.dart	26
3.4	lib/models/models.dart	27
3.5	lib/main.dart	27
3.6	lib/views/home_page.dart	28
3.7	lib/views/product_form_page.dart	28
3.8	lib/views/views.dart	29
3.9	lib/services/db_service.dart	29
3.10	lib/services/services.dart	30
3.11	OUTPUT	30

1 Tugas_1

1.1 lib/repositories/repositories.dart

```
1 export 'user_repository.dart';
```

1.2 lib/repositories/user_repository.dart

```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:tugas_1/models/models.dart';
3 import 'package:tugas_1/services/services.dart';
4
5 class UserRepository {
6   final tableName = 'users';
7
8   Future<User?> getUserByUsername(String username) async {
9     final db = await DBService.getDatabase();
10    final result = await db.query(
11      tableName,
12      columns: [
13        'id',
14        'fullname',
15        'username',
16        'password',
17      ],
18      where: 'username = ?',
19      whereArgs: [username],
20    );
21
22    db.close();
23
24    if (result.isNotEmpty) {
25      return User.fromMap(result.first);
26    } else {
27      return null;
28    }
29  }
30
31  Future<void> addUser(User newUser) async {
32    final db = await DBService.getDatabase();
33    await db.insert(
34      tableName,
35      newUser.toMap(),
36      conflictAlgorithm: ConflictAlgorithm.replace,
37    );
38    db.close();
39  }
40 }
```

1.3 lib/models/user.dart

```
1 class User {
2   int? _id;
3   String? fullname;
4   String username;
5   String password;
6
7   User({
8     this.fullname,
9     required this.username,
10    required this.password,
11    int? id,
12  }) : _id = id;
13
14  int? get id => _id;
15
16  set id(int? id) {
17    if (_id == null) {
18      _id = id;
19    } else {
20      throw Exception('ID sudah diatur dan tidak bisa diubah lagi.');
```

1.4 lib/models/models.dart

```
1 export 'user.dart';
```

1.5 lib/main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:tugas_1/views/views.dart';
3
4 void main() {
5   runApp(const App());
6 }
7
8 class App extends StatelessWidget {
9   const App({super.key});
10
11   @override
12   Widget build(BuildContext context) {
13     return MaterialApp(
14       initialRoute: '/login',
15       routes: {
16         '/login': (context) => const LoginPage(),
17         '/register': (context) => const RegisterPage(),
18         '/main': (context) => const MainPage(),
19       },
20     );
21   }
22 }
```

1.6 lib/views/register_page.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:tugas_1/services/services.dart';
3
4 class RegisterPage extends StatefulWidget {
5   const RegisterPage({super.key});
6
7   @override
8   State<RegisterPage> createState() => _RegisterPageState();
9 }
10
11 class _RegisterPageState extends State<RegisterPage> {
12   final fullnameController = TextEditingController();
13   final usernameController = TextEditingController();
14   final passwordController = TextEditingController();
15   final auth = AuthService();
```

```
16
17 void _registerButtonAction() async {
18   final fullname = fullnameController.text.trim();
19   final username = usernameController.text.trim();
20   final password = passwordController.text.trim();
21   final success = await auth.register(fullname, username, password);
22   debugPrint(auth.currentUser.toString());
23
24   if (success && mounted) {
25     _showSnackBar(content: 'Registrasi berhasil. Mengalihkan ke
26       halaman home.');
```

Navigator.of(context).pushReplacementNamed('/main');

```
27   } else {
28     _showSnackBar(content: 'Register gagal.');
```

}

```
30 }
31
32 void _loginButtonAction() {
33   Navigator.pushReplacementNamed(context, '/login');
```

}

```
35
36 void _showSnackBar({required String content}) {
37   ScaffoldMessenger.of(context).showSnackBar(
38     SnackBar(content: Text(content)),
39   );
40 }
41
42 @override
43 Widget build(BuildContext context) {
44   return Scaffold(
45     appBar: AppBar(
46       title: const Text('Halaman Register'),
47     ),
48     body: SafeArea(
49       child: Padding(
50         padding: const EdgeInsets.all(16.0),
51         child: Column(
52           children: [
53             TextField(
54               controller: fullnameController,
55               decoration: const InputDecoration(
56                 labelText: 'Fullname',
57                 border: OutlineInputBorder(),
58             ),
59           ),
60           const SizedBox(height: 16),
61             TextField(
62               controller: usernameController,
63               decoration: const InputDecoration(
64                 labelText: 'Username',
65                 border: OutlineInputBorder(),
```

```
66         ),
67     ),
68     const SizedBox(height: 16),
69     TextField(
70       controller: passwordController,
71       obscureText: true,
72       decoration: const InputDecoration(
73         labelText: 'Password',
74         border: OutlineInputBorder(),
75       ),
76     ),
77     const SizedBox(height: 16),
78     ElevatedButton(
79       onPressed: _registerButtonAction,
80       child: const Text('Daftar'),
81     ),
82     const SizedBox(height: 16),
83     TextButton(
84       onPressed: _loginButtonAction,
85       child: const Text('Sudah punya akun? Login!'),
86     ),
87   ],
88 ),
89 ),
90 ),
91 );
92 }
93 }
```

1.7 lib/views/main_page.dart

```
1  import 'package:flutter/material.dart';
2  import 'package:tugas_1/services/services.dart';
3
4  class MainPage extends StatefulWidget {
5    const MainPage({super.key});
6
7    @override
8    State<MainPage> createState() => _MainPageState();
9  }
10
11  class _MainPageState extends State<MainPage> {
12    final auth = AuthService();
13
14    void _logoutButtonAction() {
15      final success = auth.logout();
16      if (success) {
17        Navigator.pushReplacementNamed(context, '/login');
18      }
19    }
20  }
```

```
19   }
20
21   @override
22   Widget build(BuildContext context) {
23     return Scaffold(
24       body: Center(
25         child: Column(
26           mainAxisAlignment: MainAxisAlignment.center,
27           children: [
28             Text(
29               'Selamat datang ${auth.currentUser!.fullname}, Ini
                halaman home'),
30             ElevatedButton(
31               onPressed: _logoutButtonAction,
32               child: const Text('Logout'),
33             ),
34           ],
35         ),
36       ),
37     );
38   }
39 }
```

1.8 lib/views/login_page.dart

```
1  import 'package:flutter/material.dart';
2  import 'package:tugas_1/services/services.dart';
3
4  class LoginPage extends StatefulWidget {
5    const LoginPage({super.key});
6
7    @override
8    State<LoginPage> createState() => _LoginPageState();
9  }
10
11  class _LoginPageState extends State<LoginPage> {
12    final usernameController = TextEditingController();
13    final passwordController = TextEditingController();
14    final auth = AuthService();
15
16    void _loginButtonAction() async {
17      final username = usernameController.text.trim();
18      final password = passwordController.text.trim();
19      final success = await auth.login(username, password);
20
21      if (success && mounted) {
22        Navigator.of(context).pushReplacementNamed('/main');
23      } else {
```



```
24     _showSnackBar(content: 'Login gagal. Periksa kembali kredensial
      Anda.');
```

```
25   }
26 }
27
28 void _registerButtonAction() {
29   Navigator.pushReplacementNamed(context, '/register');
30 }
31
32 void _showSnackBar({required String content}) {
33   ScaffoldMessenger.of(context).showSnackBar(
34     SnackBar(content: Text(content)),
35   );
36 }
37
38 @override
39 Widget build(BuildContext context) {
40   return Scaffold(
41     appBar: AppBar(
42       title: const Text('Halaman Login'),
43     ),
44     body: SafeArea(
45       child: Padding(
46         padding: const EdgeInsets.all(16.0),
47         child: Column(
48           children: [
49             TextField(
50               controller: usernameController,
51               decoration: const InputDecoration(
52                 labelText: 'Username',
53                 border: OutlineInputBorder(),
54               ),
55             ),
56             const SizedBox(height: 16),
57             TextField(
58               controller: passwordController,
59               obscureText: true,
60               decoration: const InputDecoration(
61                 labelText: 'Password',
62                 border: OutlineInputBorder(),
63               ),
64             ),
65             const SizedBox(height: 16),
66             ElevatedButton(
67               onPressed: _loginButtonAction,
68               child: const Text('Login'),
69             ),
70             const SizedBox(height: 16),
71             TextButton(
72               onPressed: _registerButtonAction,
73               child: const Text('Belum Punya Akun? Daftar!'),
```

```
74         ),
75       ],
76     ),
77   ),
78 ),
79 );
80 }
81 }
```

1.9 lib/views/views.dart

```
1 export 'login_page.dart';
2 export 'main_page.dart';
3 export 'register_page.dart';
```

1.10 lib/services/db_service.dart

```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:path/path.dart';
3
4 class DBService {
5   static Database? _database;
6
7   static Future<Database> getDatabase() async {
8     if (_database != null) return _database!;
9
10    _database = await _initDB();
11    return _database!;
12  }
13
14  static Future<Database> _initDB() async {
15    final dbPath = await getDatabasesPath();
16    final path = join(dbPath, 'app_database.db');
17
18    return await openDatabase(
19      path,
20      version: 1,
21      onCreate: _onCreate,
22    );
23  }
24
25  static Future<void> _onCreate(Database db, int version) async {
26    await db.execute('''
27      CREATE TABLE users(
28        id INTEGER PRIMARY KEY AUTOINCREMENT,
29        fullname TEXT,
30        username TEXT,
```

```
31         password TEXT
32     )
33     ''');
34 }
35
36 static Future<void> close() async {
37     final db = await getDatabase();
38     db.close();
39 }
40 }
```

1.11 lib/services/auth_service.dart

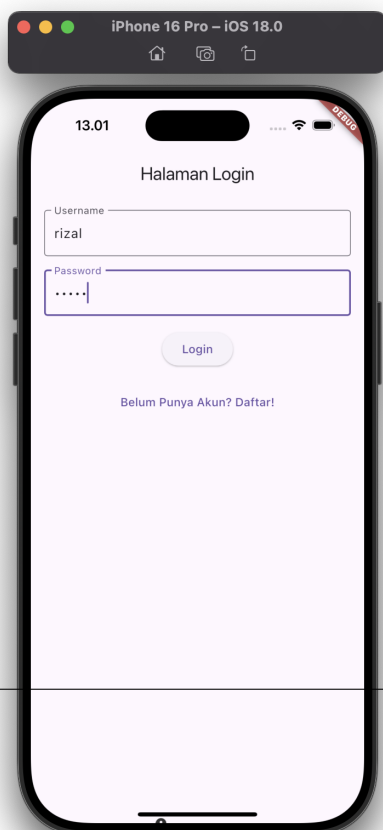
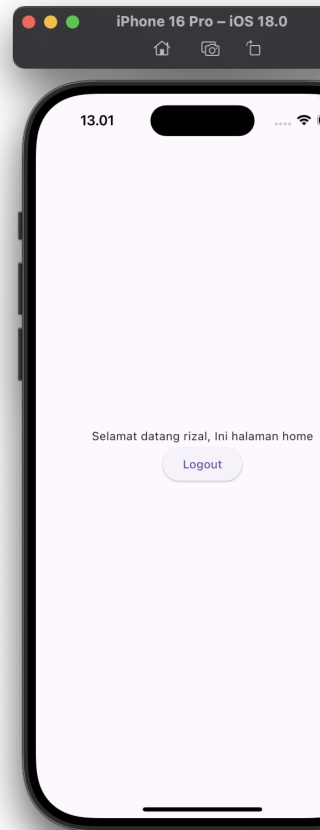
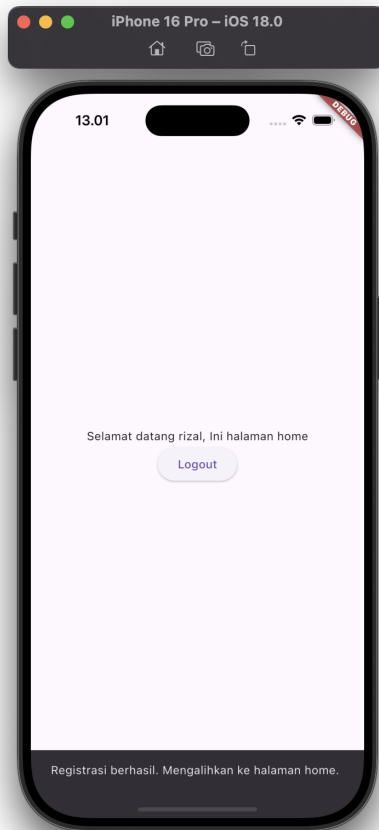
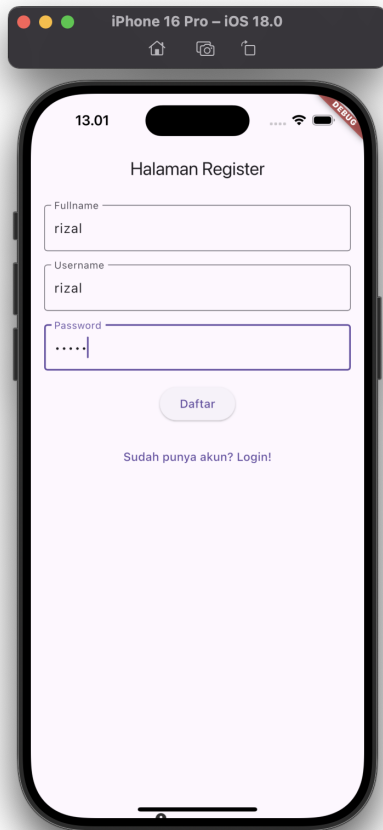
```
1 import 'dart:convert';
2 import 'package:crypto/crypto.dart';
3 import 'package:tugas_1/repositories/repositories.dart';
4 import 'package:tugas_1/models/models.dart';
5
6 class AuthService {
7     static final AuthService _instance = AuthService._internal();
8     factory AuthService() => _instance;
9     AuthService._internal();
10
11     final UserRepository _userRepository = UserRepository();
12
13     User? _currentUser;
14
15     User? get currentUser => _currentUser;
16
17     String _hashPassword(String password) {
18         final bytes = utf8.encode(password);
19         final digest = sha256.convert(bytes);
20         return digest.toString();
21     }
22
23     Future<bool> login(String username, String password) async {
24         final hashedPassword = _hashPassword(password);
25         final user = await _userRepository.getUserByUsername(username);
26
27         if (user != null && user.password == hashedPassword) {
28             _currentUser = user;
29             return true;
30         }
31
32         return false;
33     }
34
35     Future<bool> register(
36         String fullname,
```

```
37   String username,  
38   String password,  
39 ) async {  
40   final existingUser = await _userRepository.getUserByUsername(  
41     username);  
42   existingUser ?? false;  
43  
44   final newUser = User(  
45     fullname: fullname,  
46     username: username,  
47     password: _hashPassword(password),  
48   );  
49   await _userRepository.addUser(newUser);  
50   _currentUser = newUser;  
51  
52   return true;  
53 }  
54  
55 bool logout() {  
56   _currentUser = null;  
57   return true;  
58 }  
59  
60 bool isLoggedIn() {  
61   return _currentUser != null;  
62 }  
63 }
```

1.12 lib/services/services.dart

```
1 export 'auth_service.dart';  
2 export 'db_service.dart';
```


1.13 OUTPUT



2 Latihan_SQLite

2.1 lib/database/db_helper.dart

```
1 import 'package:flutter_aplication_sqlite/models/contact.dart';
2 import 'package:sqflite/sqflite.dart';
3 import 'package:path/path.dart' as p;
4
5 class DbHelper {
6   static final DbHelper _instance = DbHelper._internal();
7   static Database? _database;
8
9   final String tableName = 'tablecontact';
10  final String columnId = 'id';
11  final String columnName = 'name';
12  final String columnPhone = 'phone';
13  final String columnEmail = 'email';
14  final String columnCompany = 'company';
15  DbHelper._internal();
16  factory DbHelper() => _instance;
17
18  Future<Database?> get _db async {
19    if (_database != null) {
20      return _database;
21    }
22    _database = await _initDb();
23    return _database;
24  }
25
26  Future<Database?> _initDb() async {
27    String databasePath = await getDatabasesPath();
28    String path = p.join(databasePath, 'contact.db');
29    return await openDatabase(path, version: 1, onCreate: _onCreate);
30  }
31
32  Future<void> _onCreate(Database db, int version) async {
33    var sql = "CREATE TABLE $tableName($columnId INTEGER PRIMARY KEY, "
34              "$columnName TEXT,"
35              "$columnPhone TEXT,"
36              "$columnEmail TEXT,"
37              "$columnCompany TEXT)";
38    await db.execute(sql);
39  }
40
41  Future<int?> savecontact(Contact contact) async {
42    var dbClient = await _db;
43    return await dbClient!.insert(tableName, contact.toMap());
44  }
45
46  Future<List?> getAllcontact() async {
```

```
47     var dbClient = await _db;
48     var result = await dbClient!.query(tableName, columns: [
49         columnId,
50         columnName,
51         columnCompany,
52         columnPhone,
53         columnEmail
54     ]);
55     return result.toList();
56 }
57
58 Future<int?> updatecontact(Contact contact) async {
59     var dbClient = await _db;
60     return await dbClient!.update(tableName, contact.toMap(),
61         where: '$columnId = ?', whereArgs: [contact.id]);
62 }
63
64 Future<int?> deletecontact(int id) async {
65     var dbClient = await _db;
66     return await dbClient!
67         .delete(tableName, where: '$columnId = ?', whereArgs: [id]);
68 }
69 }
```

2.2 lib/models/contact.dart

```
1  class Contact {
2      int? id;
3      String? name;
4      String? phone;
5      String? email;
6      String? company;
7
8      Contact({
9          this.id,
10         this.name,
11         this.phone,
12         this.email,
13         this.company,
14     });
15
16     Map<String, dynamic> toMap() {
17         var map = <String, dynamic>{};
18         if (id != null) {
19             map['id'] = id;
20         }
21         map['name'] = name;
22         map['phone'] = phone;
23         map['email'] = email;
```



```
24     map['company'] = company;
25     return map;
26   }
27
28   Contact.fromMap(Map<String, dynamic> map) {
29     id = map['id'];
30     name = map['name'];
31     phone = map['phone'];
32     email = map['email'];
33     company = map['company'];
34   }
35 }
```

2.3 lib/main.dart

```
1  import 'package:flutter/material.dart';
2  import 'package:flutter_aplication_sqlite/contact_page.dart';
3
4  void main() {
5    runApp(const MyApp());
6  }
7
8  class MyApp extends StatelessWidget {
9    const MyApp({super.key});
10   @override
11   Widget build(BuildContext context) {
12     return const MaterialApp(
13       debugShowCheckedModeBanner: false,
14       title: 'Flutter Demo',
15       home: ContactPage(),
16     );
17   }
18 }
```

2.4 lib/add_contact_page.dart

```
1  // ignore_for_file: prefer_const_constructors_in_immutables
2  import 'package:flutter/material.dart';
3  import 'package:flutter_aplication_sqlite/database/db_helper.dart';
4  import 'package:flutter_aplication_sqlite/models/contact.dart';
5
6  class AddContactPage extends StatefulWidget {
7    final Contact? contact;
8
9    AddContactPage({super.key, this.contact});
10
11   @override
```

```
12   State<AddContactPage> createState() => _AddContactPageState();
13 }
14
15 class _AddContactPageState extends State<AddContactPage> {
16   final DbHelper db = DbHelper();
17
18   late TextEditingController nameController;
19   late TextEditingController phoneController;
20   late TextEditingController emailController;
21   late TextEditingController companyController;
22
23   @override
24   void initState() {
25     super.initState();
26     nameController = TextEditingController(
27       text: widget.contact == null ? '' : widget.contact!.name);
28     phoneController = TextEditingController(
29       text: widget.contact == null ? '' : widget.contact!.phone);
30     emailController = TextEditingController(
31       text: widget.contact == null ? '' : widget.contact!.email);
32     companyController = TextEditingController(
33       text: widget.contact == null ? '' : widget.contact!.company);
34   }
35
36   @override
37   void dispose() {
38     nameController.dispose();
39     phoneController.dispose();
40     emailController.dispose();
41     companyController.dispose();
42     super.dispose();
43   }
44
45   @override
46   Widget build(BuildContext context) {
47     return Scaffold(
48       appBar: AppBar(
49         title: const Text('Form Contact'),
50       ),
51       body: ListView(
52         padding: const EdgeInsets.all(16.0),
53         children: [
54           Padding(
55             padding: const EdgeInsets.only(top: 20),
56             child: TextField(
57               controller: nameController,
58               decoration: InputDecoration(
59                 labelText: 'Name',
60                 border: OutlineInputBorder(
61                   borderRadius: BorderRadius.circular(8),
62                 ),
```

```
63         ),
64     ),
65 ),
66 Padding(
67     padding: const EdgeInsets.only(top: 20),
68     child: TextField(
69         controller: phoneController,
70         decoration: InputDecoration(
71             labelText: 'Mobile No',
72             border: OutlineInputBorder(
73                 borderRadius: BorderRadius.circular(8),
74             ),
75         ),
76     ),
77 ),
78 Padding(
79     padding: const EdgeInsets.only(top: 20),
80     child: TextField(
81         controller: emailController,
82         decoration: InputDecoration(
83             labelText: 'Email',
84             border: OutlineInputBorder(
85                 borderRadius: BorderRadius.circular(8),
86             ),
87         ),
88     ),
89 ),
90 Padding(
91     padding: const EdgeInsets.only(top: 20),
92     child: TextField(
93         controller: companyController,
94         decoration: InputDecoration(
95             labelText: 'Company',
96             border: OutlineInputBorder(
97                 borderRadius: BorderRadius.circular(8),
98             ),
99         ),
100     ),
101 ),
102 Padding(
103     padding: const EdgeInsets.only(top: 20),
104     child: ElevatedButton(
105         onPressed: () => _upsertContact(),
106         child: Text(widget.contact == null ? 'Add' : 'Update'),
107     ),
108 ),
109 ],
110 ),
111 );
112 }
113
```

```
114 Future<void> _upsertContact() async {
115   if (widget.contact != null) {
116     // Update contact
117     await db.updatecontact(Contact(
118       id: widget.contact!.id,
119       name: nameController.text,
120       phone: phoneController.text,
121       email: emailController.text,
122       company: companyController.text,
123     ));
124
125     if (mounted) {
126       // Gunakan context jika widget masih aktif
127       Navigator.pop(context, 'update');
128     }
129   } else {
130     // Add new contact
131     await db.savecontact(Contact(
132       name: nameController.text,
133       phone: phoneController.text,
134       email: emailController.text,
135       company: companyController.text,
136     ));
137
138     if (mounted) {
139       // Gunakan context jika widget masih aktif
140       Navigator.pop(context, 'save');
141     }
142   }
143 }
144
145 }
```

2.5 lib/contact_page.dart

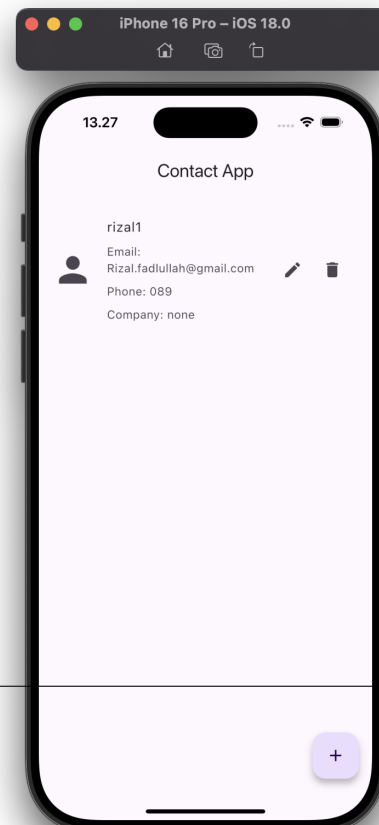
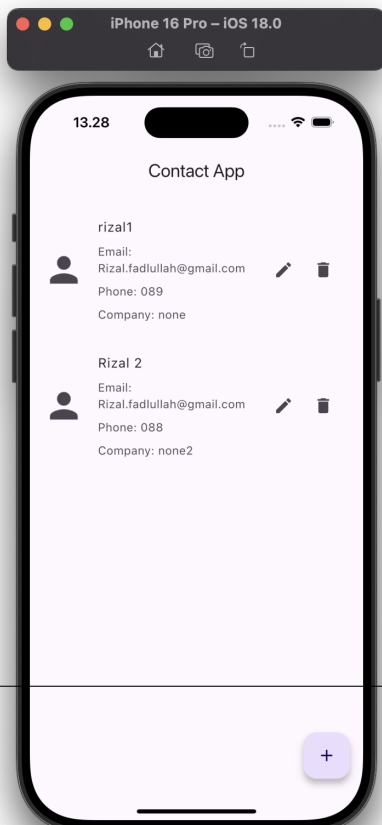
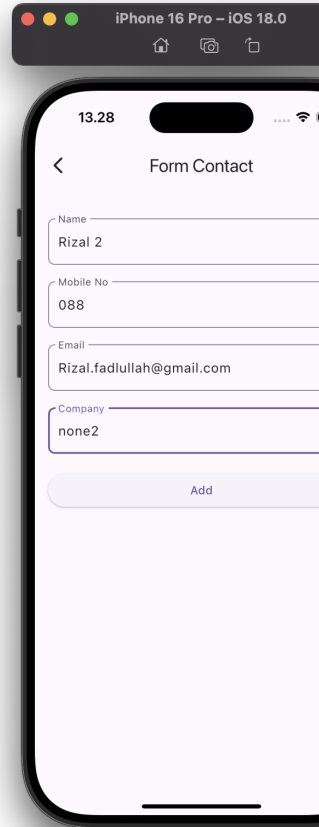
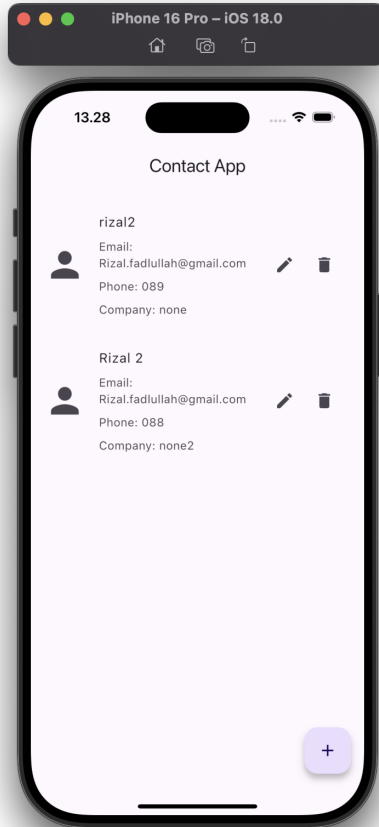
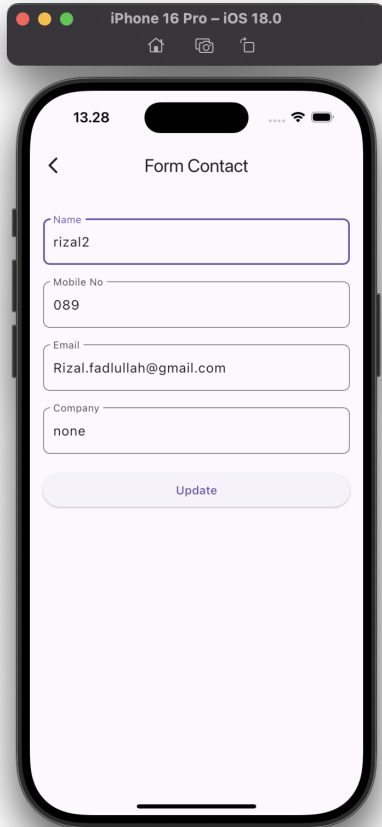
```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_aplication_sqlite/add_contact_page.dart';
3 import 'package:flutter_aplication_sqlite/database/db_helper.dart';
4 import 'package:flutter_aplication_sqlite/models/contact.dart';
5
6 class ContactPage extends StatefulWidget {
7   const ContactPage({super.key});
8
9   @override
10  State<ContactPage> createState() => _ContactPageState();
11 }
12
13 class _ContactPageState extends State<ContactPage> {
14   List<Contact> listContact = [];
```

```
15  final DBHelper db = DBHelper();
16
17  @override
18  void initState() {
19    super.initState();
20    // Menjalankan fungsi getAllContact saat pertama kali dimuat
21    _getAllContact();
22  }
23
24  @override
25  Widget build(BuildContext context) {
26    return Scaffold(
27      appBar: AppBar(
28        title: const Center(
29          child: Text("Contact App"),
30        ),
31      ),
32      body: ListView.builder(
33        itemCount: listContact.length,
34        itemBuilder: (context, index) {
35          Contact contact = listContact[index];
36          return Padding(
37            padding: const EdgeInsets.only(top: 20),
38            child: ListTile(
39              leading: const Icon(
40                Icons.person,
41                size: 50,
42              ),
43              title: Text(contact.name ?? ''),
44              subtitle: Column(
45                crossAxisAlignment: CrossAxisAlignment.start,
46                children: [
47                  const SizedBox(height: 8),
48                  Text("Email: ${contact.email ?? '-'}"),
49                  const SizedBox(height: 8),
50                  Text("Phone: ${contact.phone ?? '-'}"),
51                  const SizedBox(height: 8),
52                  Text("Company: ${contact.company ?? '-'}"),
53                ],
54              ),
55              trailing: FittedBox(
56                fit: BoxFit.fill,
57                child: Row(
58                  children: [
59                    // Button edit
60                    IconButton(
61                      onPressed: () {
62                        _openFormEdit(contact);
63                      },
64                      icon: const Icon(Icons.edit),
65                    ),
```

```
66             // Button hapus
67             IconButton(
68                 icon: const Icon(Icons.delete),
69                 onPressed: () {
70                     _showDeleteConfirmation(contact, index);
71                 },
72             ),
73         ],
74     ),
75 ),
76 ),
77 );
78 },
79 ),
80 // Membuat button mengapung di bagian bawah kanan layar
81 floatingActionButton: FloatingActionButton(
82     onPressed: _openFormCreate,
83     child: const Icon(Icons.add),
84 ),
85 );
86 }
87
88 Future<void> _getAllContact() async {
89     var list = await db.getAllcontact();
90     setState(() {
91         listContact.clear();
92         for (var contact in list ?? []) {
93             listContact.add(Contact.fromMap(contact));
94         }
95     });
96 }
97
98 Future<void> _deleteContact(Contact contact, int position) async {
99     await db.deletecontact(contact.id!);
100     setState(() {
101         listContact.removeAt(position);
102     });
103 }
104
105 Future<void> _openFormCreate() async {
106     var result = await Navigator.push(
107         context,
108         MaterialPageRoute(builder: (context) => AddContactPage()),
109     );
110     if (result == 'save') {
111         await _getAllContact();
112     }
113 }
114
115 Future<void> _openFormEdit(Contact contact) async {
116     var result = await Navigator.push(
```

```
117     context,
118     MaterialPageRoute(
119       builder: (context) => AddContactPage(contact: contact),
120     ),
121   );
122   if (result == 'update') {
123     await _getAllContact();
124   }
125 }
126
127 void _showDeleteConfirmation(Contact contact, int index) {
128   showDialog(
129     context: context,
130     builder: (context) {
131       return AlertDialog(
132         title: const Text("Information"),
133         content: Text("Yakin ingin menghapus data\n\n${contact.name}?"),
134         actions: [
135           TextButton(
136             onPressed: () {
137               _deleteContact(contact, index);
138               Navigator.pop(context);
139             },
140             child: const Text("Ya"),
141           ),
142           TextButton(
143             onPressed: () {
144               Navigator.pop(context);
145             },
146             child: const Text("Tidak"),
147           ),
148         ],
149       );
150     },
151   );
152 }
153 }
```


2.6 OUTPUT



3 Tugas_2

3.1 lib/repositories/repositories.dart

```
1 export 'repositories.dart';
```

3.2 lib/repositories/product_repository.dart

```
1 import 'package:inventory_app/services/services.dart';
2 import 'package:inventory_app/models/models.dart';
3 import 'package:sqflite/sqflite.dart';
4
5 class ProductRepository {
6   final tableName = 'products';
7
8   Future<void> addProduct(Product newProduct) async {
9     final db = await DBService.getDatabase();
10    final id = await db.insert(
11      tableName,
12      newProduct.toMap(),
13      conflictAlgorithm: ConflictAlgorithm.replace,
14    );
15    db.close();
16    newProduct.id = id;
17  }
18
19  Future<List<Product>?> getAllProduct() async {
20    final db = await DBService.getDatabase();
21    final result = await db.query(tableName, columns: [
22      'id',
23      'name',
24      'purchase_price',
25      'selling_price',
26      'stock',
27    ]);
28    db.close();
29
30    if (result.isNotEmpty) {
31      return result.map((productMap) => Product.fromMap(productMap)).
32        toList();
33    } else {
34      return null;
35    }
36  }
37
38  Future<int?> updateProduct(Product product) async {
39    final db = await DBService.getDatabase();
40    final result = await db.update(
```

```
40     tableName,
41     product.toMap(),
42     where: 'id = ?',
43     whereArgs: [product.id.toString()],
44   );
45   db.close();
46
47   return result;
48 }
49
50 Future<int?> deleteProduct(Product product) async {
51   final db = await DBService.getDatabase();
52   final result = await db.delete(
53     tableName,
54     where: 'id = ?',
55     whereArgs: [product.id.toString()],
56   );
57   db.close();
58
59   return result;
60 }
61 }
```

3.3 lib/models/product.dart

```
1  class Product {
2    int? _id;
3    String name;
4    int purchasePrice;
5    int sellingPrice;
6    int stock;
7
8    Product({
9      int? id,
10     required this.name,
11     required this.purchasePrice,
12     required this.sellingPrice,
13     required this.stock,
14   }) : _id = id;
15
16   int? get id => _id;
17
18   set id(int? id) {
19     if (_id == null) {
20       _id = id;
21     } else {
22       throw Exception('ID sudah diatur dan tidak bisa dirubah lagi');
23     }
24   }
25 }
```

```
25
26   Map<String, dynamic> toMap() {
27     return {
28       'id': _id,
29       'name': name,
30       'purchasePrice': purchasePrice,
31       'sellingPrice': sellingPrice,
32       'stock': stock,
33     };
34   }
35
36   factory Product.fromMap(Map<String, dynamic> map) {
37     return Product(
38       name: map['name'],
39       purchasePrice: map['purchaseSell'],
40       sellingPrice: map['sellingPrice'],
41       stock: map['stock'],
42       id: map['id'],
43     );
44   }
45 }
```

3.4 lib/models/models.dart

```
1 export 'product.dart';
```

3.5 lib/main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:inventory_app/views/views.dart';
3
4 void main() {
5   runApp(const App());
6 }
7
8 class App extends StatefulWidget {
9   const App({super.key});
10
11   @override
12   State<App> createState() => _AppState();
13 }
14
15 class _AppState extends State<App> {
16   @override
17   Widget build(BuildContext context) {
18     return MaterialApp(
19       initialRoute: '/home',
```

```
20     routes: {
21       '/home': (context) => const HomePage(),
22       '/product_form_page': (context) => const ProductFormPage(),
23     },
24   );
25 }
26 }
```

3.6 lib/views/home_page.dart

```
1 import 'package:flutter/material.dart';
2
3 class HomePage extends StatefulWidget {
4   const HomePage({super.key});
5
6   @override
7   State<HomePage> createState() => _HomePageState();
8 }
9
10 class _HomePageState extends State<HomePage> {
11   @override
12   Widget build(BuildContext context) {
13     return const Placeholder();
14   }
15 }
```

3.7 lib/views/product_form_page.dart

```
1 import 'package:flutter/material.dart';
2
3 class ProductFormPage extends StatefulWidget {
4   const ProductFormPage({super.key});
5
6   @override
7   State<ProductFormPage> createState() => _ProductFormPageState();
8 }
9
10 class _ProductFormPageState extends State<ProductFormPage> {
11   @override
12   Widget build(BuildContext context) {
13     return const Placeholder();
14   }
15 }
```

3.8 lib/views/views.dart

```
1 export 'home_page.dart';
2 export 'product_form_page.dart';
```

3.9 lib/services/db_service.dart

```
1 import 'package:sqflite/sqflite.dart';
2 import 'package:path/path.dart';
3
4 class DBService {
5   static Database? _database;
6
7   static Future<Database> getDatabase() async {
8     if (_database != null) return _database!;
9
10    _database = await _initDB();
11    return _database!;
12  }
13
14  static Future<Database> _initDB() async {
15    final dbPath = await getDatabasesPath();
16    final path = join(dbPath, 'app_database.db');
17
18    return await openDatabase(
19      path,
20      version: 1,
21      onCreate: _onCreate,
22    );
23  }
24
25  static Future<void> _onCreate(Database db, int version) async {
26    await db.execute('''
27      CREATE TABLE products(
28        id INTEGER PRIMARY KEY AUTOINCREMENT,
29        name TEXT,
30        purchase_price INTEGER,
31        selling_price INTEGER,
32        stock INTEGER
33      )
34    ''');
35  }
36
37  static Future<void> close() async {
38    final db = await getDatabase();
39    db.close();
40  }
41 }
```

3.10 lib/services/services.dart

```
1 export 'db_service.dart';
```

3.11 OUTPUT