**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY**

(Deemed to be University)

Patiala, Punjab

# A Mini Project On

**"Mobile APP Development Activity"**

**For the complement of CSED-Experiential Learning Activities-E110-2023**

**Under the supervision of**
**Department of Computer Science and Engineering**

**Submitted By:**

**"Sajid Miya"**

**102367013**

**Submitted To:**

**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY**
**Department of Computer Science and Engineering**
**Patiala, Punjab, India**
**"June 2024"**

## ACKNOWLEDGEMENT

We express our deep sense of gratitude to all those who have an immense help in completing this project. We are grateful for opportunity the opportunity to the research and explore mobile app development and all those necessary concepts involved in it.

We would like to convey our special thanks to the Computer Science and Engineering Department of Thapar Institute of Engineering and Technology for providing us with this opportunity. Their support and resources were invaluable throughout the project.

We are also thankful to our friends and teammates for their immeasurable help a collaboration in the completion of this project. This project was a team effort, and we are proud to have worked together to achieve the required results.

Lastly, we acknowledge all of the websites and video on the internet which were of great use in completing of the codes and acquiring the necessary knowledge.

Thank You!

--------------------
Name: Sajid Miya
Level: B.E./B.Tech (1st year)
Enrollment Id: 102367013
Date: June, 2024

## ABSTRACT

This project presents the development of a mobile application using Flutter for the app development and SQLite for database management, with MySQL as the query language. The app provides an intuitive and user-friendly interface for users to write and manage their notes. Additionally, a login page has been added to ensure that only registered users can access and check their content. The interactivity of the app is enhanced through Flutter’s rich set of widgets, which allow for real-time validation and error messaging. SQLite is used to create a robust and secure local database-driven system that enables efficient storage, retrieval, and management of user notes. The project demonstrates the effective use of mobile development technologies in creating a dynamic and functional note-taking application. Future work could involve improving the app's scalability and further enhancing its security features. Overall, the mobile app project provides a valuable learning experience and equips students with practical skills in mobile app development and database management.

## ABBREVIATIONS:

1. UI: User Interface
2. SQL: Structured Query Language
3. XSS: Cross-Site Scripting
4. IDE: Integrated Development Environment
5. SQLite: Structured Query Language
6. VS Code: Visual Studio Code

## Table of Contents

## INTRODUCTION

This report presents the development of a mobile application designed to streamline the process of writing and managing personal notes. The project was developed using a combination of mobile app development technologies, including Flutter for the app interface and SQLite for database management, with MySQL as the query language, and was undertaken as part of a course in mobile app development.

The need for a note-taking application arose from the limitations of previous manual note-taking methods, which were time-consuming, error-prone, and difficult to manage. The new system provides an efficient and streamlined process for users to write and update their notes, and for registered users to manage and view their content securely.

The app's interface was designed to be intuitive and user-friendly, providing a responsive design that works on a range of devices. The use of Flutter’s rich set of widgets enhanced the app's interactivity, allowing for real-time validation and error messaging. SQLite was used to create a robust and secure local database-driven system that enables efficient storage, retrieval, and management of user notes.

This report outlines the methodology used in the development of the application, including requirements gathering, system design, implementation, testing, deployment, and maintenance. The report also discusses the app's features, functionality, limitations, and future recommendations.

Overall, this mobile app project provides a valuable learning experience and equips students with practical skills in mobile app development and database management.

## PROBLEM STATEMENT

Many users face challenges with the default note-taking applications available on their mobile devices. These apps often lack essential security features, are not optimized for multi-device use, and do not offer an intuitive user experience. This project aims to develop small scale application, a secure and user-friendly mobile application for writing and managing notes, ensuring a seamless and efficient experience for users.

- **Security and Privacy**: The default note apps on mobile devices lack a login system, making it easy for anyone to access sensitive information stored in them. It will include a secure login feature to ensure that only registered users can access their notes, protecting their privacy.

- **Accessibility and Synchronization**: Many existing note-taking methods do not provide real-time synchronization across multiple devices. This application will allow users to access their notes seamlessly from any device, ensuring their information is always up-to-date and readily available.

- **Enhanced User Experience**: Default note apps often lack advanced features and an intuitive interface. It will utilize Flutter to create a responsive and interactive design, offering real-time validation, error messaging, and a user-friendly experience.

## OBJECTIVES

The following objectives were kept in mind while making this project:

1. **Development of Mobile App Using Flutter**: Utilize Flutter for the app's front-end development to create a responsive and intuitive user interface.
2. **Incorporation of Secure Login System**: Implement a login feature to ensure that only registered users can access and manage their notes, enhancing security and privacy.
3. **Efficient Note Management**: Provide users with easy access to write, update, and modify their notes seamlessly.
4. **User-Friendly Interface**: Design an intuitive and graphical user interface that allows both registered users and administrators to navigate and manage their data easily.
5. **Local Data Storage with SQLite**: Use SQLite for robust and secure local database management, ensuring efficient storage, retrieval, and management of user notes.

## METHODOLOGY

This mobile application is a platform where users can register, set their password, and access their notes by entering their credentials on the login page. The data is stored in a secure local database and can be accessed, updated, and deleted by the users themselves without the need to contact the support team. The app is built using Flutter and SQLite, with MySQL as the query language and Android Studio as the development environment.

Flutter is an open-source UI software development toolkit used to create natively compiled applications for mobile, web, and desktop from a single codebase. It allows for a responsive and intuitive user interface. SQLite is a lightweight, serverless, self-contained SQL database engine used for local storage in mobile applications, ensuring efficient and secure data management. MySQL is used as the query language for database interactions. Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

The app is tested and developed locally using the Android Emulator and SQLite database tools, which provide all the necessary components needed to run and test a mobile application efficiently. The use of these technologies ensures a well-designed and functional mobile app that is accessible to both beginners and experienced developers.

**Key Technologies Used:**

- **Flutter**: For creating the app's front-end and ensuring a responsive and interactive user interface.
- **SQLite**: For secure local storage and management of user data.
- **MySQL**: As the query language for efficient database interactions.
- **Android Studio and VS Code**: As the development environment for building and testing the mobile application.

**Features:**

- **User Registration and Login**: Secure login system to protect user data.
- **Note Management**: Users can create, update, and delete notes easily.
- **Graphical User Interface**: Intuitive and user-friendly interface for seamless navigation.
- **Local Data Storage**: Efficient and secure storage of data using SQLite.

## WIREFRAMES

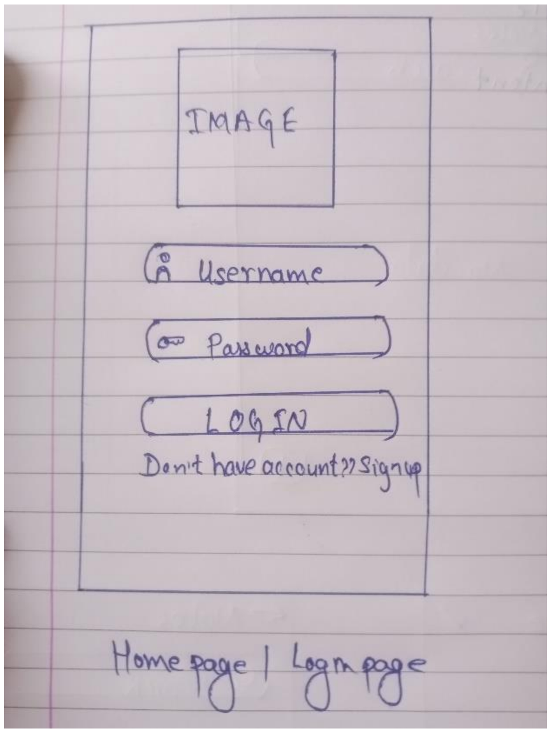
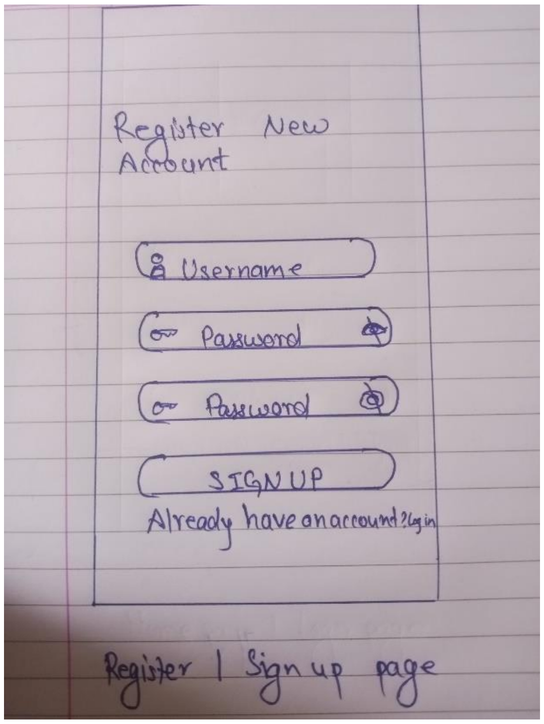
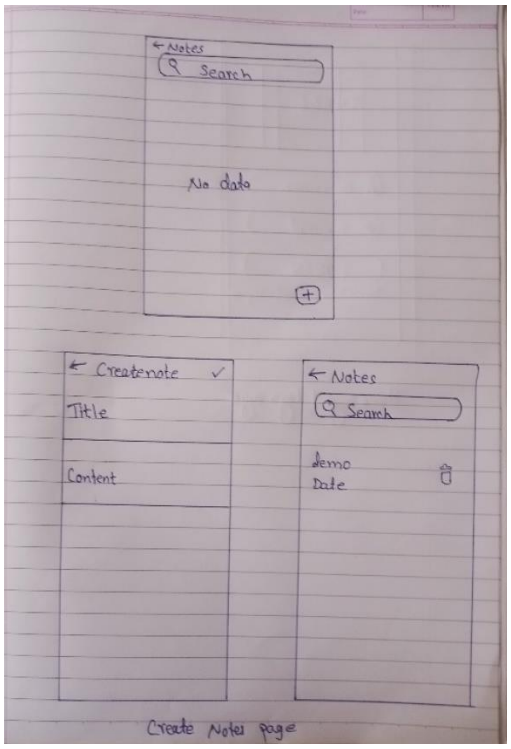


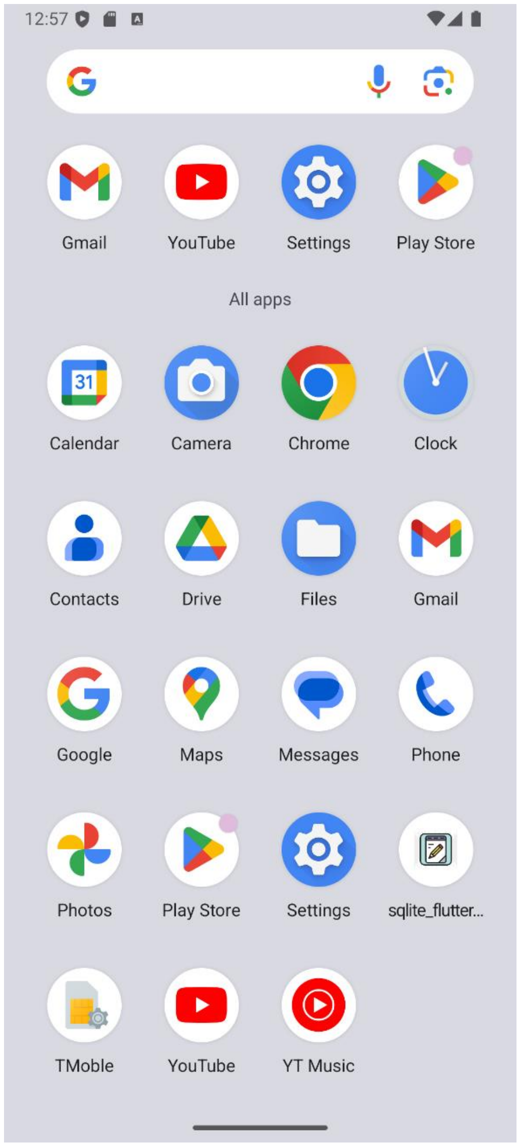
IMAGE
Username
Password
LOGIN
Don't have account?? Signup
Home page | Login page


Register New Account
Username
Password
Password
SIGNUP
Already have an account? Login
Register | Sign up page


← Notes
Search
No data
← Createnote ✓
Title
Content
← Notes
Search
demo
Date
Create Notes page

## IMPLEMENTATION

**App in mobile:**

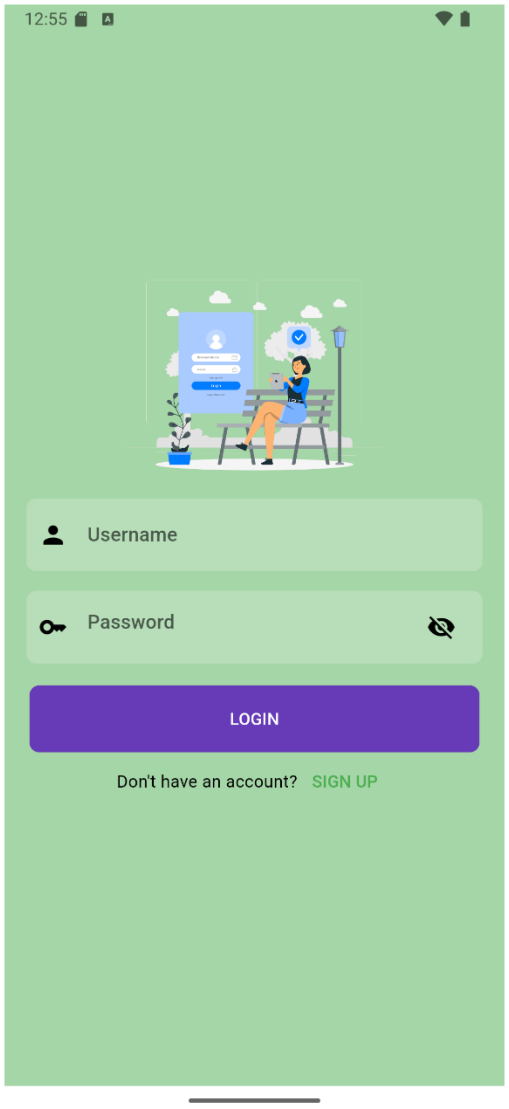


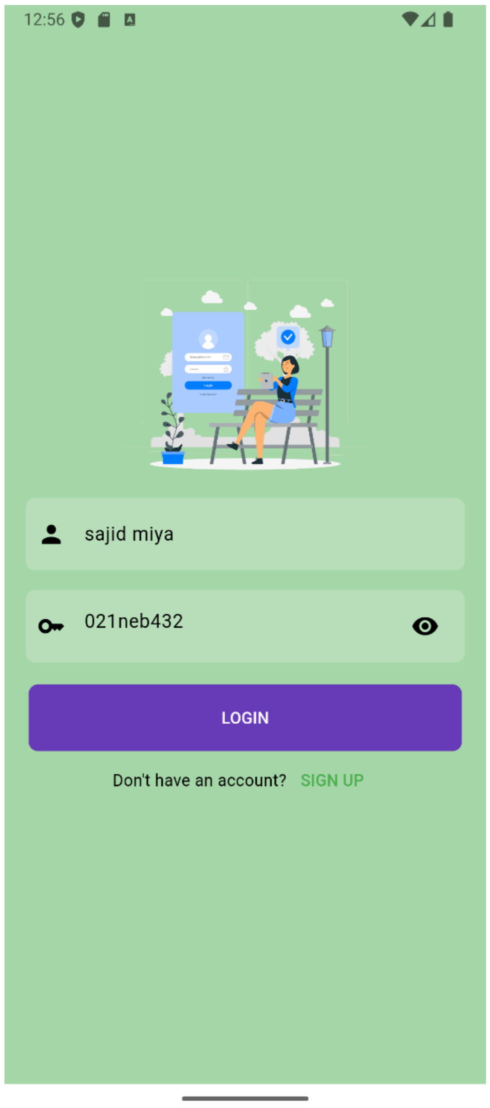
## Codes for pubspec.yaml

```
name: sqlite_flutter_crud
description: A new Flutter project.

publish_to: 'none'
```

```
version: 1.0.0+1

environment:
  sdk: '>=3.1.1 <4.0.0'

dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.2
  sqflite: ^2.3.0
  path: ^1.8.3
  intl: ^0.18.1
  path_provider: ^2.1.1

dev_dependencies:
  flutter_test:
    sdk: flutter


  flutter_lints: ^2.0.0

flutter:

  uses-material-design: true

  assets:
   - lib/assets/
```

## Index or login page:





**Codes:**

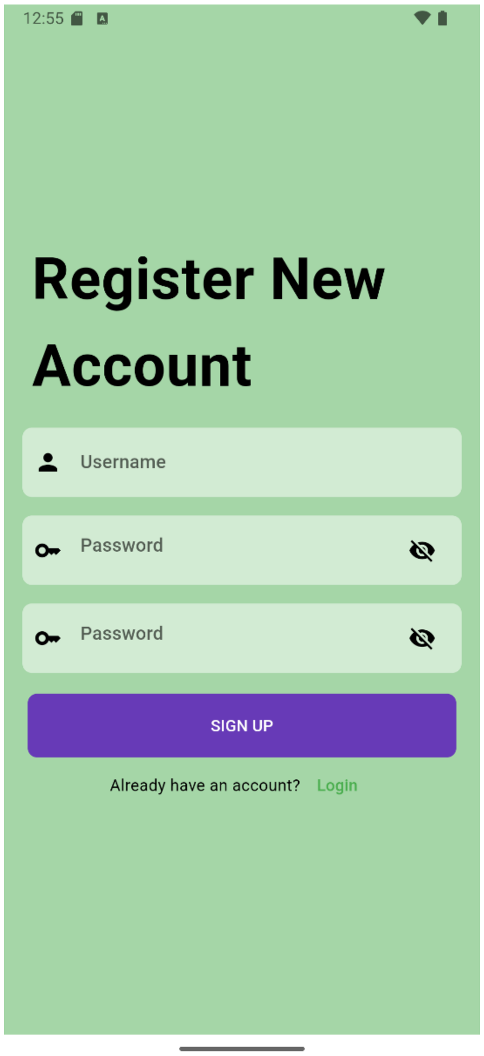
```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:sqlite_flutter_crud/Authtentication/login.dart';

void main() {
  SystemChrome.setSystemUIOverlayStyle(const SystemUiOverlayStyle(
    statusBarColor: Colors.transparent,
    statusBarBrightness: Brightness.dark,
    systemNavigationBarColor: Colors.transparent,
    systemNavigationBarDividerColor: Colors.transparent,
    systemNavigationBarIconBrightness: Brightness.dark,
    statusBarIconBrightness: Brightness.dark,
  ));
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'CRUD',
      theme: ThemeData(
        primarySwatch: Colors.green,
        colorScheme: ColorScheme.fromSwatch(
            primarySwatch: Colors.green),
        useMaterial3: true,
      ),
      home: const LoginScreen(),
    );
  }
}
```

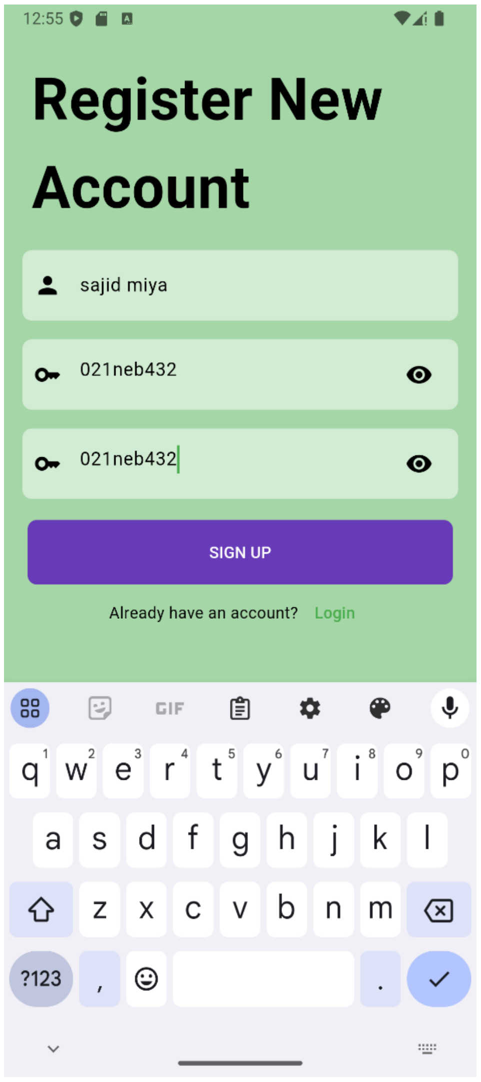
Login code:

```
import 'package:flutter/material.dart';
import 'package:sqlite_flutter_crud/Authtentication/signup.dart';
import 'package:sqlite_flutter_crud/JsonModels/users.dart';
import 'package:sqlite_flutter_crud/SQLite/sqlite.dart';
import 'package:sqlite_flutter_crud/Views/notes.dart';

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

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final username = TextEditingController();
  final password = TextEditingController();
```

```
bool isVisible = false;
bool isLoginTrue = false;

final db = DatabaseHelper();
login() async {
  var response = await db
      .login(Users(usrName: username.text, usrPassword: password.text));
  if (response == true) {
    if (!mounted) return;
    Navigator.pushReplacement(
        context, MaterialPageRoute(builder: (context) => const Notes()));
  } else {
    setState(() {
      isLoginTrue = true;
    });
  }
}

final formKey = GlobalKey<FormState>();
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(10.0),
          child: Form(
            key: formKey,
            child: Column(
              children: [
                Image.asset(
                  "lib/assets/login.png",
                  width: 210,
                ),
                const SizedBox(height: 15),
                Container(
                  margin: const EdgeInsets.all(8),
                  padding:
                      const EdgeInsets.symmetric(horizontal: 10, vertical: 6),
                  decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(8),
                      color: Colors.white.withOpacity(.2)),
                  child: TextFormField(
                    controller: username,
    validator: (value) {
      if (value!.isEmpty) {
        return "required";
      }
      return null;
    },
    decoration: const InputDecoration(
      icon: Icon(Icons.person),
      border: InputBorder.none,
      hintText: "Username",
    ),
  ),
),
Container(
  margin: const EdgeInsets.all(8),
  padding:
      const EdgeInsets.symmetric(horizontal: 10, vertical: 6),
  decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(8),
      color: Colors.white.withOpacity(.2)),
  child: TextFormField(
    controller: password,
    validator: (value) {
      if (value!.isEmpty) {
        return "required";
      }
      return null;
    },
    obscureText: !isVisible,
    decoration: InputDecoration(
      icon: const Icon(Icons.key),
      border: InputBorder.none,
      hintText: "Password",
      suffixIcon: IconButton(
        onPressed: () {
          setState(() {
            isVisible = !isVisible;
          });
        },
        icon: Icon(isVisible
            ? Icons.visibility
            : Icons.visibility_off),
      ),
    ),
          ),
        ),
        const SizedBox(height: 10),
        Container(
          height: 55,
          width: MediaQuery.of(context).size.width * .9,
          decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(8),
              color: Colors.deepPurple),
          child: TextButton(
              onPressed: () {
                if (formKey.currentState!.validate()) {
                  login();
                }
              },
              child: const Text(
                "LOGIN",
                style: TextStyle(color: Colors.white),
              )),
        ),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text("Don't have an account?"),
            TextButton(
                onPressed: () {
                  Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) => const SignUp()));
                },
                child: const Text("SIGN UP"))
          ],
        ),
        isLoginTrue
            ? const Text(
                "invalid credentials",
                style: TextStyle(color: Colors.red),
              )
            : const SizedBox(),
      ],
    ),
  ),
),
        ),
      ),
    );
  }}
```

## Register/ sign up page:



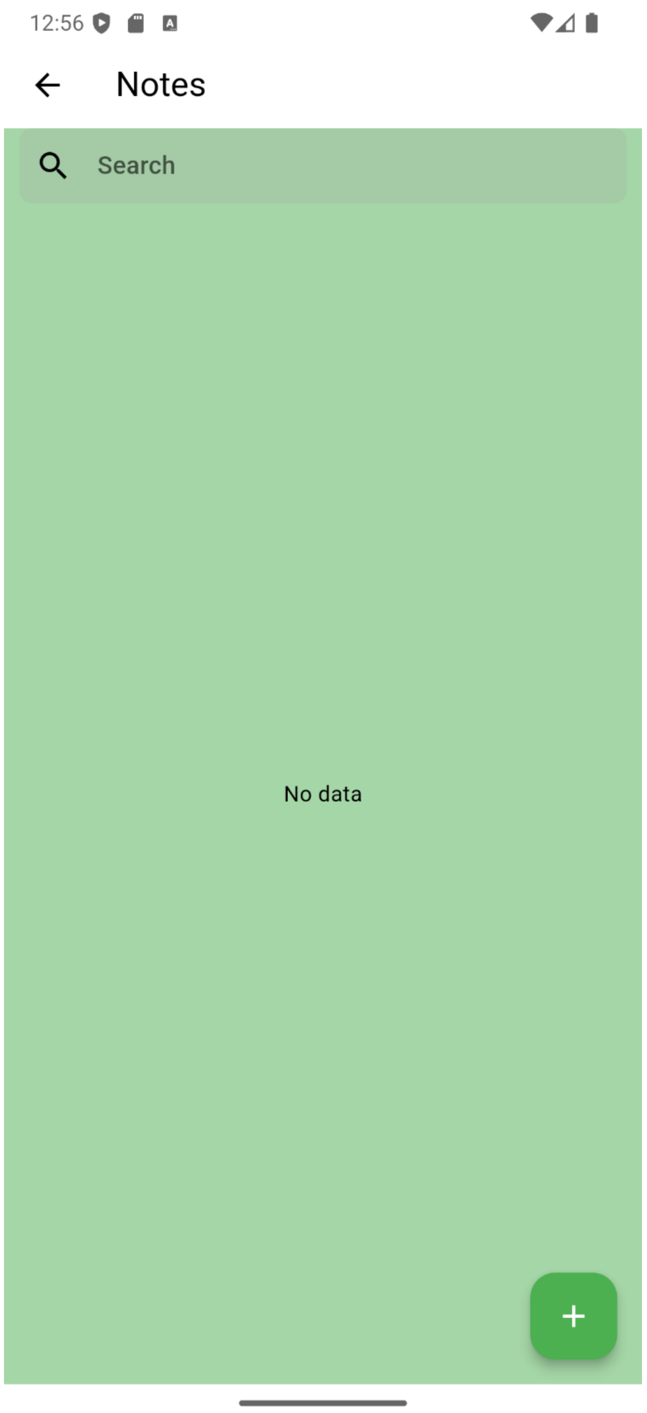


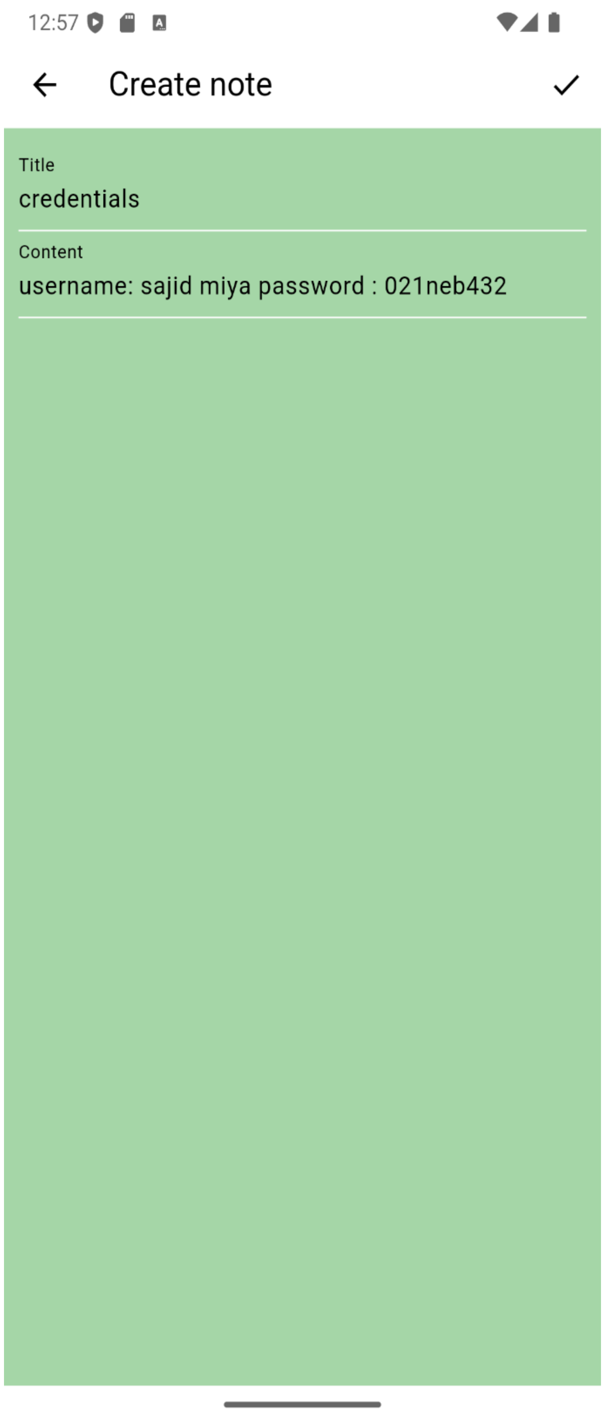
**Codes:**

```
import 'package:flutter/material.dart';
import 'package:sqlite_flutter_crud/Authtentication/login.dart';
import 'package:sqlite_flutter_crud/JsonModels/users.dart';
import 'package:sqlite_flutter_crud/SQLite/sqlite.dart';

class SignUp extends StatefulWidget {
  const SignUp({super.key});
  @override
  State<SignUp> createState() => _SignUpState();
```

```
}

class _SignUpState extends State<SignUp> {
  final username = TextEditingController();
  final password = TextEditingController();
  final confirmPassword = TextEditingController();
  final formKey = GlobalKey<FormState>();
  bool isVisible = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: SingleChildScrollView(
          child: Form(
            key: formKey,
            child: Padding(
              padding: const EdgeInsets.all(8.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  const ListTile(
                    title: Text(
                      "Register New Account",
                      style:
                          TextStyle(fontSize: 50, fontWeight: FontWeight.bold),
                    ),
                  ),
                  Container(
                    margin: EdgeInsets.all(8),
                    padding:
                        const EdgeInsets.symmetric(horizontal: 10, vertical: 6),
                    decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(8),
                        color: Colors.white.withOpacity(.5)),
                    child: TextFormField(
                      controller: username,
                      validator: (value) {
                        if (value!.isEmpty) {
                          return "required";
                        }
                        return null;
                      },
                      decoration: const InputDecoration(
                        icon: Icon(Icons.person),
                        border: InputBorder.none,
                        hintText: "Username",
                      ),
                    ),
                  ),
                  Container(
                    margin: const EdgeInsets.all(8),
                    padding:
                        const EdgeInsets.symmetric(horizontal: 10, vertical: 6),
                    decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(8),
                        color: Colors.white.withOpacity(.5)),
                    child: TextFormField(
                      controller: password,
                      validator: (value) {
                        if (value!.isEmpty) {
                          return "required";
                        }
                        return null;
                      },
                      obscureText: !isVisible,
                      decoration: InputDecoration(
                        icon: const Icon(Icons.key),
                        border: InputBorder.none,
                        hintText: "Password",
                        suffixIcon: IconButton(
                          onPressed: () {
                            setState(() {
                              isVisible = !isVisible;
                            });
                          },
                          icon: Icon(isVisible
                              ? Icons.visibility
                              : Icons.visibility_off),
                        ),
                      ),
                    ),
                  ),
                  Container(
                    margin: const EdgeInsets.all(8),
                    padding:
                        const EdgeInsets.symmetric(horizontal: 10, vertical: 6),
                    decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(8),
                          color: Colors.white.withOpacity(.5)),
                      child: TextFormField(
                        controller: confirmPassword,
                        validator: (value) {
                          if (value!.isEmpty) {
                            return "required";
                          } else if (password.text != confirmPassword.text) {
                            return "Passwords don't match";
                          }
                          return null;
                        },
                        obscureText: !isVisible,
                        decoration: InputDecoration(
                            icon: const Icon(Icons.key),
                            border: InputBorder.none,
                            hintText: "Password",
                            suffixIcon: IconButton(
                                onPressed: () {
                                  setState(() {
                                    isVisible = !isVisible;
                                  });
                                },
                                icon: Icon(isVisible
                                    ? Icons.visibility
                                    : Icons.visibility_off))),
                      ),
                    ),
                    const SizedBox(height: 10),
                    Container(
                      height: 55,
                      width: MediaQuery.of(context).size.width * .9,
                      decoration: BoxDecoration(
                          borderRadius: BorderRadius.circular(8),
                          color: Colors.deepPurple),
                      child: TextButton(
                        onPressed: () {
                          if (formKey.currentState!.validate()) {
                            final db = DatabaseHelper();
                            db
                                .signup(Users(
                                    usrName: username.text,
                                    usrPassword: password.text))
                                .whenComplete(() {
                              Navigator.push(
                                      context,
                                      MaterialPageRoute(
                                          builder: (context) => const LoginScreen()));
                                });
                              }
                            },
                            child: const Text(
                              "SIGN UP",
                              style: TextStyle(color: Colors.white),
                            ),
                          ),
                        ),
                        Row(
                          mainAxisAlignment: MainAxisAlignment.center,
                          children: [
                            const Text("Already have an account?"),
                            TextButton(
                                onPressed: () {
                                  Navigator.push(
                                      context,
                                      MaterialPageRoute(
                                          builder: (context) => const LoginScreen()));
                                },
                                child: const Text("Login"))
                          ],
                        )
                      ],
                     ),
                    ),
                   ),
                   ),
                   ),
                   );
                   }
                  }
```

## Create notes page:





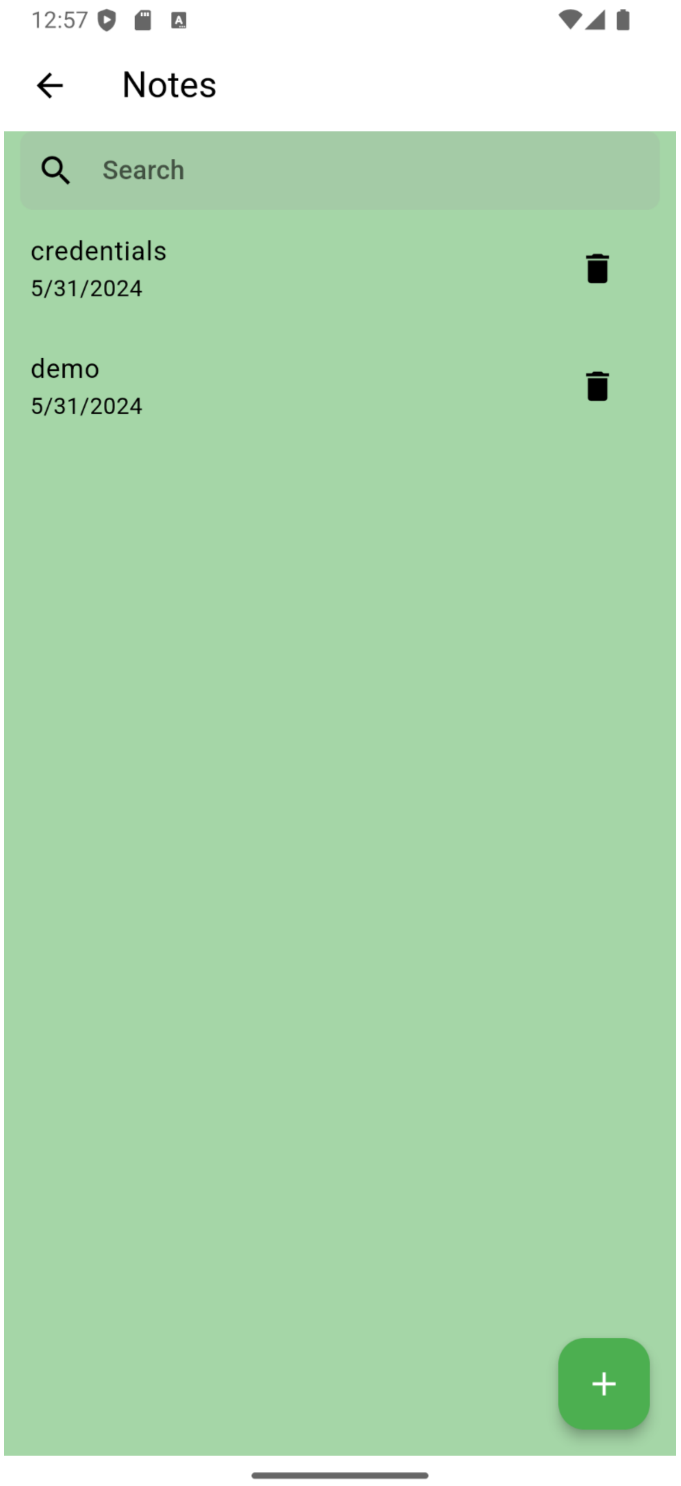
credentials

username: sajid miya password : 021neb432

**Codes:**
import 'package:flutter/material.dart';
import 'package:sqlite_flutter_crud/JsonModels/note_model.dart';
import 'package:sqlite_flutter_crud/SQLite/sqlite.dart';

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

  @override
  State<CreateNote> createState() => _CreateNoteState();
}

class _CreateNoteState extends State<CreateNote> {
  final title = TextEditingController();
  final content = TextEditingController();
  final formKey = GlobalKey<FormState>();

  final db = DatabaseHelper();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Create note"),
        actions: [
          IconButton(
              onPressed: () {
                if (formKey.currentState!.validate()) {
                  db
                      .createNote(NoteModel(
                          noteTitle: title.text,
                          noteContent: content.text,
                          createdAt: DateTime.now().toIso8601String()))
                      .whenComplete(() {
                    //When this value is true
                    Navigator.of(context).pop(true);
                  });
                }
              },
              icon: Icon(Icons.check))
        ],
      ),
      body: Form(
          key: formKey,
          child: Padding(
            padding: const EdgeInsets.all(10.0),
            child: Column(
              children: [
                TextFormField(
                  controller: title,
                  validator: (value) {
                    if (value!.isEmpty) {
                      return "Title is required";
                    }
                    return null;
                  },
                  decoration: const InputDecoration(
                    label: Text("Title"),
                  ),
                ),
                TextFormField(
                  controller: content,
                  validator: (value) {
                    if (value!.isEmpty) {
                      return "Content is required";
                    }
                    return null;
                  },
                  decoration: const InputDecoration(
                    label: Text("Content"),
                  ),
                ),
              ],
            ),
          )),
    );
  }
}
```

## Created notes:



credentials
5/31/2024
demo
5/31/2024

**Codes :**

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:sqlite_flutter_crud/JsonModels/note_model.dart';
import 'package:sqlite_flutter_crud/SQLite/sqlite.dart';
import 'package:sqlite_flutter_crud/Views/create_note.dart';

class Notes extends StatefulWidget {
  const Notes({super.key});
  @override
  State<Notes> createState() => _NotesState();
}

class _NotesState extends State<Notes> {
  late DatabaseHelper handler;
  late Future<List<NoteModel>> notes;
  final db = DatabaseHelper();
  final title = TextEditingController();
  final content = TextEditingController();
  final keyword = TextEditingController();
  @override
  void initState() {
    handler = DatabaseHelper();
    notes = handler.getNotes();
    handler.initDB().whenComplete(() {
      notes = getAllNotes();
    });
    super.initState();
  }

  Future<List<NoteModel>> getAllNotes() {
    return handler.getNotes();
  }

  Future<List<NoteModel>> searchNote() {
    return handler.searchNotes(keyword.text);
  }

  Future<void> _refresh() async {
    setState(() {
      notes = getAllNotes();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
  title: const Text("Notes"),
),
floatingActionButton: FloatingActionButton(
  onPressed: () {
    Navigator.push(context,
            MaterialPageRoute(builder: (context) => const CreateNote()))
        .then((value) {
      if (value) {
        _refresh();
      }
    });
  },
  child: const Icon(Icons.add),
),
body: Column(
  children: [
    Container(
      padding: const EdgeInsets.symmetric(horizontal: 10),
      margin: const EdgeInsets.symmetric(horizontal: 10),
      decoration: BoxDecoration(
          color: Colors.grey.withOpacity(.2),
          borderRadius: BorderRadius.circular(8)),
      child: TextFormField(
        controller: keyword,
        onChanged: (value) {
          if (value.isNotEmpty) {
            setState(() {
              notes = searchNote();
            });
          } else {
            setState(() {
              notes = getAllNotes();
            });
          }
        },
        decoration: const InputDecoration(
            border: InputBorder.none,
            icon: Icon(Icons.search),
            hintText: "Search"),
      ),
    ),
    Expanded(
      child: FutureBuilder<List<NoteModel>>(
        future: notes,
builder: (BuildContext context,
    AsyncSnapshot<List<NoteModel>> snapshot) {
  if (snapshot.connectionState == ConnectionState.waiting) {
    return const CircularProgressIndicator();
  } else if (snapshot.hasData && snapshot.data!.isEmpty) {
    return const Center(child: Text("No data"));
  } else if (snapshot.hasError) {
    return Text(snapshot.error.toString());
  } else {
    final items = snapshot.data ?? <NoteModel>[];
    return ListView.builder(
        itemCount: items.length,
        itemBuilder: (context, index) {
          return ListTile(
            subtitle: Text(DateFormat("yMd").format(
                DateTime.parse(items[index].createdAt))),
            title: Text(items[index].noteTitle),
            trailing: IconButton(
              icon: const Icon(Icons.delete),
              onPressed: () {
                db
                    .deleteNote(items[index].noteId!)
                    .whenComplete(() {
                  _refresh();
                });
              },
            ),
            onTap: () {
              setState(() {
                title.text = items[index].noteTitle;
                content.text = items[index].noteContent;
              });
              showDialog(
                  context: context,
                  builder: (context) {
                    return AlertDialog(
                      actions: [
                        Row(
                          children: [
                            TextButton(
                              onPressed: () {
                                db
                                    .updateNote(
                                        title.text,
                  content.text,
                  items[index].noteId)
                .whenComplete(() {
              _refresh();
              Navigator.pop(context);
            });
          },
          child: const Text("Update"),
        ),
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text("Cancel"),
        ),
      ],
    ),
  ],
  title: const Text("Update note"),
  content: Column(
    mainAxisSize: MainAxisSize.min,
    children: [
      TextFormField(
        controller: title,
        validator: (value) {
          if (value!.isEmpty) {
            return "Title is required";
          }
          return null;
        },
        decoration: const InputDecoration(
          label: Text("Title"),
        ),
      ),
      TextFormField(
        controller: content,
        validator: (value) {
          if (value!.isEmpty) {
            return "Content is required";
          }
          return null;
        },
        decoration: const InputDecoration(
          label: Text("Content"),
                                        ),
                                      ),
                                    ]),
                                );
                              });
                            },
                          );
                        });
                      }
                    },
                  ),
                ),
              ],
            ));
  }
}
```

## Backend / database code:

```
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'package:sqlite_flutter_crud/JsonModels/note_model.dart';
import 'package:sqlite_flutter_crud/JsonModels/users.dart';

class DatabaseHelper {
  final databaseName = "notes.db";
  String noteTable =
      "CREATE TABLE notes (noteId INTEGER PRIMARY KEY AUTOINCREMENT, noteTitle
TEXT NOT NULL, noteContent TEXT NOT NULL, createdAt TEXT DEFAULT
CURRENT_TIMESTAMP)";


  String users =
      "create table users (usrId INTEGER PRIMARY KEY AUTOINCREMENT, usrName TEXT
UNIQUE, usrPassword TEXT)";

  Future<Database> initDB() async {
    final databasePath = await getDatabasesPath();
    final path = join(databasePath, databaseName);

    return openDatabase(path, version: 1, onCreate: (db, version) async {
      await db.execute(users);
      await db.execute(noteTable);
    });
  }


  Future<bool> login(Users user) async {
    final Database db = await initDB();

    var result = await db.rawQuery(
        "select * from users where usrName = '${user.usrName}' AND usrPassword =
'${user.usrPassword}'");
    if (result.isNotEmpty) {
      return true;
    } else {
      return false;
    }
  }

  //Sign up
  Future<int> signup(Users user) async {
    final Database db = await initDB();

    return db.insert('users', user.toMap());
  }

  //Search Method
  Future<List<NoteModel>> searchNotes(String keyword) async {
    final Database db = await initDB();
    List<Map<String, Object?>> searchResult = await db
        .rawQuery("select * from notes where noteTitle LIKE ?", ["%$keyword%"]);
    return searchResult.map((e) => NoteModel.fromMap(e)).toList();
  }

  Future<int> createNote(NoteModel note) async {
    final Database db = await initDB();
    return db.insert('notes', note.toMap());
  }


  Future<List<NoteModel>> getNotes() async {
    final Database db = await initDB();
    List<Map<String, Object?>> result = await db.query('notes');
    return result.map((e) => NoteModel.fromMap(e)).toList();
  }
  Future<int> deleteNote(int id) async {
    final Database db = await initDB();
    return db.delete('notes', where: 'noteId = ?', whereArgs: [id]);
  }

  Future<int> updateNote(title, content, noteId) async {
    final Database db = await initDB();
    return db.rawUpdate(
        'update notes set noteTitle = ?, noteContent = ? where noteId = ?',
        [title, content, noteId]);
  }
}
```

## How to Open, Run, and Build This Project in Visual Studio Code

**Step 1: Open Visual Studio Code**

1. **Launch VS Code:**
    - Open Visual Studio Code on your computer.

**Step 2: Open Your Flutter Project**

1. **Browse to the Project Location:**
    - Press **Ctrl + K + O**.
    - Navigate to the location where your Flutter project is downloaded.
    - Select the project folder to open it in VS Code.

**Step 3: Set Up a Device or Emulator**

1. **Physical Device:**
    - For an Android device, ensure Developer Mode and USB Debugging are enabled.
    - For an iOS device, ensure you have Xcode installed and the device is set up.
2. **Emulator:**
    - For Android, open Android Studio, set up the Android Emulator, and create a virtual device.
    - For iOS, use the Simulator that comes with Xcode.

**Step 4: Run Your Flutter App**

1. **Select Device:**
    - Ensure a device or emulator is connected and selected in the bottom status bar of VS Code.
2. **Run the App:**
    - Open the main Flutter application file (usually **lib/main.dart**).
    - Start debugging by pressing **F5** or by going to **Run > Start Debugging**.
    - Alternatively, open the terminal in VS Code and run: `flutter run`

**Step 5: Verify the Installation**

1. **Open a Terminal in VS Code:**
    - Go to **View > Terminal** or press **Ctrl +** `(backtick).
2. **Run flutter doctor:**
    - In the terminal, run the following command to check for any issues:`flutter doctor`
    - Follow any instructions provided to resolve issues.

**Step 6: Build an APK**

1. **Open the Terminal:**
    - Open the integrated terminal in VS Code by pressing **Ctrl +** `(backtick).
2. **Build the APK:**
    - In the terminal, run: ` flutter build apk`
3. **Locate the APK File:**
    - Once the build process is complete, find the generated APK file in the **build/app/outputs/flutter-apk/** directory within your project folder. The file will typically be named **app-release.apk**.

## Conclusions

In conclusion, the development of this mobile application using Flutter, SQLite, and MySQL has been a success. The app provides a user-friendly interface for users to register, log in, and manage their notes securely. The use of Flutter has enhanced the app's interactivity, allowing for a responsive and intuitive user experience. SQLite has been used to create a robust and secure local database system that enables efficient storage, retrieval, and management of user data. MySQL serves as the query language, facilitating effective database interactions. The project has demonstrated the effective use of mobile development technologies in creating a dynamic and functional note-taking application. Future work could involve improving the app's scalability and further enhancing its security features. Overall, the mobile app project has provided a valuable learning experience and has equipped us with practical skills in mobile app development and database management.

## Limitations and Future Recommendations

As this is just a sample for the mobile application, it has various limitations like the absence of password recovery, data validations, data security, and others. Here are some potential limitations of a note-taking mobile app using Flutter, SQLite, and MySQL:

- **Security Vulnerabilities**: Although SQLite and MySQL are capable of providing robust security features, any mobile application is inherently vulnerable to attacks such as SQL injection, cross-site scripting (XSS), and other exploits. Ensuring that the app is secure and remains up-to-date with the latest security patches and protocols is essential.

- **Compatibility Issues**: Different mobile devices and operating system versions may render the app's UI differently, which could affect the app's usability and functionality. Ensuring that the app is compatible across multiple platforms and devices is important to ensure a consistent user experience.

- **User Input Errors**: While Flutter can provide real-time validation and error messaging, user input errors could still occur. These errors may be due to factors such as incomplete or incorrect information provided by users, leading to data integrity issues.

- **Scalability**: Depending on the number of users, the app may need to support a large number of concurrent users and handle a high volume of data. Ensuring that the app is scalable and optimized to handle the workload is crucial to ensure the app remains performant and responsive

- **Limited User Privacy**: Currently, only registered users can log in, but there is a limitation where registered users can see other users' notes. This poses a privacy concern, as users expect their notes to be accessible only to themselves.

- **Maintenance**: Like any software system, regular maintenance and updates are required to ensure that the app remains functional and up-to-date. This may involve fixing bugs, addressing user feedback, or adding new features to the app, which could be time-consuming and resource-intensive.

## Recommendations for Future Developments:

- **Improved User Experience**: To improve the user experience, the app should be designed to be more intuitive and user-friendly. This can include features such as automatic form filling, autocomplete, and progress indicators to guide users through the note-taking process.

- **Integration with External Systems**: The note-taking app can be integrated with other systems, such as cloud storage services and productivity tools, to streamline the flow of information and provide a more unified experience for users.

- **Enhanced Security**: As security threats continue to evolve, the app should be regularly updated to ensure the latest security patches are applied, and that appropriate measures are taken to protect user data.

- **Analytics and Reporting**: To improve decision-making and track performance, the app can be integrated with analytics and reporting tools to provide insights into user behavior and system performance. This can help administrators identify areas for improvement and optimize the app accordingly.

## REFERENCES

https://docs.flutter.dev/

https://youtube.com/playlist?list=PLjVLYmrlmjGfGLShoW0vVX_tcyT8u1Y3E&si=r7609dE3-oeYC0hE

https://youtube.com/playlist?list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ&si=RamURXL0RCQw1fZP

https://youtu.be/oZujEA99kXA?si=QG1-w6-Dc_cEwSPS