https://github.com/BaqirSultan201/MAD-THEORY

Database\_helper  
  
import 'dart:async';

import 'dart:io';

import 'package:path/path.dart';

import 'package:sqflite/sqflite.dart';

import '../module/user.dart'; // Fixed import path

import '../module/location\_note.dart'; // Fixed import path

import 'package:path\_provider/path\_provider.dart';

class DatabaseHelper {

  static final DatabaseHelper \_instance = DatabaseHelper.\_internal();

  factory DatabaseHelper() => \_instance;

  DatabaseHelper.\_internal();

  static Database? \_database;

  Future<Database> get database async {

    if (\_database != null) return \_database!;

    \_database = await \_initDatabase();

    return \_database!;

  }

  Future<Database> \_initDatabase() async {

    Directory documentsDirectory = await getApplicationDocumentsDirectory();

    String path = join(documentsDirectory.path, 'location\_reminder.db');

    return await openDatabase(

      path,

      version: 1,

      onCreate: \_onCreate,

    );

  }

  Future \_onCreate(Database db, int version) async {

    await db.execute('''

      CREATE TABLE users (

        username TEXT PRIMARY KEY,

        address TEXT,

        password TEXT

      )

    ''');

    await db.execute('''

      CREATE TABLE location\_notes (

        id INTEGER PRIMARY KEY AUTOINCREMENT,

        name TEXT,

        description TEXT,

        imagePath TEXT

      )

    ''');

  }

  // User operations

  Future<int> insertUser(User user) async {

    Database db = await database;

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

  }

  Future<User?> getUser(String username) async {

    Database db = await database;

    List<Map<String, dynamic>> maps = await db.query(

      'users',

      where: 'username = ?',

      whereArgs: [username],

    );

    if (maps.isNotEmpty) {

      return User.fromMap(maps.first);

    }

    return null;

  }

  // LocationNote operations

  Future<int> insertLocationNote(LocationNote note) async {

    Database db = await database;

    return await db.insert('location\_notes', note.toMap());

  }

  Future<List<LocationNote>> getLocationNotes() async {

    Database db = await database;

    List<Map<String, dynamic>> maps = await db.query('location\_notes');

    return maps.map((map) => LocationNote.fromMap(map)).toList();

  }

  Future<int> updateLocationNote(LocationNote note) async {

    Database db = await database;

    return await db.update(

      'location\_notes',

      note.toMap(),

      where: 'id = ?',

      whereArgs: [note.id],

    );

  }

  Future<int> deleteLocationNote(int id) async {

    Database db = await database;

    return await db.delete(

      'location\_notes',

      where: 'id = ?',

      whereArgs: [id],

    );

  }

}

Location\_note

class LocationNote {

  final int? id;

  final String name;

  final String description;

  final String? imagePath;

  LocationNote({

    this.id,

    required this.name,

    required this.description,

    this.imagePath,

  });

  Map<String, dynamic> toMap() {

    return {

      'id': id,

      'name': name,

      'description': description,

      'imagePath': imagePath,

    };

  }

  factory LocationNote.fromMap(Map<String, dynamic> map) {

    return LocationNote(

      id: map['id'],

      name: map['name'],

      description: map['description'],

      imagePath: map['imagePath'],

    );

  }

}

User

class User {

  final String username;

  final String address;

  final String password;

  User({required this.username, required this.address, required this.password});

  Map<String, dynamic> toMap() {

    return {

      'username': username,

      'address': address,

      'password': password,

    };

  }

  factory User.fromMap(Map<String, dynamic> map) {

    return User(

      username: map['username'],

      address: map['address'],

      password: map['password'],

    );

  }

}

**# Code Citations**

**## License: unknown**

https://github.com/MarkoZoreski/MIS\_KolokviumApp/tree/934a6256568af44950aa2b9ee814055ab6856a8c/lab3/lib/screens/login\_screen.dart

```

;

      } else {

        ScaffoldMessenger.of(context).showSnackBar(

          SnackBar(content: Text('Invalid username or password')),

        );

      }

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Login

```

**## License: unknown**

https://github.com/Ahmed-2020-98/login\_screen\_flutter/tree/d54a268e01d20ee016e8259c45369695f26d2518/lib/login.dart

```

}

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Login')),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Form(

          key: \_formKey,

          child: Column

```

import 'package:flutter/material.dart';

import '../db/database\_helper.dart';

import '../module/location\_note.dart';

class LocationNotesScreen extends StatefulWidget {

  const LocationNotesScreen({super.key});

  @override

  \_LocationNotesScreenState createState() => \_LocationNotesScreenState();

}

class \_LocationNotesScreenState extends State<LocationNotesScreen> {

  List<LocationNote> \_notes = [];

  final DatabaseHelper \_dbHelper = DatabaseHelper();

  @override

  void initState() {

    super.initState();

    \_loadNotes();

  }

  void \_loadNotes() async {

    List<LocationNote> notes = await \_dbHelper.getLocationNotes();

    setState(() {

      \_notes = notes;

    });

  }

  void \_showNoteDialog({LocationNote? note}) {

    final nameController = TextEditingController(text: note?.name);

    final descController = TextEditingController(text: note?.description);

    showDialog(

      context: context,

      builder: (\_) => AlertDialog(

        title: Text(note == null ? 'Add Note' : 'Edit Note'),

        content: Column(

          mainAxisSize: MainAxisSize.min,

          children: [

            TextField(

              controller: nameController,

              decoration: InputDecoration(labelText: 'Name'),

            ),

            TextField(

              controller: descController,

              decoration: InputDecoration(labelText: 'Description'),

            ),

          ],

        ),

        actions: [

          ElevatedButton(

            onPressed: () async {

              final newNote = LocationNote(

                id: note?.id,

                name: nameController.text,

                description: descController.text,

              );

              if (note == null) {

                await \_dbHelper.insertLocationNote(newNote);

              } else {

                await \_dbHelper.updateLocationNote(newNote);

              }

              Navigator.pop(context);

              \_loadNotes();

            },

            child: Text('Save'),

          )

        ],

      ),

    );

  }

  void \_deleteNote(int id) async {

    await \_dbHelper.deleteLocationNote(id);

    \_loadNotes();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Location Notes')),

      body: ListView.builder(

        itemCount: \_notes.length,

        itemBuilder: (\_, index) {

          final note = \_notes[index];

          return ListTile(

            title: Text(note.name),

            subtitle: Text(note.description),

            trailing: Row(

              mainAxisSize: MainAxisSize.min,

              children: [

                IconButton(

                  icon: Icon(Icons.edit),

                  onPressed: () => \_showNoteDialog(note: note),

                ),

                IconButton(

                  icon: Icon(Icons.delete),

                  onPressed: () => \_deleteNote(note.id!),

                ),

              ],

            ),

          );

        },

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_showNoteDialog(),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import '../db/database\_helper.dart';

import '../module/user.dart'; // Fixed import path

import 'location\_notes.dart';

class Login extends StatefulWidget {

  const Login({super.key});

  @override

  \_LoginState createState() => \_LoginState();

}

class \_LoginState extends State<Login> {

  final \_formKey = GlobalKey<FormState>();

  final \_usernameController = TextEditingController();

  final \_passwordController = TextEditingController();

  final DatabaseHelper \_dbHelper = DatabaseHelper();

  @override

  void dispose() {

    \_usernameController.dispose();

    \_passwordController.dispose();

    super.dispose();

  }

  void \_loginUser() async {

    if (\_formKey.currentState!.validate()) {

      User? user = await \_dbHelper.getUser(\_usernameController.text.trim());

      if (user != null && user.password == \_passwordController.text.trim()) {

        Navigator.pushReplacement(

          context,

          MaterialPageRoute(builder: (\_) => LocationNotesScreen()),

        );

      } else {

        ScaffoldMessenger.of(context).showSnackBar(

          SnackBar(content: Text('Invalid username or password')),

        );

      }

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Login')),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Form(

          key: \_formKey,

          child: Column(

            children: [

              TextFormField(

                controller: \_usernameController,

                decoration: InputDecoration(labelText: 'Username'),

                validator: (value) => value!.isEmpty ? 'Enter username' : null,

              ),

              TextFormField(

                controller: \_passwordController,

                decoration: InputDecoration(labelText: 'Password'),

                obscureText: true,

                validator: (value) => value!.isEmpty ? 'Enter password' : null,

              ),

              SizedBox(height: 20),

              ElevatedButton(

                onPressed: \_loginUser,

                child: Text('Login'),

              )

            ],

          ),

        ),

      ),

    );

  }

}

import 'package:flutter/material.dart';

import '../db/database\_helper.dart';

import '../module/user.dart'; // Fixed import path

import 'location\_notes.dart';

import 'login.dart';

class Signup extends StatefulWidget {

  const Signup({super.key});

  @override

  \_SignupState createState() => \_SignupState();

}

class \_SignupState extends State<Signup> {

  final \_formKey = GlobalKey<FormState>();

  final \_usernameController = TextEditingController();

  final \_addressController = TextEditingController();

  final \_passwordController = TextEditingController();

  final \_confirmPasswordController = TextEditingController();

  final DatabaseHelper \_dbHelper = DatabaseHelper();

  @override

  void dispose() {

    \_usernameController.dispose();

    \_addressController.dispose();

    \_passwordController.dispose();

    \_confirmPasswordController.dispose();

    super.dispose();

  }

  void \_registerUser() async {

    if (\_formKey.currentState!.validate()) {

      User? existingUser =

          await \_dbHelper.getUser(\_usernameController.text.trim());

      if (existingUser != null) {

        ScaffoldMessenger.of(context).showSnackBar(

          SnackBar(content: Text('Username already exists')),

        );

        return;

      }

      User newUser = User(

        username: \_usernameController.text.trim(),

        address: \_addressController.text.trim(),

        password: \_passwordController.text.trim(),

      );

      await \_dbHelper.insertUser(newUser);

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text('Account created successfully!')),

      );

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(builder: (context) => LocationNotesScreen()),

      );

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Sign Up')),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Form(

          key: \_formKey,

          child: Column(

            children: [

              TextFormField(

                controller: \_usernameController,

                decoration: InputDecoration(labelText: 'Username'),

                validator: (value) => value!.isEmpty ? 'Enter username' : null,

              ),

              TextFormField(

                controller: \_addressController,

                decoration: InputDecoration(labelText: 'Address'),

                validator: (value) => value!.isEmpty ? 'Enter address' : null,

              ),

              TextFormField(

                controller: \_passwordController,

                decoration: InputDecoration(labelText: 'Password'),

                obscureText: true,

                validator: (value) => value!.length < 6 ? 'Min 6 chars' : null,

              ),

              TextFormField(

                controller: \_confirmPasswordController,

                decoration: InputDecoration(labelText: 'Confirm Password'),

                obscureText: true,

                validator: (value) => value != \_passwordController.text

                    ? 'Passwords don\'t match'

                    : null,

              ),

              SizedBox(height: 20),

              ElevatedButton(

                onPressed: \_registerUser,

                child: Text('Sign Up'),

              ),

              TextButton(

                onPressed: () => Navigator.pushReplacement(

                  context,

                  MaterialPageRoute(builder: (\_) => Login()),

                ),

                child: Text('Already have an account? Login'),

              )

            ],

          ),

        ),

      ),

    );

  }

}

