Flutter projects ideas

1-todo app tracks:

 floatingActionButton: FloatingActionButton(

              onPressed: () {

                if (cubit.isBottomSheetShown) {

                  if (formkey.currentState!.validate()) {

                    cubit.insertToDatabase(

                        date: taskDatecontr.text,

                        time: taskTimecontr.text,

                        title: taskTitlecontr.text);

                    print("object");

                  }

                } else {

                  scaflkey.currentState!

                      .showBottomSheet((context) => Container(

                            child: Form(

                              key: formkey,

                              child: Column(

                                mainAxisSize: MainAxisSize.min,

                                children: [

                                  defaultFormText(

                                      control: taskTitlecontr,

                                      type: TextInputType.name,

                                      validator: (value) {

                                        if (value.isEmpty) {

                                          return "please write task Title";

                                        }

                                        return null;

                                      },

                                      label: "task Title",

                                      prefix: Icons.add\_location),

                                  SizedBox(

                                    height: 10,

                                  ),

                                  defaultFormText(

                                      control: taskTimecontr,

                                      type: TextInputType.name,

                                      onTap: () {

                                        showTimePicker(

                                                context: context,

                                                initialTime: TimeOfDay.now())

                                            .then((value) {

                                          taskTimecontr.text =

                                              value!.format(context).toString();

                                        });

                                      },

                                      validator: (value) {

                                        if (value.isEmpty) {

                                          return "please write task Time";

                                        }

                                        return null;

                                      },

                                      label: "task Time",

                                      prefix: Icons.watch\_later\_outlined),

                                  SizedBox(

                                    height: 10,

                                  ),

                                  defaultFormText(

                                      control: taskDatecontr,

                                      type: TextInputType.name,

                                      onTap: () {

                                        showDatePicker(

                                          context: context,

                                          initialDate: DateTime.now(),

                                          firstDate: DateTime.now(),

                                          lastDate:

                                              DateTime.parse('2023-12-01'),

                                        ).then((value) {

                                          taskDatecontr.text =

                                              DateFormat.yMMMd().format(value!);

                                        });

                                      },

                                      validator: (value) {

                                        if (value.isEmpty) {

                                          return "please write task Date";

                                        }

                                        return null;

                                      },

                                      label: "task Date",

                                      prefix: Icons.date\_range),

                                  SizedBox(

                                    height: 10,

                                  ),

                                ],

                              ),

                            ),

                          ))

                      .closed

                      .then((value) {

                    cubit.changeBottomSheet(isshow: false, icon: Icons.edit);

                  });

                  cubit.changeBottomSheet(isshow: true, icon: Icons.add);

                }

              },

              child: Icon(cubit.flotIcon),

            ),

////////////////////////////////////////////////////

Dismissible

علشان نعمل deleteFromdatabase

Database:

  late Database database;

  void createdatabade() {

    openDatabase(

      "kareem.db",

      version: 1,

      onCreate: (database, version) {

        print("database has created");

        database

            .execute(

                'CREATE TABLE tasks (id INTEGER PRIMARY KEY,title TEXT,date TEXT,time TEXT,status TEXT)')

            .then((value) {

          print("Table has created");

        }).catchError((Error) {

          print(Error);

        });

      },

      onOpen: (database) {

        print("database has opened");

        getDataFromDatabase(database);

      },

    ).then((value) {

      database = value;

      emit(CreateDataBaseState());

    });

  }

  insertToDatabase({

    required String title,

    required String date,

    required String time,

  }) async {

    await database.transaction((txn) {

      return txn

          .rawInsert(

              'INSERT INTO tasks(title, date, time, status) VALUES( "$title", "$date", "$time", "new" )')

          .then((value) {

        print("$value data is inserted");

        emit(TodoInsertDataBaseState());

        getDataFromDatabase(database);

      }).catchError((error) {

        print(error);

      });

    });

  }

  void getDataFromDatabase(database) {

    newTasks = [];

    doneTasks = [];

    archiveTasks = [];

    database.rawQuery('SELECT \* FROM tasks').then((value) {

      value.forEach((element) {

        if (element['status'] == 'new')

          newTasks.add(element);

        else if (element['status'] == 'done')

          doneTasks.add(element);

        else

          archiveTasks.add(element);

        // print(element['status']);

      });

      emit(GetDataBaseState());

    });

  }

  void updateData({required String status, required int id}) {

    database.rawUpdate('UPDATE tasks SET status = ? WHERE id = ?',

        ['$status', id]).then((value) {

      getDataFromDatabase(database);

      emit(UpdateDataBaseState());

    });

  }

  void deleteFromdatabase({required int id}) {

    database.rawDelete('DELETE FROM tasks WHERE id = ?', [id]).then((value) {

      getDataFromDatabase(database);

      print('deleted');

      emit(deleteDataBaseState());

    }).catchError((error) {

      print(error);

    });

  }

}