# Flutter Contact Book App

Name: Fatma Ahmed Hammad

Aim: This app allows users to add, view, edit, and delete contact details.

In my application, I used MVVM (Model–view–viewmodel) architectural pattern to make it easier to deal with structural application.

In part Model, I created class as model named ContactModel contains all factors which belong to every user such as id, name, phone number, email address and take them as parameters in constructor of this class to make it easier to call object of this class than calling them all individually every time I want to read or store data in them.

In part View, I created four screens and them reusable widgets. First Screen, Home Screen contains app bar with its name, body contains column contains in the top refresh button that every time I clicked on it refresh list of contacts to get data of it and display it as list in widget named contactsBuilder take list of ContactModel as parameter this widget contains ConditionalBuilder that get it from external package to check list in its condition attribute if is empty it executes what is in its attribute fallback that show menu icon with text said No Contacts Yet, Please Add Some Contacts in center of home screen but if list does not empty that showed list view of data of every contact of users in list such as first character in its name in circle avatar and beside it showed name and phone number of user all item of list was wrapped with Dismissible widget to make it available to delete this item by dragging it and also this item wrapped with GestureDetector widget to make it available by clicking on it to navigate to Details Screen if I want to display full details of contact, and in Home Screen there was another button as FloatingActionButton to make it available by clicking on it to navigate to Add Contact Screen. Second Screen, Add Screen contains app bar with its name, body contains form that contains column contains three text form field that makes available for user to enter his data (name, phone number, email address) that wanted to store in local database in application, this widget was existing as custom widget named defaultFormField that because I used it many times, so I wanted to make it reusable to be easier to call it by its name only and give the properties to every one of the that was different about other, this

custom widget contains Text Form Field as widget through which I can received data that user entered it by get text of controller of each text form field and make sure validation for each field through anonymous function that was in validate attribute in this widget by check value of data that entered from user and check if each value of field was empty that is not allowed because each value must be non-empty and in phone field check if its numbers was equal eleven number or not that make it validated and in email field check valid email format that by each email was contains @ symbol to make sure that was valid, and each field contains its name and icon that different of others and when click on each one of them that show type of keyboard that suitable for the types of data to be entered and also in add screen there was button named save that when click on it that was check validate of current state of form that controlled of body of screen by key of form and make sure if valid that make available to insert all data was entered in local database and navigate to home screen to click on refresh button to refresh list and get data that stored in local database and showed in screen (that was presented newest item was added in list), when I wanted to know all details of any contact, I just click of any item that showed in screen that make available to navigate to Contact Details Screen to Display the full details of a current contact. Third Screen, Details Screen contains app bar with its name, body contains column contains circle avatar with first character of contact's name and three rows, first row shows name and its value, second row show phone number and its value and third row show email address and its value and in the end of screen there was two button one of them named delete when click on it that delete the current contact from local database and navigate to home screen to click on refresh button to refresh list and get data that stored in local database and showed in screen (that was presented current item was deleted from list) and also in details screen, there was another button named edit when click on it I navigate to Edit Contact Screen to allow editing of an existing contact's details and, I passed value of id of this current item to this screen because every edit of contact details belongs to this contact not other contacts. Last Screen, Edit Screen contains app bar with its name, body contains all typical details of structure of add screen with the same validations and formats for widgets and has same button named save but, in this screen, it has different function because

when click on it that was check validate of current state of form that controlled of body of screen by key of form and make sure if valid that make available to update all data of this current item was entered in local database and I can make two navigations first navigate to details screen second navigate to home screen (I used two statements of navigation at one time if I want back to home screen without passing throw details screen) to click on refresh button to refresh list and get data that stored in local database and showed in screen (that was presented current item was edited with the newest data in the list).

In part View Model, I choose SQLite as local database to store contact details persistently, so I used an external package named sqflite for flutter.

First, Using setState() as State Management:

I created class named SqfliteHelper that contains all functions and details that was type of static (To make available to call them with class name does not need to create object of class then call them by object name) that I needed to deal with it I created database variable of Database type to store all local database and contactsList variable of List of Map because each record in database was about Map<String,dynamic> and first function was used to create database named createDatabase() with type of Future<void> that because What will be implemented in this function will be in the future and this function opened database Which takes path (its location) as its name.db to make this step It will take longer than the normal time to perform normal operations and at the same time all functions need to wait for it because what will be carried out next will depend on what will come from it, so I wrote await keyword before open function to delay execution of other functions until this function has a result, this is called asynchronous programming so createDatabase() was asynchronous function that was different from other functions was normal (synchronous function) The main difference when talking about operation synchronicity is in their thread usage. Synchronous processing uses only one thread (main) where it executes all operations in succession (sequentially). In contrast, each asynchronous operation is done in a different thread that reports back to the main thread with the result when complete or with an error in case of failure, leaving that thread opens to process other requests. when creating the database, create the table named contacts with for fields id as primary key, name, phone, and email.

when opening database I triggered getDataFromDatabase() to get any data stored in local database, then meaning after implementation of function ended that executes what in it so, stored its value in database variable. I used insertToDatabase() function to Insert record in a transaction by passing data that is carried by contact model as values to each record that belongs to it. To get the records I used getDataFromDatabase(), in this function I selected all elements that exists in database and then added them to list to show them in home screen, but note that I should empty this list before I selected elements because if I do not do this, it will add to what was previously stored and some elements will be duplicated, which is a mistake. I used updateData() function to update record by passing data that is carried by contact model as values to each record that belongs to it where value of id that was as pointer to update from there. I used deleteData() function to delete any record by passing value of id that is carried by contact model to record that belongs to it where value of id that was as pointer to delete this record. After I finished this class, buttons in application had special function of them for it, in add screen when click on save button, insertToDatabase() function was triggered and insert data to newest record and in details screen when click on delete button, deleteData() function was triggered and delete the current record and in edit screen when click on save button, updateData() was triggered and update data with newest values to current record. In home screen to note change of state that must be stateful widget and in its initState(), createDatabase() was triggered that because Once the home screen opens, the init() function is executed that make it available to create database if it not found or opened it if it found and when click on refresh button, in its function, setState() was triggered and inside it, getDataFromDatabase() was triggered so every time I clicked on refresh button it change state of screen and refresh list with result of getDataFromDatabase() function to show them in home screen Note from the above that database is read only once, and any change that occurs to it does not appear except by clicking on the refresh button and it takes long time to show result.
This is in the code of second comment some changes for update(edit) function in my repository.
It is better to be able to listen to the change the first time it occurs and every time it occurs. We can make it by using cubit (A Cubit is a simple and efficient way to handle state management in Flutter apps. It's a part of the broader Bloc (Business Logic Component) library, which helps you manage your app's state in a structured and organized manner.)

Second, Using cubit as State Management:

I choose cubit as another State Management, so I used an external package named flutter_bloc for flutter to deal with cubit.

I created class for states named ContactState of type abstract because I do not want to take an object from it, I just want it to be the parent of a group of states that inherit from it so that I can send it to cubit and I created seven states every existing state will have a need in a specific place all them extend of parent class ContactState. In class named ContactCubit extend of Cubit of type from ContactState so that I can call any one of seven states that created in this class in any position I needed it in cubit class so constructor of ContactCubit I used super constructor because eliminate the need to repeat initialization logic for inherited members in each subclass constructor so I pass state named ContactInitialState to it because it was initial state of class states, and I created static function to take instance of cubit class to make in available in every position I needed then all functions I created before in SqfliteHelper class and in every class, I called one or two of states that I created before in class states and to allow for the possibility of notifying listeners I used emit() function and pass what state I wanted as parameter to it because emit updates the state to the provided state and in createDatabase() function I called ContactCreateDatabaseState() and so on for the rest of the functions after I ended dealing with all functions, I backed to all function I called them by SqfliteHelper and update them to called by cubit to make it work without any error I must provide cubit in position can every screens can listen to it and this position was main() function so I wrapped MyApp() with BlocProvider and in create attribute I called ContactCubit() with createDatabase() to call it when I opened application to create database immediately such as what I do in init() function when I used setState() as State Management and in home screen I wrapped column in its body with BlocConsumer to in its listen attribute I can listen to states and check if state is ContactGetDatabaseLoadingState() This means that the database is still loading so I can show a loading icon to express to the user an operation, and in builder attribute build screen based on state so this screen return to Stateless Widget and refresh button will not be needed to change state (hot reload) because it happened by listener and all functions, I called them before by SqfliteHelper I called them by cuit that means get function I created before to make instance of cubit or by BlocProvider.of<ContactCubit>(*context*)