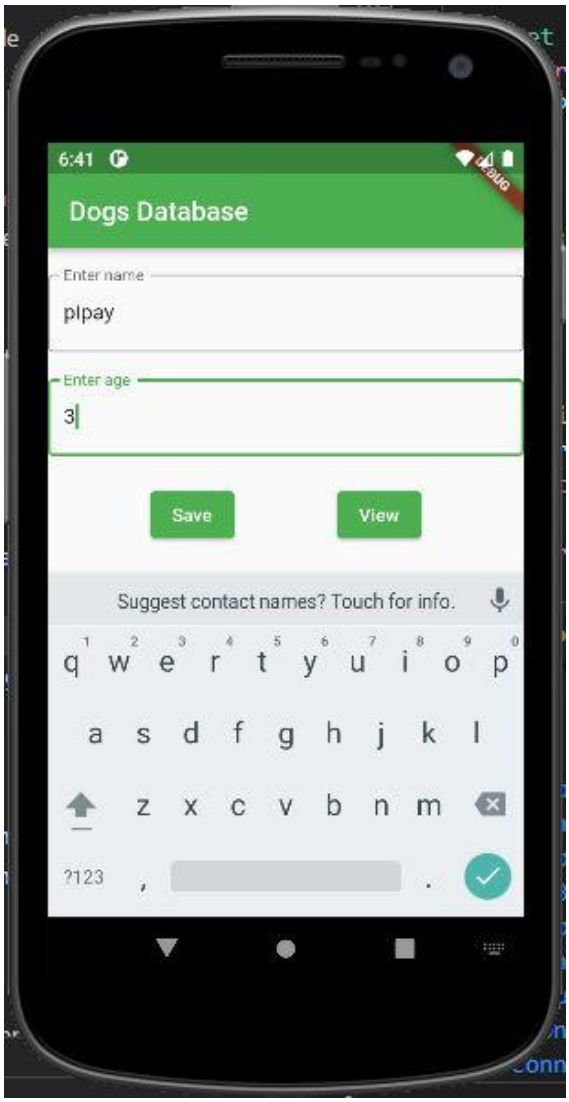
Persistence

For this week's topic, you are to persist data with SQLite. Below is a step by step guide for you to understand this topic and how to connect your flutter application to a database.

1. Review previous weeks' topics.
    a. You might also want to review or refresh basic topics from CMSC 22 regarding abstraction, encapsulation and inheritance.
    b. If you have time, you may watch this video which combines reviews for object oriented programming and the basics of building a flutter application. You may also skip parts of the video that you think isn't necessary for your review.
2. If you are using a browser as a device to render your flutter applications in previous topics, you need to create an emulator device or use an actual android phone to test your flutter applications that uses persistence.
3. Study persistence and follow the instructions on this link which demonstrates the basics of using sqflite to insert, read, update, and remove data.
4. You may be wondering, how can I use the code I've implemented in step 3? For you to be able to utilize it, you need a user interface. You may create your own user interface (using the topics discussed from previous weeks) as long as it has input fields and buttons to process your input data just like the screenshot below. For the sake of learning, I highly encourage you to create your own interface from scratch. But if you're lost and quite short of time, you may use the user interface I have created that can be found inside the *user interface template* folder (main.dart).



5. After having your user interface for user input, let us create a new dart file named DBHelper and copy your code from db_test dart file or you may just rename the db_test.dart to DBHelper.dart.
6. Create a new dart file named Dog.dart. This must contain your code for the dog model. To do this, simply copy the Dog class from the DBHelper and remove it in the DBHelper.
7. In the DBHelper, remove lines of codes for creating, updating and deleting the dog named Fido (lines 92-119). We no longer need those as we will implement it in the main dart file later. Don't forget to import the Dog.dart in your DBHelper.dart file
8. In order to access database functions, we need a class to instantiate objects that will call these database methods. We already have the database methods in the DBHelper, we just need to replace the main method

DBHelper.dart

```
import 'dart:async';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'package:flutter/widgets.dart';
import './Dog.dart';

void main() async {
```

with

```
import 'dart:async';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'package:flutter/widgets.dart';
import './Dog.dart';

class DBHelper {
```

9. In your DBHelper class, remove the widgets flutter binding and paste it to the main method of main.dart

main.dart

```
import 'package:flutter/material.dart';
import './Dog.dart';
import './DBHelper.dart';
void main() {
  WidgetsFlutterBinding.ensureInitialized();
  runApp(const MyApp());
}
```

10. In the DBHelper class, create a method to initialize and return your database. Put inside this method the lines of codes for the opendatabase method and return the database.

DBHelper.dart

```
Future<Database> initializeDB() async {
    // Open the database and store the reference.
    final database = openDatabase(
      // Set the path to the database. Note: Using the `join` function
from the
      // `path` package is best practice to ensure the path is correctly
      // constructed for each platform.
      join(await getDatabasesPath(), 'doggie_database.db'),
      // When the database is first created, create a table to store dogs.
      onCreate: (db, version) {
        // Run the CREATE TABLE statement on the database.
        return db.execute(
          'CREATE TABLE dogs(id INTEGER PRIMARY KEY, name TEXT, age
INTEGER)',
        );
      },
      // Set the version. This executes the onCreate function and provides
a
      // path to perform database upgrades and downgrades.
      version: 1,
    );

    return database;
  }
```

11. To fix the remaining errors in the DBHelper.dart, replace all instance of

```
final db = await database;
```

with

```
final db = await initializeDB();
```

12. Now, we go to the main.dart file so that we can insert data to our database. To do that, we need to create an instance of the DBHelper inside the _MyStatefulWidgetState class

```
class _MyStatefulWidgetState extends State<MyStatefulWidget> {
```

Insert

```
  //create a DBHelper object to access database functions
```

```
  DBHelper db = DBHelper();
```

Don't forget to import DBHelper.dart in your main.dart file

13. Inside your save button widget, put in the onPressed() function an instance of a Dog and insert it to your database. Don't forget to import the Dog.dart file

```
//instantiate a dog object
Dog dog1 = Dog(id: counter++, name: _controller.text,
age:int.parse(_controller2.text));

//insert dog object to database
db.insertDog(dog1);
```

14. To instantiate a Dog object, parameters require an id, name and age. Create a counter variable that automatically generates ids for dogs.

```
  //create a DBHelper object to access database functions
  DBHelper db = DBHelper();

  //counter variable for Dog id
  int counter = 0;
```

15. When you run your flutter application and try to insert data to the database, it only shows you 'Processing Data'. Don't forget to clear the input fields after inserting data to your database inside the onPressed function. Here's a sample code for the save button widget if you used the templated user interface (the main.dart template file)

```
Widget buildSaveButton() {
    return ElevatedButton(
        child: const Text('Save'),
        onPressed: () {
          if (_formKey.currentState!.validate()) {

              //instantiate a dog object
              Dog dog1 = Dog(id: counter++, name: _controller.text,age
:int.parse(_controller2.text));

              //insert dog object to database
              db.insertDog(dog1);
```

```
            ScaffoldMessenger.of(context).showSnackBar(
              const SnackBar(content: Text('Processing Data')),
            );
            //clear text on controllers after successful insert of data to
database
            _controller.clear();
            _controller2.clear();
          }
        });
  }
```

16. Create a new dart file (ShowTextPage.dart) that will render the second screen containing a list of dogs from the database. You may use the template provided for the ShowTextPage.dart

17. To access the database inside _ShowTextPageState, you need to create an instance of the DBHelper class and create a variable that will store the returned data from the database. Don't forget to import the DBHelper.dart and Dog.dart

```
class _ShowTextpageState extends State<ShowTextpage> {

  //create a DBHelper object to access database functions
  DBHelper db = DBHelper();

  //create a future list of dog that will store all the dog data from
database
  late Future<List<Dog>> myDog;
```

18. Initialize myDog list by getting data from the database.

```
//initialize myDog list by getting data from database
    myDog = db.dogs();
```

19. Once you already have access to the data returned from the database, you are free to choose how you are going to render it to the user.

20. The provided ShowTextPage template only shows the names of the dogs but you can modify it and do a lot more than that, you can also access the age using the age attribute and the right index from the list. You can add delete and update icons as well, as long as you have access to the id of the dog that you want to

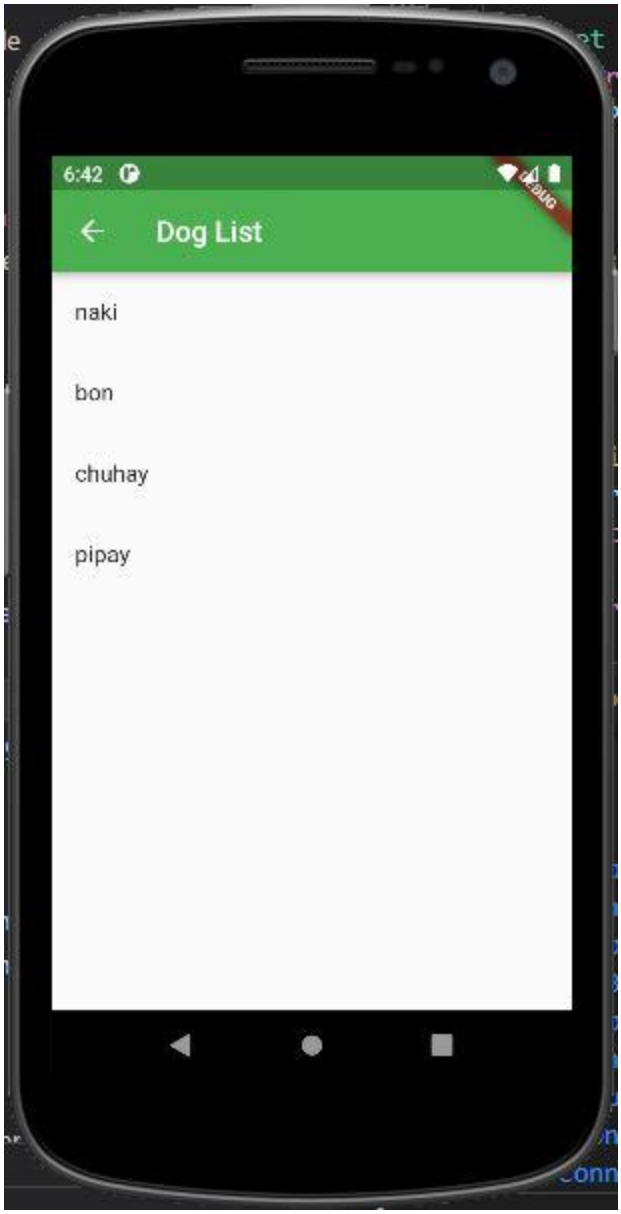
delete or update. If in the middle of this topic you got lost, don't worry, a working code (the repository you've cloned from the exercise-03-template) has also been provided for your reference.

21. Finally, we need to tell the main dart file that when the view button is clicked, it should render the ShowTextPage.dart. To do that, we can use the Navigator.push() method (You can read more about navigation and routing here). Don't forget to import ShowTextPage.dart in your main.dart file.

    main.dart

```
 Widget buildViewButton() {
    return ElevatedButton(
      onPressed: () {
        //navigates to ShowTextpage that contains list of database content
        Navigator.push(
             context, MaterialPageRoute(builder: (context) =>
ShowTextpage()));
      },
      child: const Text("View"),
    );
  }
```

22. Here's a sample screen using the template ShowTextPage provided when the view button has been clicked by the user.



If you have any questions or clarifications, don't hesitate to contact your laboratory instructor.

References:

- Persist data with SQLite - https://docs.flutter.dev/cookbook/persistence/sqlite
- Flutter Crash Course for Beginners 2021 - Build a Flutter App with Google's Flutter & Dart - https://www.youtube.com/watch?v=x0uinJvhNxI
- Using SQLite in Flutter - https://petercoding.com/flutter/2021/03/21/using-sqlite-in-flutter/