

Cross Platform Application Development (Merged - SEZG585/SSZG585) Assignment: 1

Name: Matin Shaikh

ID No.-2022MT93558

Email: 2022mt93558@wilp.bits-pilani.ac.in

This assignment involves creating a Flutter application integrated with the **Back4App** service for backend functionality. It requires an understanding of both Flutter development and using APIs for backend communication. Here's how you can approach each step:

Before proceeding to the steps, here are the prerequisites:

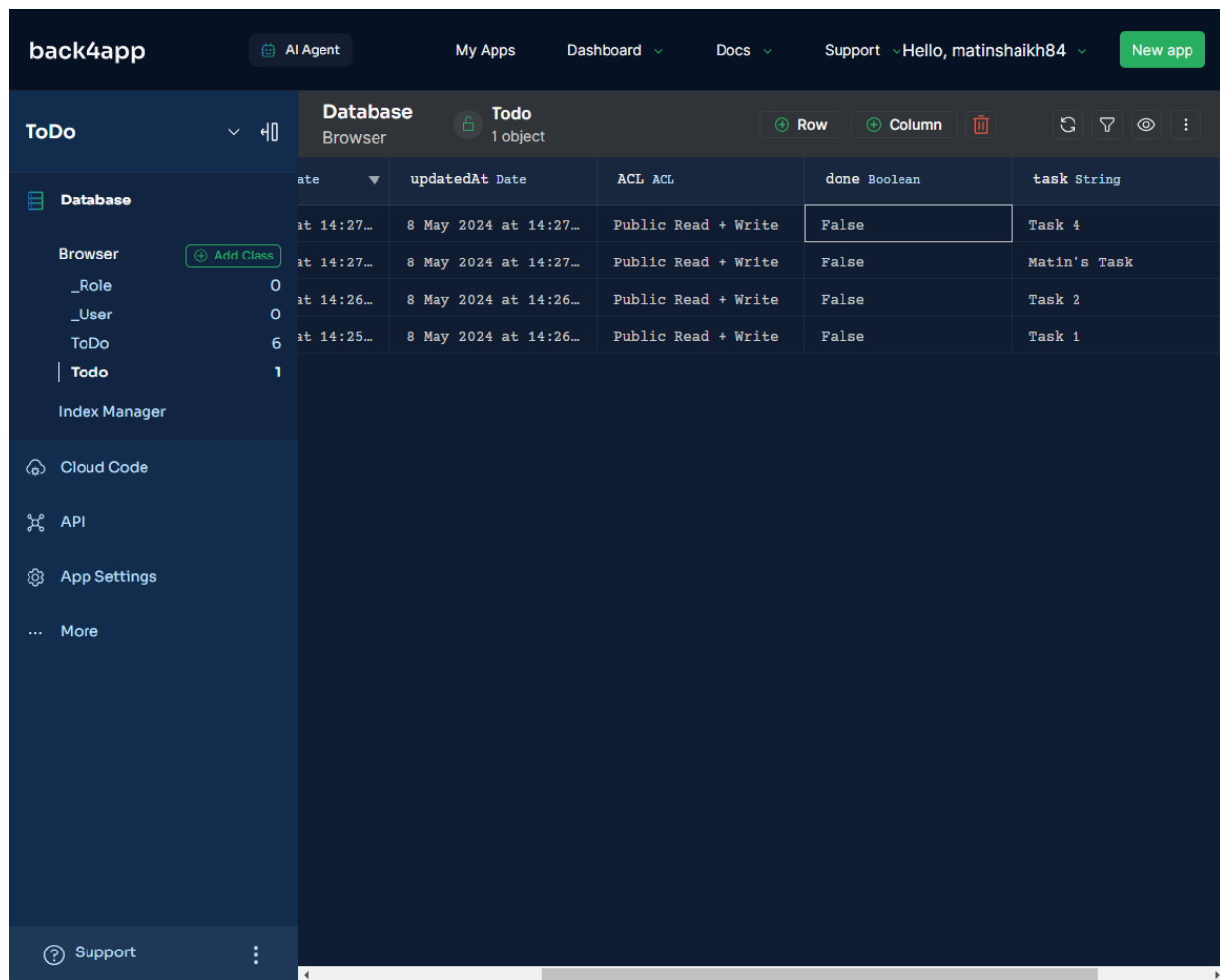
- [Flutter version 2.2.x or later](#)
- [Android Studio](#) or [VS Code installed](#) (with [Plugins](#) Dart and Flutter)
- An app created on Back4App.
Note: Follow the [New Parse App Tutorial](#) to learn how to create a Parse App on Back4App.
- A Flutter app connected to Back4app.
Note: Follow [the Install Parse SDK on Flutter project](#) to create a Flutter Project connected to Back4App.
- A device (or virtual device) running Android or iOS.

```
D:\FlutterApp\flutter\bin>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.19.5, on Microsoft Windows [Version 10.0.22631.3296], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.9.6)
[✓] Android Studio (version 2023.2)
[✓] VS Code (version 1.89.0)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!
```

Step 1: Set Up Back4App:

1. Go to the Back4App website and sign up for a new account or log in if you already have one.
2. Once logged in, create a new app by following the prompts provided on the platform.
3. After creating the app, navigate to your app's Dashboard and find the Database Browser.
Here, create a new class called "**ToDo**". Add two columns to this class:
 - a. **title**: This will be of type **String**.
 - b. **done**: This is also a **String Boolean**.



Step 2: Flutter Setup:

1. Set up a new Flutter project in your IDE (preferably Visual Studio Code or Android Studio).
2. Open the **pubspec.yaml** file in your project and add the necessary dependencies. For this project, you would need the **parse_server_sdk_flutter** package. Add it under dependencies and Run flutter pub get to install the new dependency .

```

25 # Dependencies specify other packages that your package needs in order to work.
26 # To automatically upgrade your package dependencies to the latest versions
27 # consider running `flutter pub upgrade --major-versions`. Alternatively,
28 # dependencies can be manually updated by changing the version numbers below to
29 # the latest version available on pub.dev. To see which dependencies have newer
30 # versions available, run `flutter pub outdated`.
31 dependencies:
32   parse_server_sdk_flutter: ^4.0.0
33   flutter:
34     sdk: flutter
35 
```

3. Initialize the Parse SDK in your Flutter app. This usually goes in the **main.dart** file's main function or the initState method of your widget. You'll need your Back4App app's Application Id and Client Key, which can be found in the app's settings on the Back4App platform.

Step 3: Create Todo App Template:

```
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:parse_server_sdk/parse_server_sdk.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  final keyApplicationId = '5DBvjcMkzer3l72KSGYq0Qd1M2DDuLJmUHJDUpQ1';
  final keyClientKey = 'H1U3fKcn63bHF1mYl0Gw1DKDN5jqGqCghvus8DhW';
  final keyParseServerUrl = 'https://parseapi.back4app.com';

  await Parse().initialize(keyApplicationId, keyParseServerUrl,
    clientKey: keyClientKey, debug: true);

  /*var loginDetails = ParseObject('ToDo')
  ..set('Title', 'ABC')
  ..set('Description', 'Test')
  ..set('Status', 'True');
  await loginDetails.save();*/

  runApp(MaterialApp(
    home: Home(),
  ));
}

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  final todoController = TextEditingController();
```

```

void addToDo() async {
  if (todoController.text.trim().isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text("Empty title"),
      duration: Duration(seconds: 2),
    ));
    return;
  }
  await saveTodo(todoController.text);
  setState(() {
    todoController.clear();
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Matin's Todo List"),
      backgroundColor: Colors.blueAccent,
      centerTitle: true,
    ),
    body: Column(
      children: <Widget>[
        Container(
          padding: EdgeInsets.fromLTRB(17.0, 1.0, 7.0, 1.0),
          child: Row(
            children: <Widget>[
              Expanded(
                child: TextField(
                  autocorrect: true,
                  textCapitalization: TextCapitalization.sentences,
                  controller: todoController,
                  decoration: InputDecoration(
                    labelText: "New To Do",
                    labelStyle: TextStyle(color: Colors.blueAccent)),
              ),
            ],
          ),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              onPrimary: Colors.white,
              primary: Colors.blueAccent,
            ),
            onPressed: addToDo,
            child: Text("Insert")),

```



```

        varDone ? Icons.check : Icons.error),
        backgroundColor:
            varDone ? Colors.green : Colors.blue,
        foregroundColor: Colors.white,
    ),
    trailing: Row(
        mainAxisAlignment: MainAxisAlignment.min,
        children: [
            Checkbox(
                value: varDone,
                onChanged: (value) async {
                    await updateTodo(
                        varTodo.objectId!, value!);
                    setState(() {
                        //Refresh UI
                    });
                },
            ),
            IconButton(
                icon: Icon(
                    Icons.delete,
                    color: Colors.blue,
                ),
                onPressed: () async {
                    await deleteTodo(varTodo.objectId!);
                    setState(() {
                        final snackBar = SnackBar(
                            content: Text("Todo deleted!"),
                            duration: Duration(seconds: 2),
                        );
                        ScaffoldMessenger.of(context)
                            ..removeCurrentSnackBar()
                            ..showSnackBar(snackBar);
                    });
                },
            ),
        ],
    ),
),
);
});
}
}
})))
],
),
);
);

```

```

}

Future<void> saveTodo(String title) async {
  /*await Future.delayed(Duration(seconds: 1), () {});*/
  final todo = ParseObject('Todo')..set('title', title)..set('done', false);
  await todo.save();
}

Future<List<ParseObject>> getTodo() async {
  /*await Future.delayed(Duration(seconds: 2), () {});*/
  QueryBuilder<ParseObject> queryTodo =
    QueryBuilder<ParseObject>(ParseObject('Todo'));
  final ParseResponse apiResponse = await queryTodo.query();

  if (apiResponse.success && apiResponse.results != null) {
    return apiResponse.results as List<ParseObject>;
  } else {
    return [];
  }
}

Future<void> updateTodo(String id, bool done) async {
  /*await Future.delayed(Duration(seconds: 1), () {});*/
  var todo = ParseObject('Todo')
    ..objectId = id
    ..set('done', done);
  await todo.save();
}

Future<void> deleteTodo(String id) async {
  /*await Future.delayed(Duration(seconds: 1), () {});*/
  var todo = ParseObject('Todo')..objectId = id;
  await todo.delete();
}
}

```

Step 4: Connect Template to Back4app Project:

Updated Application Id and Client Key credentials in code in **main.dart** with the values of your project's **ApplicationId** and **ClientKey** in Back4app.

keyApplicationId = App Id

keyClientKey = Client Key

Step 5: Code for Create Object:

```
Future<void> saveTodo(String title) async {  
  final todo = ParseObject('Todo')..set('title', title)..set('done', false);  
  await todo.save();  
}
```

Step 6: Code for Read Object:

```
Future<List<ParseObject>> getTodo() async {  
  QueryBuilder<ParseObject> queryTodo =  
    QueryBuilder<ParseObject>(<ParseObject>('Todo'));  
  final ParseResponse apiResponse = await queryTodo.query();  
  
  if (apiResponse.success && apiResponse.results != null) {  
    return apiResponse.results as List<ParseObject>;  
  } else {  
    return [];  
  }  
}
```

Creating **ListView.builder** function:

```
//Get Parse Object Values  
final varTodo = snapshot.data![index];  
final varTitle = varTodo.get<String>('title')!;  
final varDone = varTodo.get<bool>('done')!;
```

Step 6: Code for Update Object:

```
Future<void> updateTodo(String id, bool done) async {  
  var todo = ParseObject('Todo')  
    ..objectId = id  
    ..set('done', done);  
  await todo.save();  
}
```

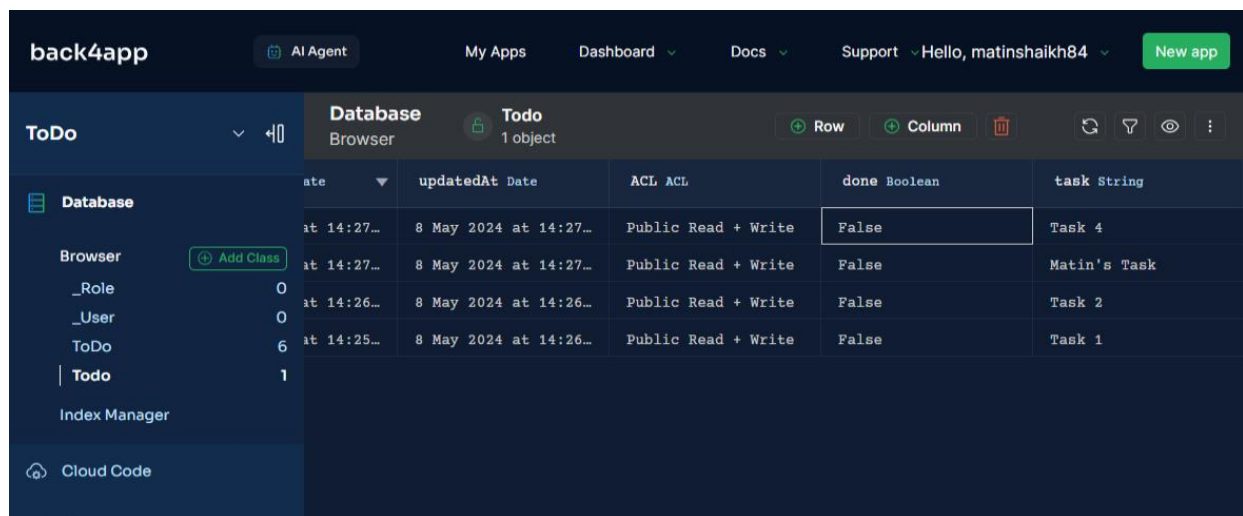

Step 7: Code for Delete Object:

```
Future<void> deleteTodo(String id) async {  
    var todo = ParseObject('Todo')..objectId = id;  
    await todo.delete();  
}
```

To Test it, click on the **Run** button in Android Studio/VSCode.

Choose a task, click on the Delete icon in Task.

The screen will be reloaded and the removed task will not be listed.



id	createdAt	updatedAt	ACL	done	task
at 14:27...	8 May 2024	at 14:27...	Public Read + Write	False	Task 4
at 14:27...	8 May 2024	at 14:27...	Public Read + Write	False	Matin's Task
at 14:26...	8 May 2024	at 14:26...	Public Read + Write	False	Task 2
at 14:25...	8 May 2024	at 14:26...	Public Read + Write	False	Task 1



Step 8:

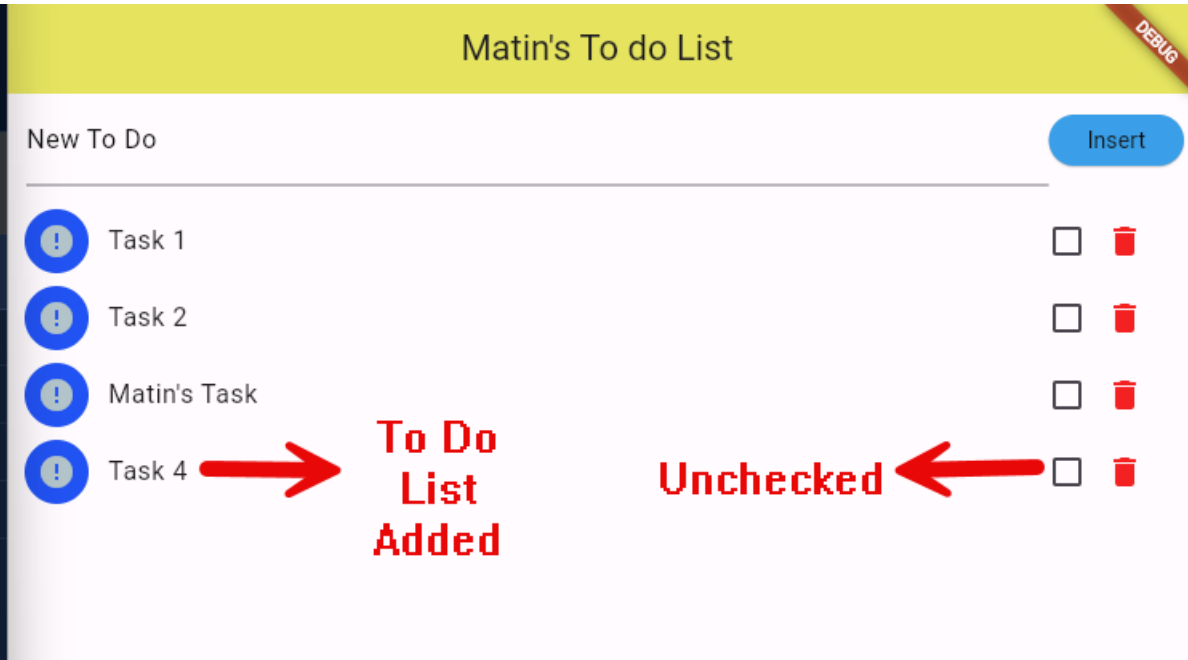
1. Implemented the ability to **Edit/Update tasks**. This could be on the task detail screen. You would need to update the **ParseObject** with the new information and save it.

2. Add functionality to **Delete tasks**. This involves removing the **ParseObject** from the database.

Edit Task:

To implement the ability to edit/update tasks in your Flutter app with Back4App, particularly changing a task's status from complete to incomplete (True to False).

In **ToDo** class in Back4App has a boolean field to represent the status of a task. It's called **done** (Boolean) Value. This field should be a boolean where true represents a completed task and false an incomplete one.



Currently **done** (Boolean) status is **False**

back4app

AI AgentMy AppsDashboard▼Docs▼Support▼Hello, matinshaikh84▼New app

ToDo	Database	Browser	ToDo	1 object	Row	Column				
	ate	▼	updatedAt	Date	ACL	ACL	done	Boolean	task	String
	at 14:27...		8 May 2024	at 14:27...	Public	Read + Write	False		Task 4	
	at 14:27...		8 May 2024	at 14:27...	Public	Read + Write	False		Matin's Task	
	at 14:26...		8 May 2024	at 14:28...	Public	Read + Write	True		Task 2	
	at 14:25...		8 May 2024	at 14:28...	Public	Read + Write	True		Task 1	

Database

Browser

Index Manager

0

0

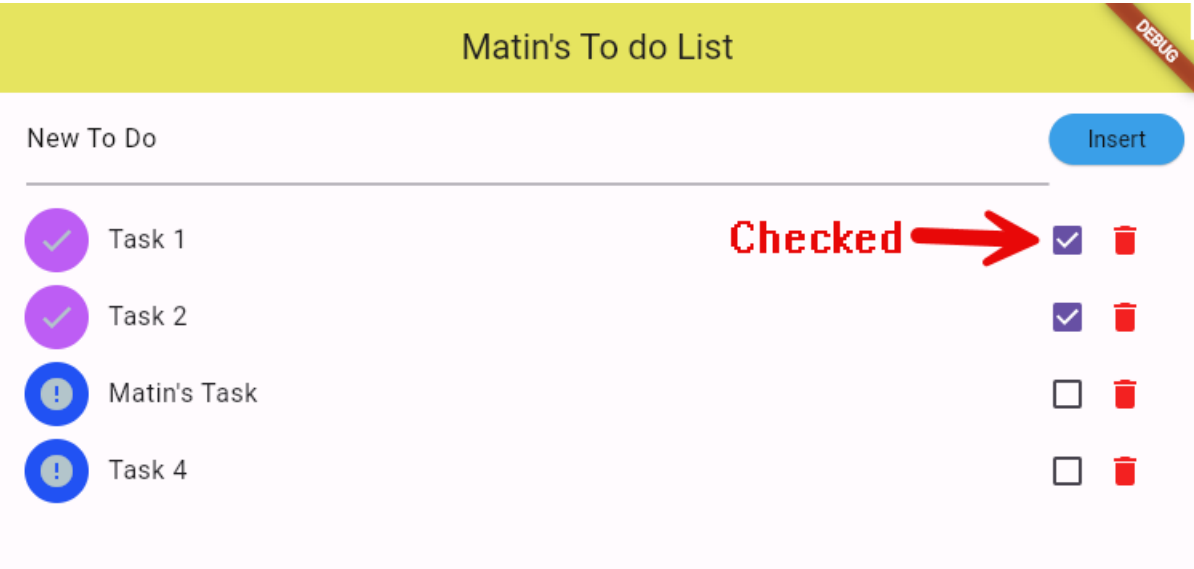
6

1

Add Class

On the task detail screen, want to display a checkbox or a toggle switch that shows the current status of the task. The user can change this status by interacting with the checkbox or switch.

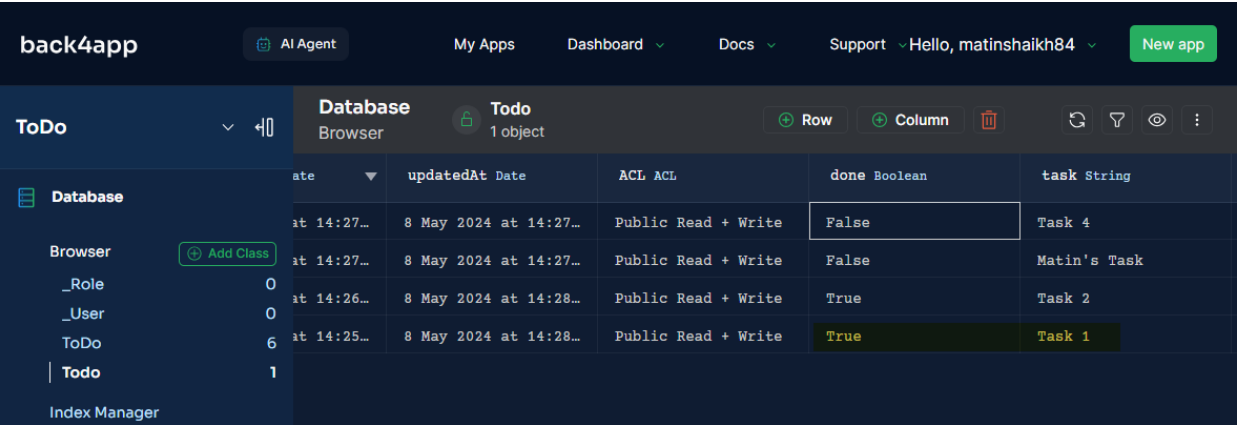
When the user updates the status of the task (like unchecking a completed task), you need to capture this event and update the task's is Completed field in Flutter app with Status **"True"** or **"False"**.



To save this updated status back to Back4App, you would use the Parse SDK. (As mentioned above).

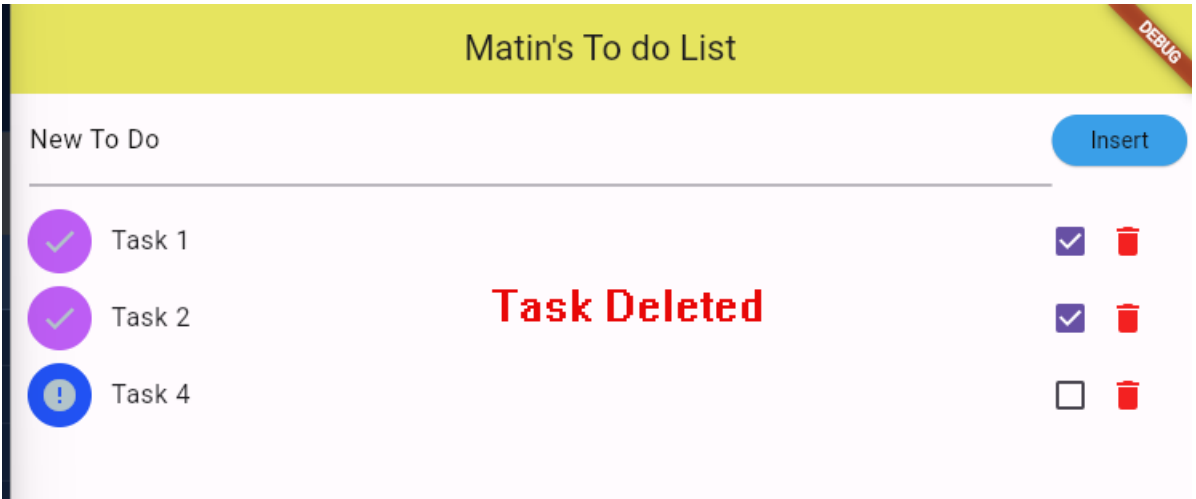
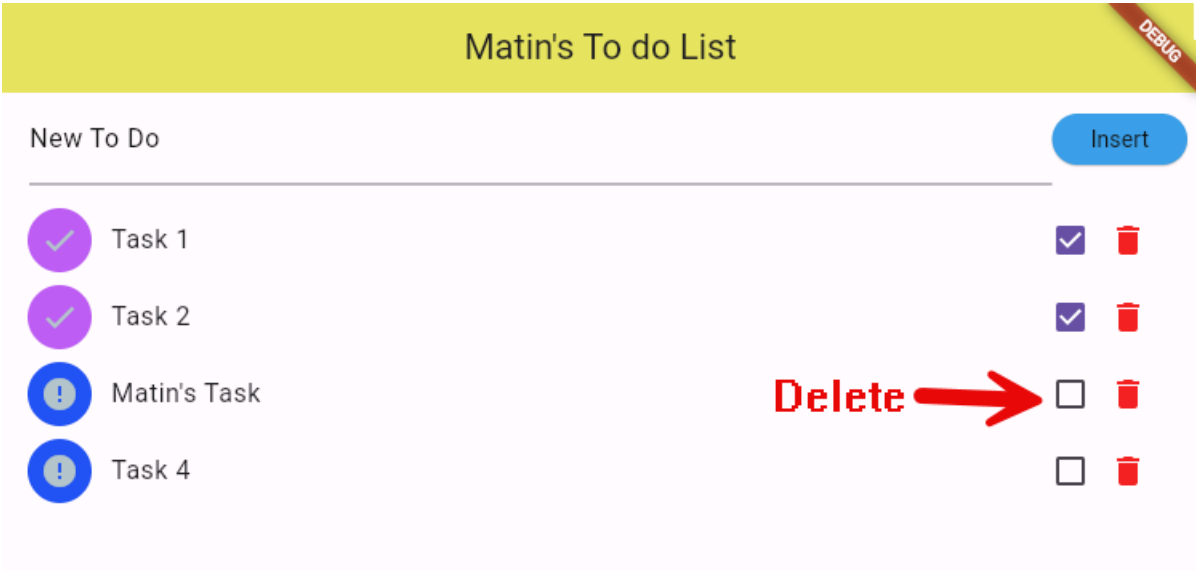
After successfully updating the task in Back4App, make sure to update your local state as well so that the UI reflects this change. If you're using a state management solution (like Provider, Bloc, etc.), you would update the state accordingly. (Please refer below screenshot).

Currently **done** (Boolean) status is **True**



Delete Task:

And this could be a button or an icon that, when tapped, triggers the deletion process. To delete a task, use the task's object ID to remove it from the database.



The screenshot shows the back4app interface. At the top, there's a navigation bar with 'back4app', 'AI Agent', 'My Apps', 'Dashboard', 'Docs', 'Support', and a user profile 'Hello, matinshaikh84'. A 'New app' button is on the right. Below the navigation bar, the 'Database' section is active, showing a 'ToDo' table with 1 object. The table has columns: _id, createdAt, updatedAt, ACL, done, and task. The data is as follows:

_id	createdAt	updatedAt	ACL	done	task
at 14:27...	8 May 2024	at 14:27...	Public Read + Write	False	Task 4
at 14:26...	8 May 2024	at 14:28...	Public Read + Write	True	Task 2
at 14:25...	8 May 2024	at 14:28...	Public Read + Write	True	Task 1

Here is the GIT Repository Link – <https://github.com/matinbits/CPADassignmet>

Recording location:

[https://drive.google.com/file/d/1RaMqJowkcgfGMzV293Wm2xJbQi-hJ0kM/view?usp=drive link](https://drive.google.com/file/d/1RaMqJowkcgfGMzV293Wm2xJbQi-hJ0kM/view?usp=drive_link)