Cross Platform Application Development (Merged - SEZG585/SSZG585) (S1-23)

Assignment 1: Flutter App with Back4App Integration

Siva Sankar A

2022MT93006

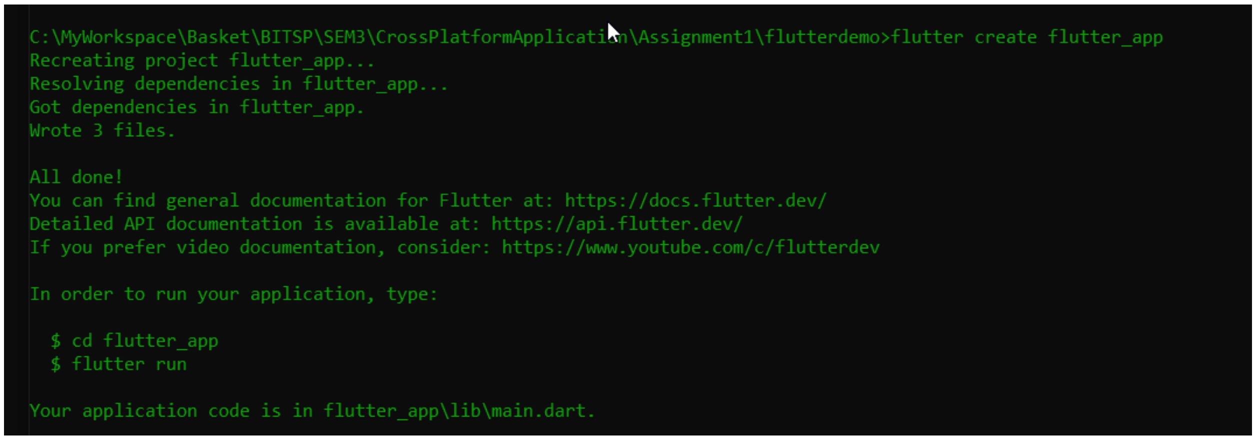
14-Nov-2023

Objective: To create a Flutter app that connects to Back4App, a Backend-as-a-Service (BaaS) platform, to manage tasks. You will be responsible for setting up the Back4App backend, creating the Flutter app, and implementing the necessary functionality to interact with the backend.

Let us start by setting up the flutter sdk on our machine. To do so follow the below steps,

- Prerequisite tools
  - Android Studio
  - Flutter app sdk
  - Account in back4app
  - Java sdk as expected by above software packages.
  - Browser: Chrome, Edge etc

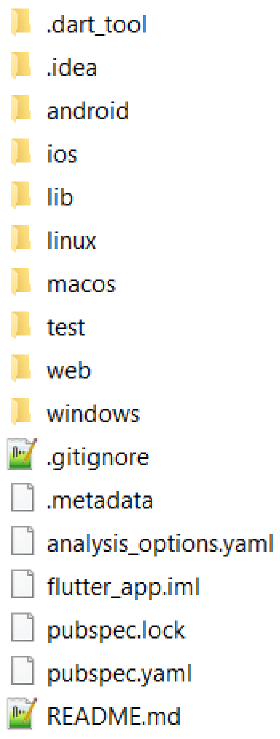
- Download flutter sdk for windows.
  - https://docs.flutter.dev/get-started/install/windows
  - Follow basic configuration steps like
    - Setting up environment path
    - Check flutter installation health using "*flutter doctor*" command.
    - Configure the Android directioy path using "*flutter config --android-studio-dir="C:\Program Files\Android\Android Studio1\bin"*"
    - Once flutter is set up then let us start creating the first application
- To create flutter application run the below command in windows command prompt as shown in Figure 1

➢ 

```
C:\MyWorkspace\Basket\BITSP\SEM3\CrossPlatformApplication\Assignment1\flutterdemo>flutter create flutter_app
Recreating project flutter_app...
Resolving dependencies in flutter_app...
Got dependencies in flutter_app.
Wrote 3 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd flutter_app
  $ flutter run

Your application code is in flutter_app\lib\main.dart.
```

Figure 1: Create flutter app

➢ Running the above command will create folder with several platform options as shown in Figure 2



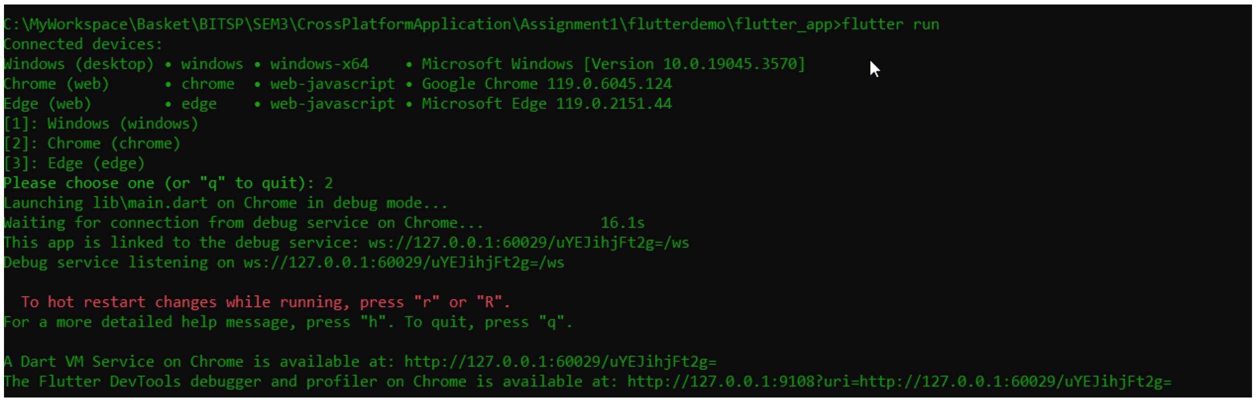
```
.dart_tool
.idea
android
ios
lib
linux
macos
test
web
windows
.gitignore
.metadata
analysis_options.yaml
flutter_app.iml
pubspec.lock
pubspec.yaml
README.md
```

➢

Figure 2: Flutter app workspace

➢ Under lib folder there is the main.dart file. This is the root of the application, or this is the file from where the application starts,

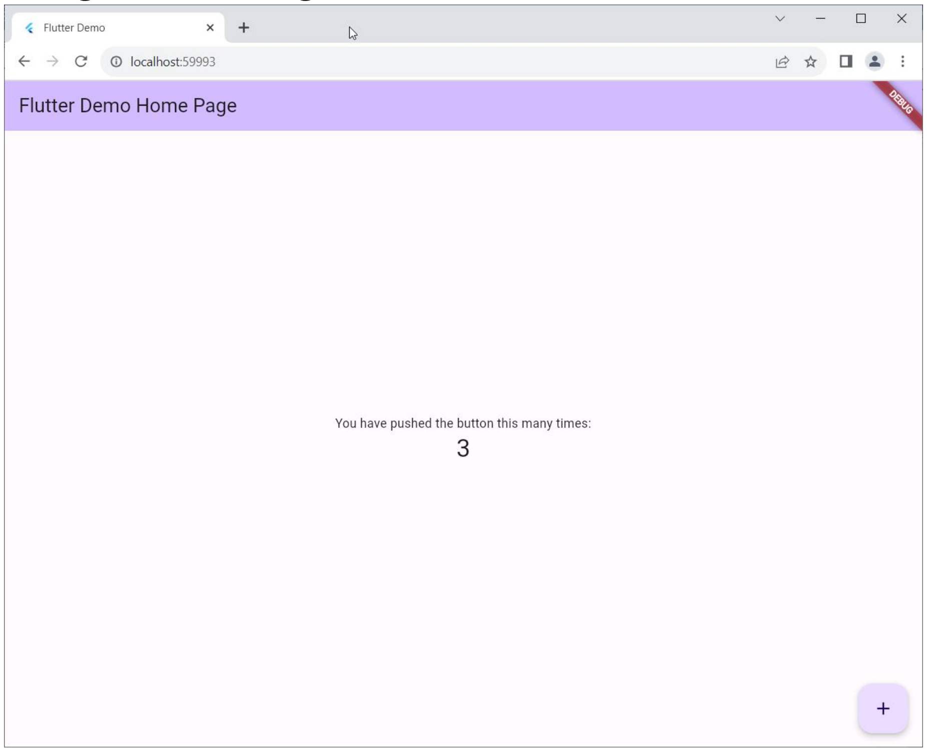
- Next, we need to run the flutter run command as shown below in Figure 3

- 

```
C:\MyWorkspace\Basket\BITSP\SEM3\CrossPlatformApplication\Assignment1\flutterdemo\flutter_app>flutter run
Connected devices:
Windows (desktop) • windows • windows-x64    • Microsoft Windows [Version 10.0.19045.3570]
Chrome (web)      • chrome  • web-javascript • Google Chrome 119.0.6045.124
Edge (web)        • edge    • web-javascript • Microsoft Edge 119.0.2151.44
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...             16.1s
This app is linked to the debug service: ws://127.0.0.1:60029/uYEJihjFt2g=/ws
Debug service listening on ws://127.0.0.1:60029/uYEJihjFt2g=/ws

  To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

A Dart VM Service on Chrome is available at: http://127.0.0.1:60029/uYEJihjFt2g=
The Flutter DevTools debugger and profiler on Chrome is available at: http://127.0.0.1:9108?uri=http://127.0.0.1:60029/uYEJihjFt2g=
```

Figure 3: Flutter run command.

- The output of “flutter run” command was chosen for 2 targets, browser and android emulation and the respective outputs are shown in Figure 4 and Figure 5

- 

Flutter Demo Home Page

You have pushed the button this many times:
3

Figure 4: Flutter run on Browser (Chrome)

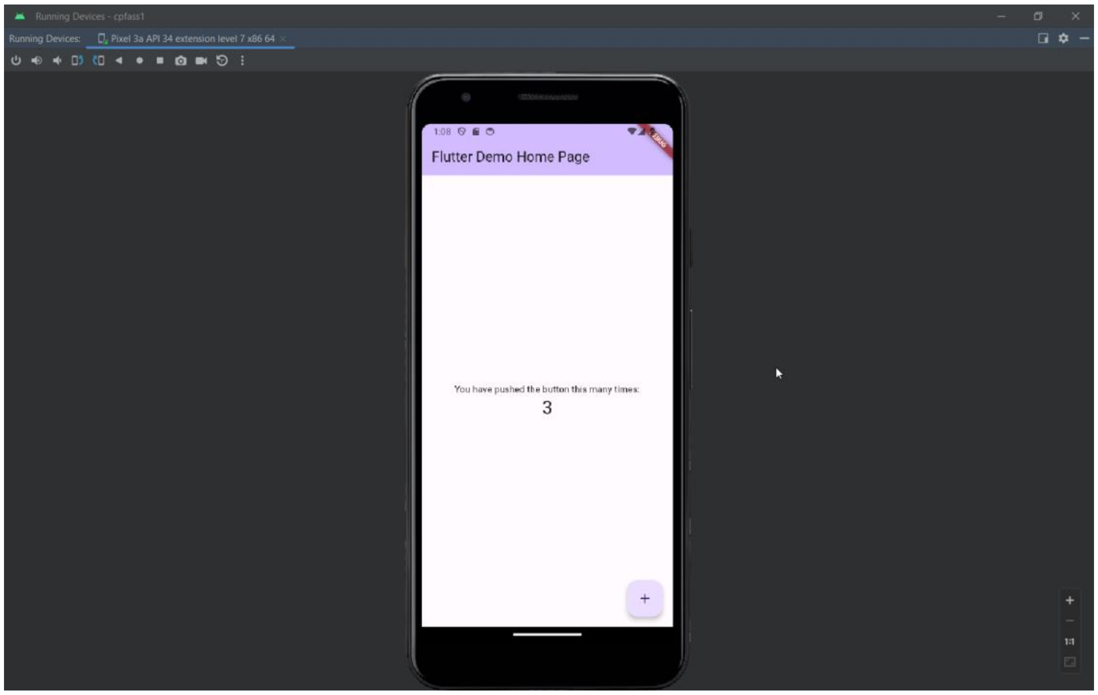
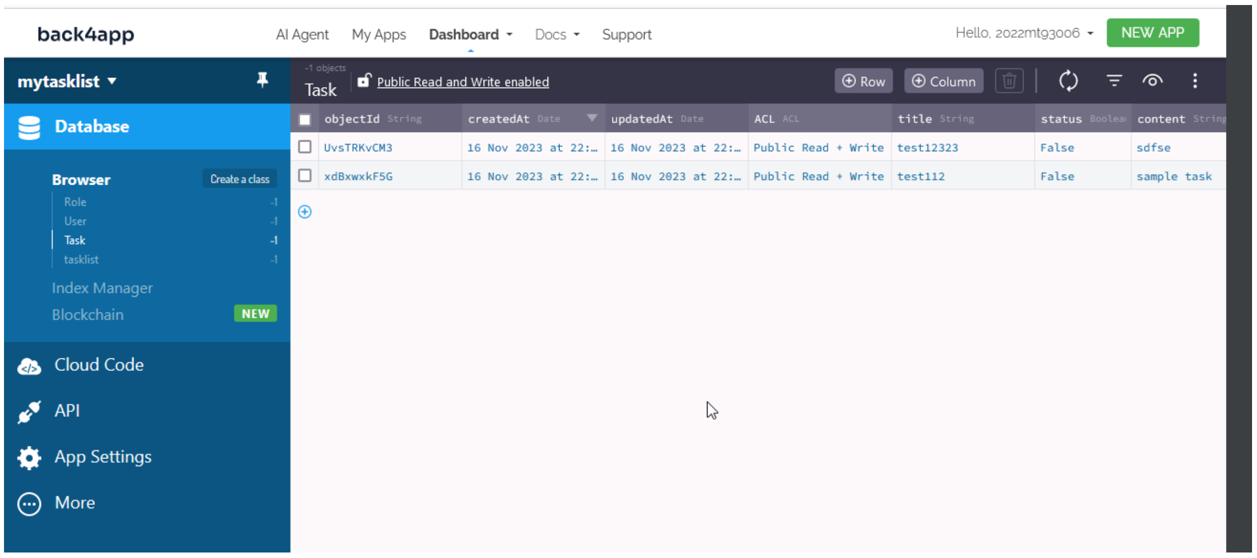
➢ 
Running Devices - cpfass1
Running Devices: Pixel 3a API 34 extension level 7 x86 64
Flutter Demo Home Page
You have pushed the button this many times:
3

Figure 5: Flutter run on Android emulation.

- ➢ So far, we have only set up the flutter sdk and tested a simple application locally,

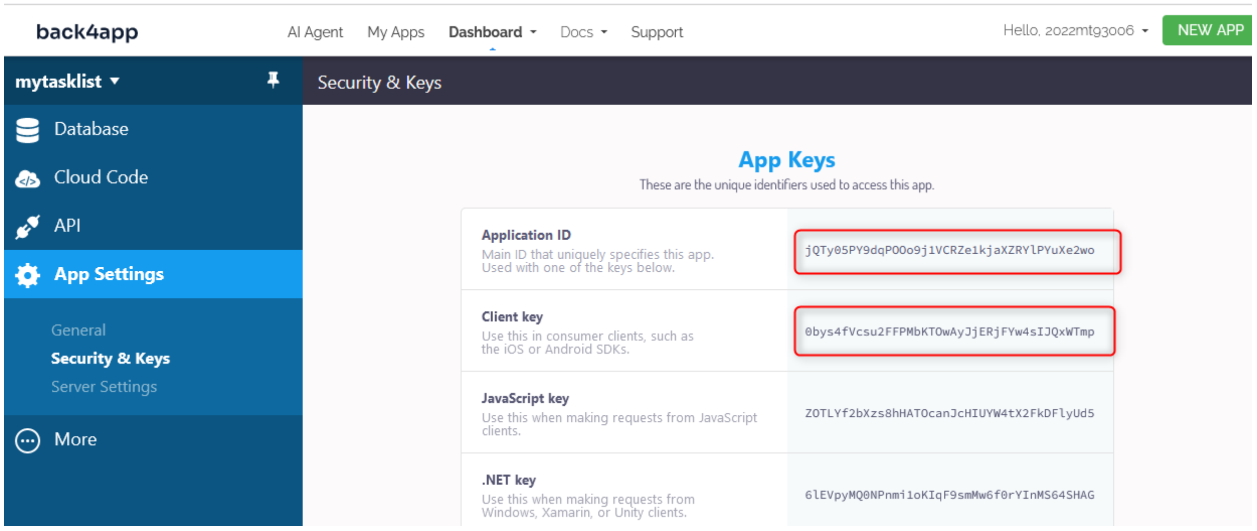
- ➢ Now let us use Back4App Backend-as-a-Service(Baas) platform that provides us with set of pre build backend services and infrastructure. We will use this to as a database in cloud to store and maintain our task list.
- ➢ Below Figure 6 shows the portal where we created simple task in back4app

➢ 

	objectId String	createdAt Date	updatedAt Date	ACL ACL	title String	status Boolean	content String
☐	UvsTRKvCM3	16 Nov 2023 at 22:…	16 Nov 2023 at 22:…	Public Read + Write	test12323	False	sdfse
☐	xdBxwxkF5G	16 Nov 2023 at 22:…	16 Nov 2023 at 22:…	Public Read + Write	test112	False	sample task

Figure 6: back4app portal

➢ From the above task, we need the access keys the application ID and the Client key for the flutter app to authenticate to back4app portal as shown in Figure 7

➢ 

App Keys

These are the unique identifiers used to access this app.

Application ID Main ID that uniquely specifies this app. Used with one of the keys below.	jQTy05PY9dqPOOo9j1VCRZe1kjaXZRYlPYuXe2wo
Client key Use this in consumer clients, such as the iOS or Android SDKs.	0bys4fVcsu2FFPMbKTOwAyJjERjFYw4sIJQxWTmp
JavaScript key Use this when making requests from JavaScript clients.	ZOTLYf2bXzs8hHATOcanJcHIUYW4tX2FkDFlyUd5
.NET key Use this when making requests from Windows, Xamarin, or Unity clients.	6lEVpyMQ0NPnmi1oKIqF9smMw6f0rYInMS64SHAG

Figure 7: Access keys

➢ Now, let us configure the main.dart file inside the flutter application for task management system for which we need the following,

➢ We need to configure the dart file with the application id and client key and the parse server url to enable the interaction of the application with the back4app backend as shown in the Figure 8

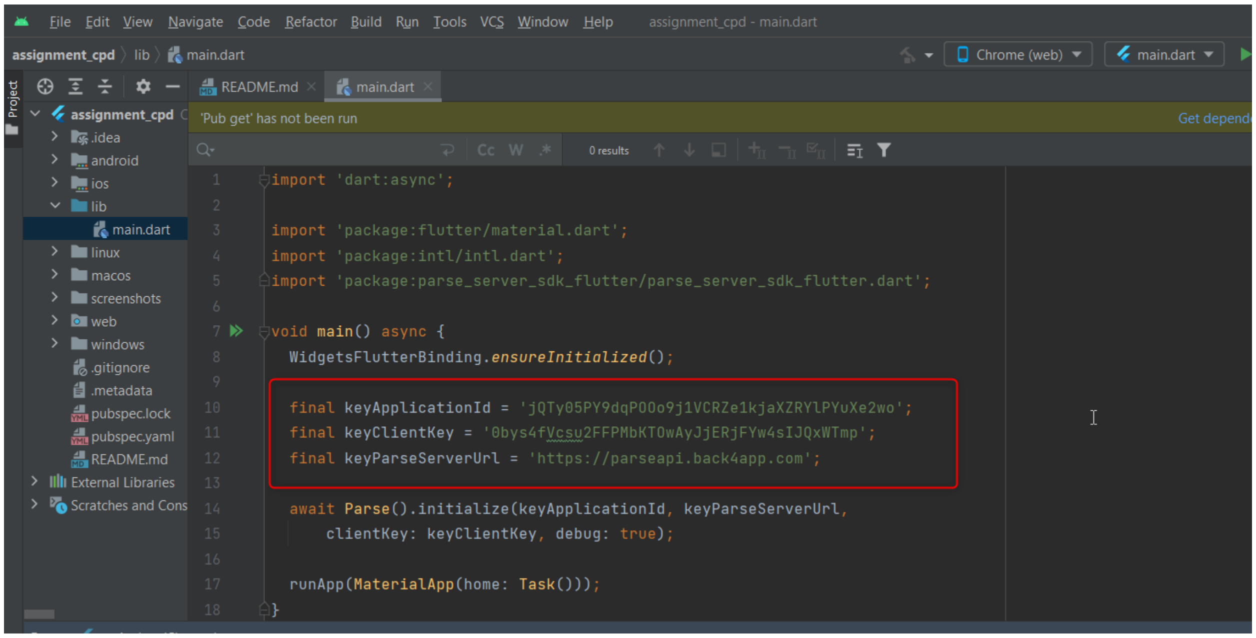
➢ 
```dart
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:parse_server_sdk_flutter/parse_server_sdk_flutter.dart';

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

  final keyApplicationId = 'jQTy05PY9dqPOOo9j1VCRZe1kjaXZRYlPYuXe2wo';
  final keyClientKey = '0bys4fVcsu2FFPMbKTOwAyJjERjFYw4sIJQxWTmp';
  final keyParseServerUrl = 'https://parseapi.back4app.com';

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

  runApp(MaterialApp(home: Task()));
}
```

Figure 8: main.dart authentication access

➢ Next we need to define screens, methods, widgets and the UI structure of the widgets as shown in Figure 9 Figure 10 Figure 11 and Figure 12

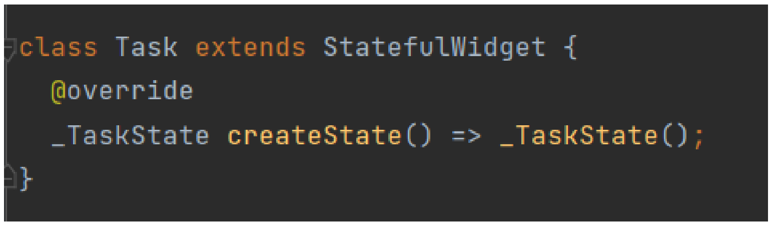
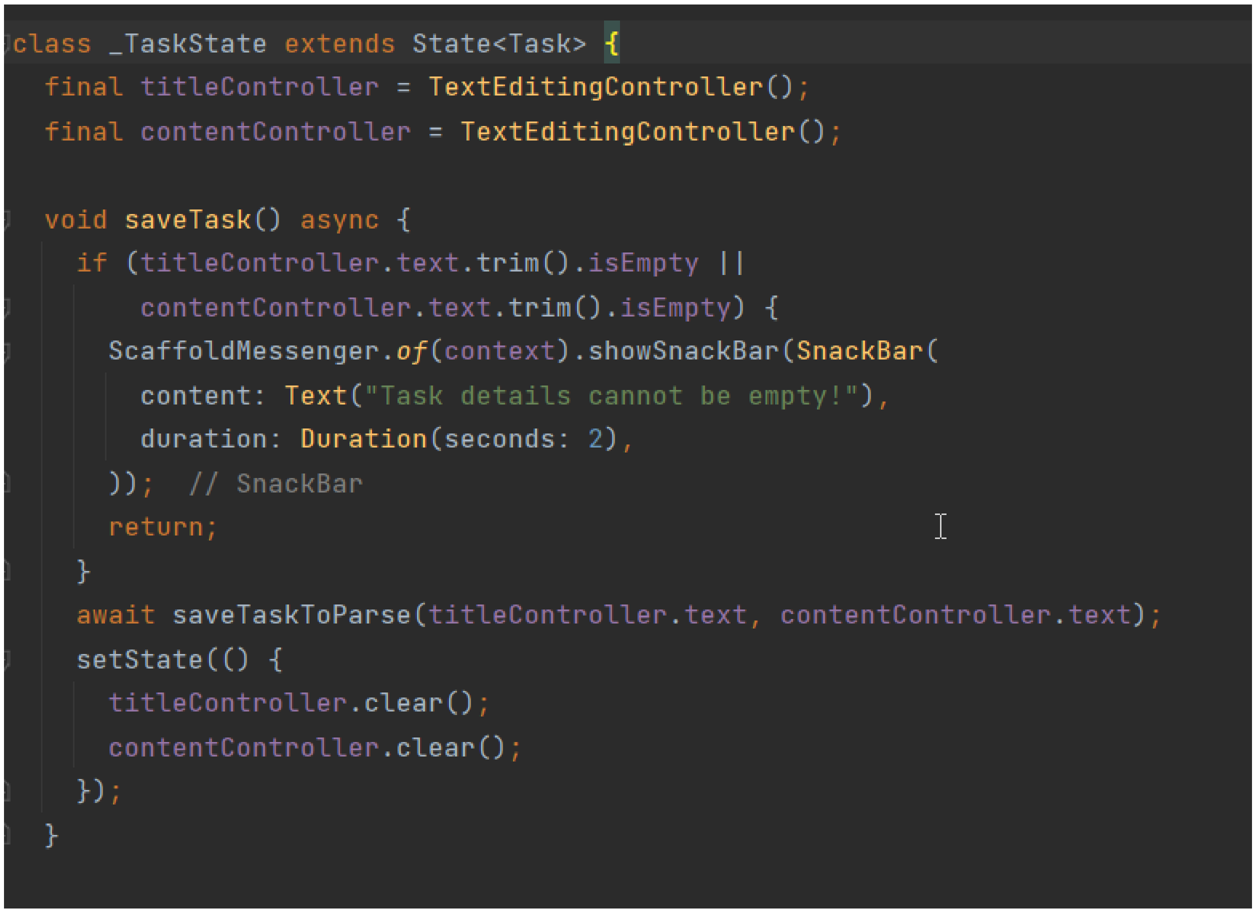
➢ 
```dart
class Task extends StatefulWidget {
  @override
  _TaskState createState() => _TaskState();
}
```

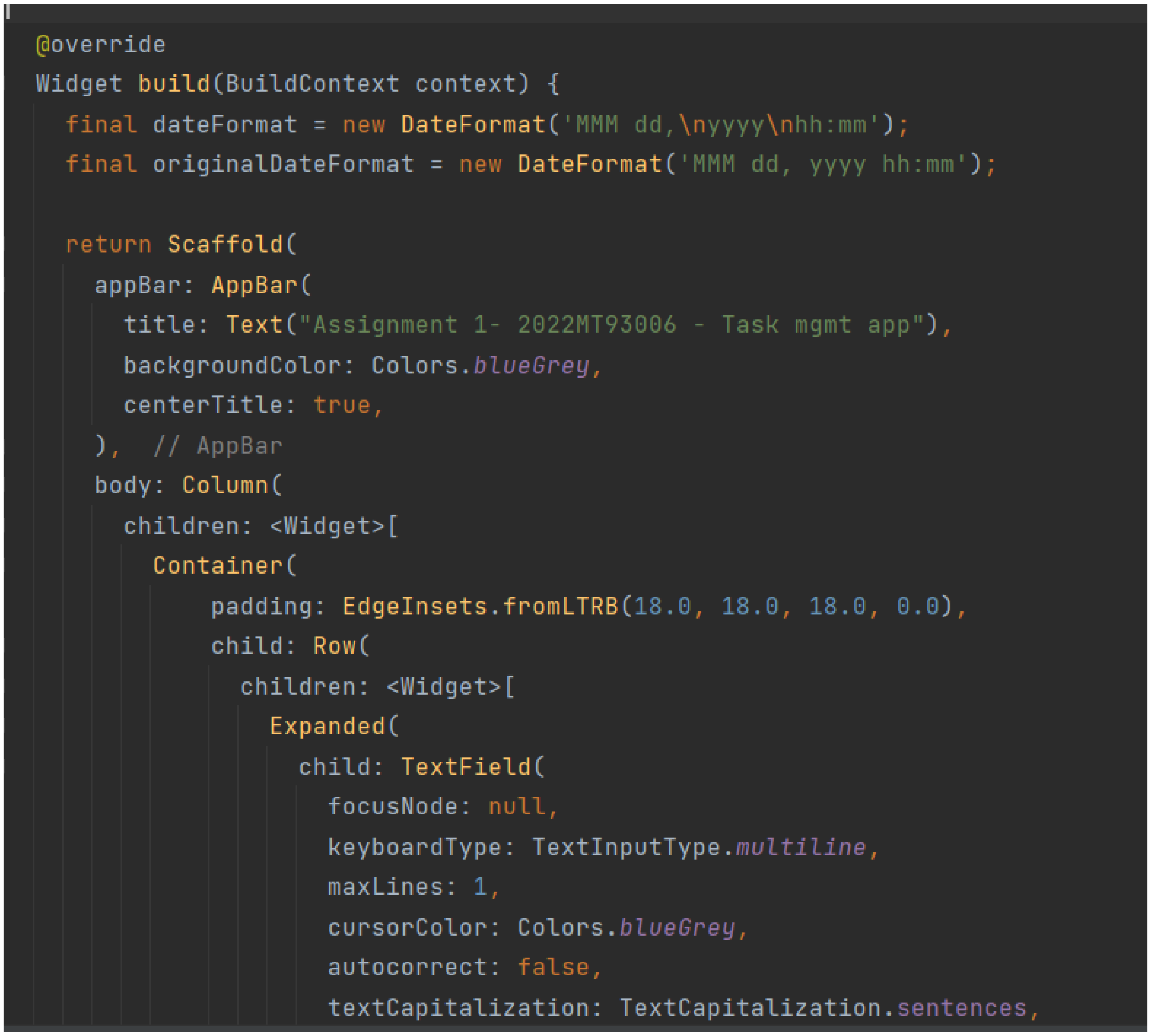
Figure 9: Task list screen



```
class _TaskState extends State<Task> {
  final titleController = TextEditingController();
  final contentController = TextEditingController();

  void saveTask() async {
    if (titleController.text.trim().isEmpty ||
        contentController.text.trim().isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text("Task details cannot be empty!"),
        duration: Duration(seconds: 2),
      ));  // SnackBar
      return;
    }
    await saveTaskToParse(titleController.text, contentController.text);
    setState(() {
      titleController.clear();
      contentController.clear();
    });
  }
```

➢

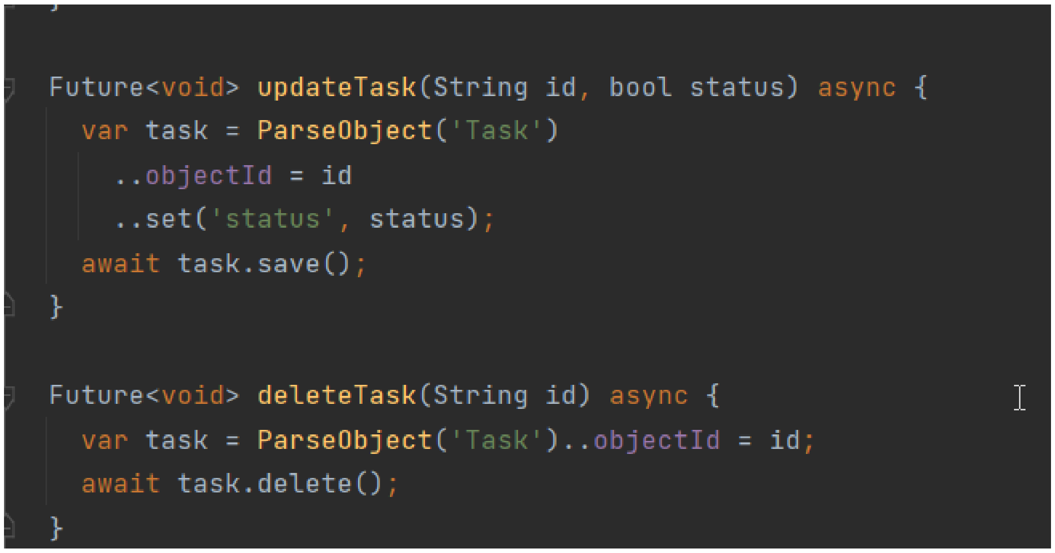
Figure 10: State for Task list screen



```
@override
Widget build(BuildContext context) {
  final dateFormat = new DateFormat('MMM dd,\nyyyy\nhh:mm');
  final originalDateFormat = new DateFormat('MMM dd, yyyy hh:mm');

  return Scaffold(
    appBar: AppBar(
      title: Text("Assignment 1- 2022MT93006 - Task mgmt app"),
      backgroundColor: Colors.blueGrey,
      centerTitle: true,
    ),  // AppBar
    body: Column(
      children: <Widget>[
        Container(
            padding: EdgeInsets.fromLTRB(18.0, 18.0, 18.0, 0.0),
            child: Row(
              children: <Widget>[
                Expanded(
                  child: TextField(
                    focusNode: null,
                    keyboardType: TextInputType.multiline,
                    maxLines: 1,
                    cursorColor: Colors.blueGrey,
                    autocorrect: false,
                    textCapitalization: TextCapitalization.sentences,
```

➢

Figure 11: Widget build methods

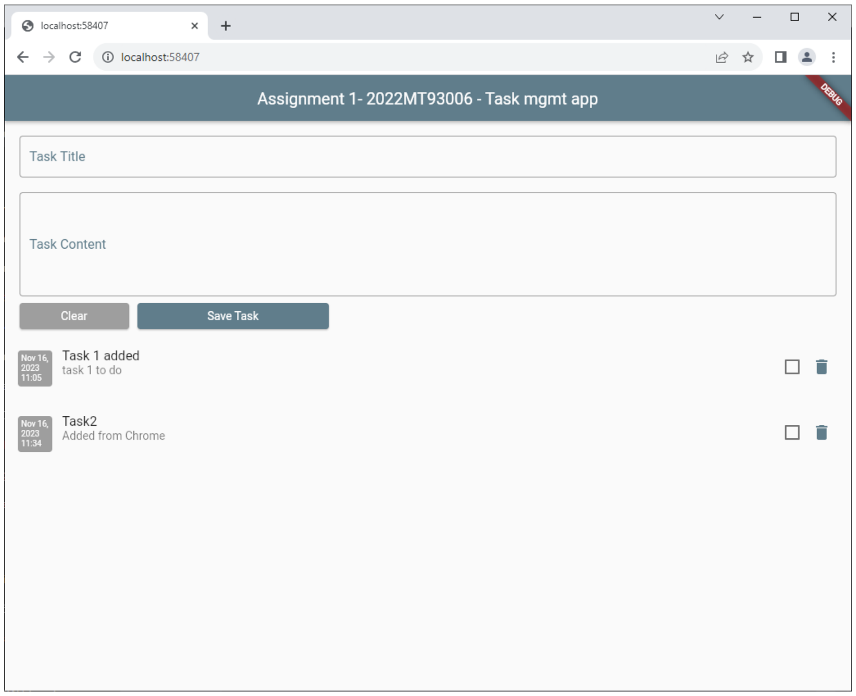


```
Future<void> updateTask(String id, bool status) async {
  var task = ParseObject('Task')
    ..objectId = id
    ..set('status', status);
  await task.save();
}

Future<void> deleteTask(String id) async {
  var task = ParseObject('Task')..objectId = id;
  await task.delete();
}
```

➢

Figure 12: Update and Delete methods,

➢ Let us run the application. To do so, create a flutter project and load the created flutter app to Android studio.
➢ Run the project by choosing the target
➢ Below is execution on chrome browser Figure 13
➢

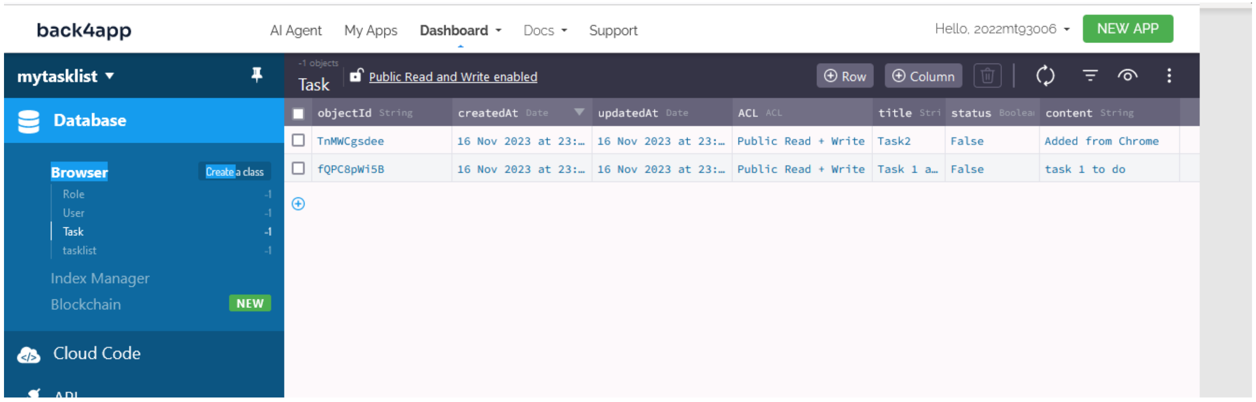


➢

Figure 13: Execution on Chrome Browser

- ➢ Below is execution on android emulation
- ➢ Task updated to the back4app is listed as shown in Figure 14

➢



objectId String	createdAt Date	updatedAt Date	ACL ACL	title Stri	status Boolean	content String
TnMWCgsdee	16 Nov 2023 at 23:…	16 Nov 2023 at 23:…	Public Read + Write	Task2	False	Added from Chrome
fQPC8pWi5B	16 Nov 2023 at 23:…	16 Nov 2023 at 23:…	Public Read + Write	Task 1 a…	False	task 1 to do

Figure 14: Updated task in Back4app

Encl:

- ➢ Project git hub: https://github.com/sivasankararumugham/Crossplatformdevelopment
- ➢ Video demo
    - ○ https://drive.google.com/file/d/1qaAY78o9hw2g_RN23mDcBTRpegeVzpdr/view?usp=sharing

Reference:

- ➢ https://github.com/gokulnathperiasamy/cpa_task_app