Lab session 2: UI Design and Creating Flutter Apps

Task-4 Use ChatGPT to create a Flutter Mobile App

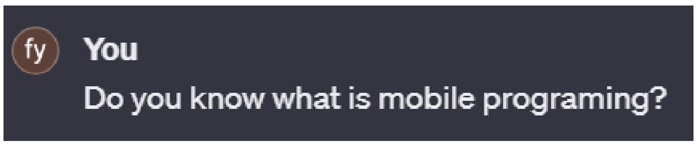
By leveraging ChatGPT, a powerful language model developed by OpenAI, developers can receive guidance, suggestions, and code samples to assist them in the app development process.

This documentation outlines how ChatGPT was used to generate instructions and code snippets for creating a To-Do List app.

Prompting ChatGPT

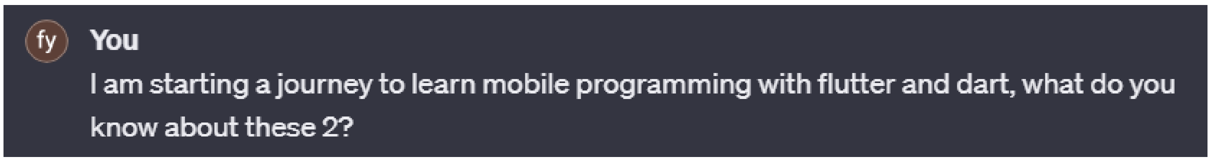
To utilize ChatGPT effectively, we began by providing it with context through prompt messages. These prompt messages were designed to gather information about mobile programming, Flutter, Dart, and the process of creating a first app. By providing this context, we aimed to ensure that ChatGPT's responses would be accurate and relevant to our specific needs.

Example prompt messages:



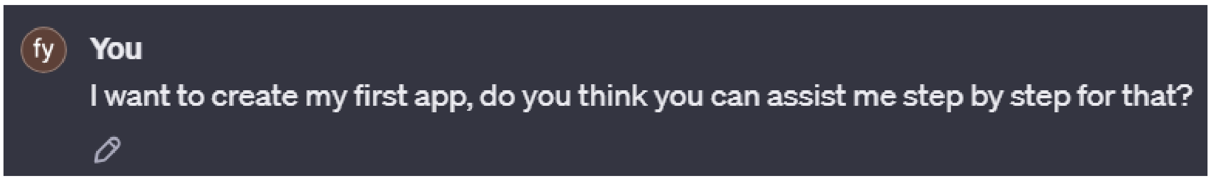
fy You
Do you know what is mobile programing?

- ➔ This prompt helps ChatGPT understand the significance of mobile programming and provides a broader context for the subsequent questions.



fy You
I am starting a journey to learn mobile programming with flutter and dart, what do you know about these 2?

- ➔ This prompt seeks information about Flutter and Dart.



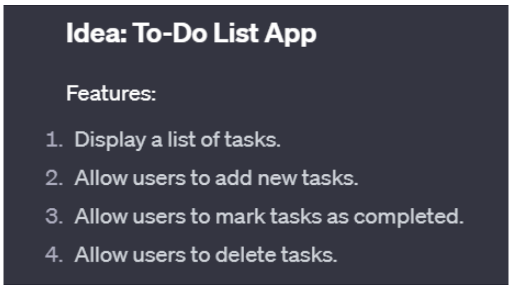
fy You
I want to create my first app, do you think you can assist me step by step for that?

- ➔ This prompt sets the stage for ChatGPT to provide instructions and guidance on the steps involved in creating a mobile app with Flutter.

By providing these prompts, we establish a foundation of knowledge and context for ChatGPT, enabling it to generate accurate and helpful responses tailored to mobile programming, Flutter, Dart, and the process of creating a flutter app.

Reviewing and Implementing Responses

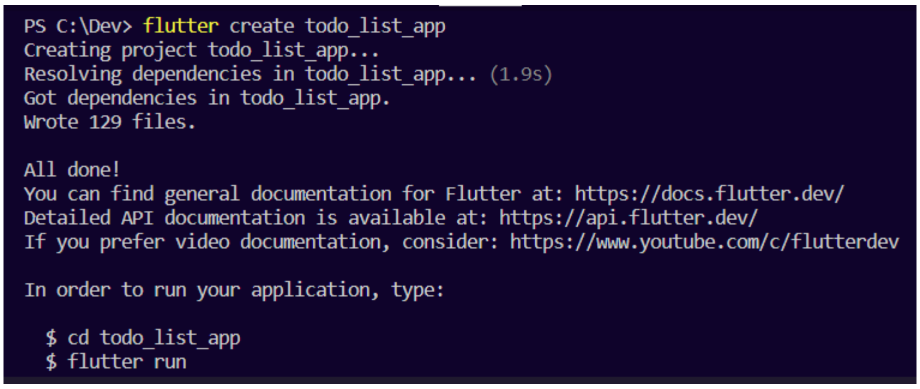
In the initial responses, ChatGPT suggested modifying the default app in Flutter as a first app. However, after further inquiry and insistence, ChatGPT suggested creating a simple to-do list app with the following features:


Idea: To-Do List App

Features:

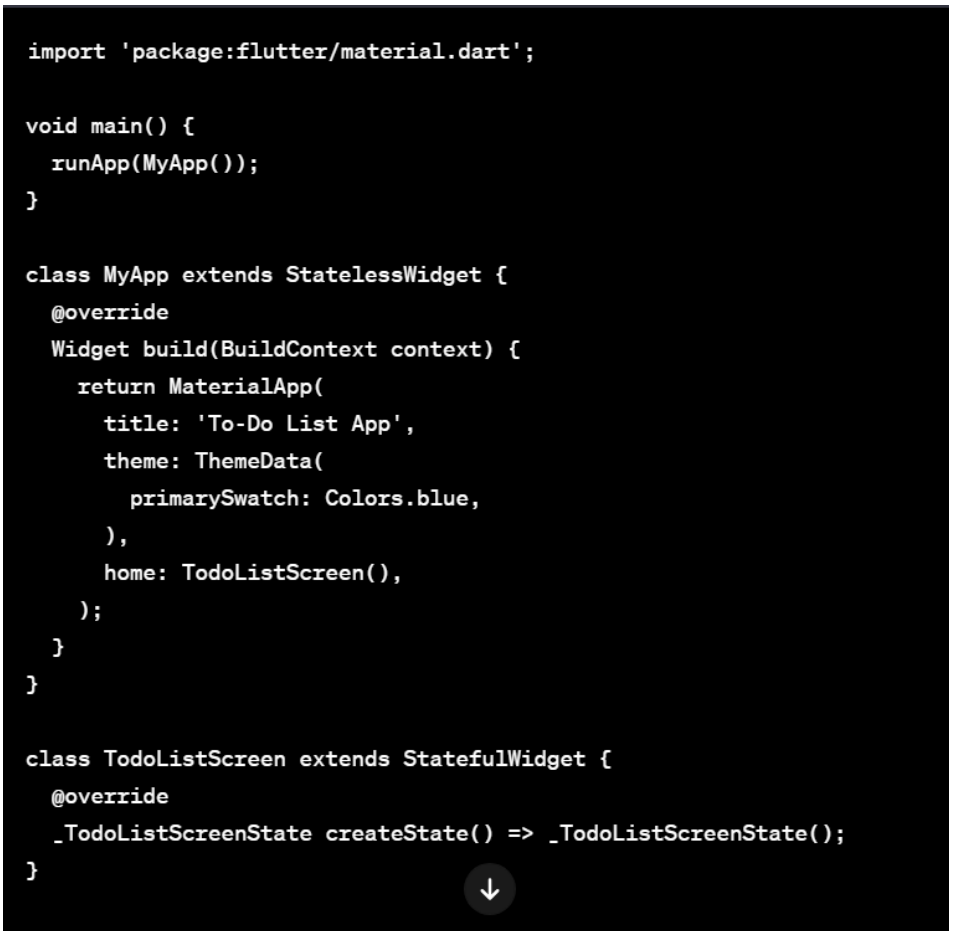
1. Display a list of tasks.
2. Allow users to add new tasks.
3. Allow users to mark tasks as completed.
4. Allow users to delete tasks.

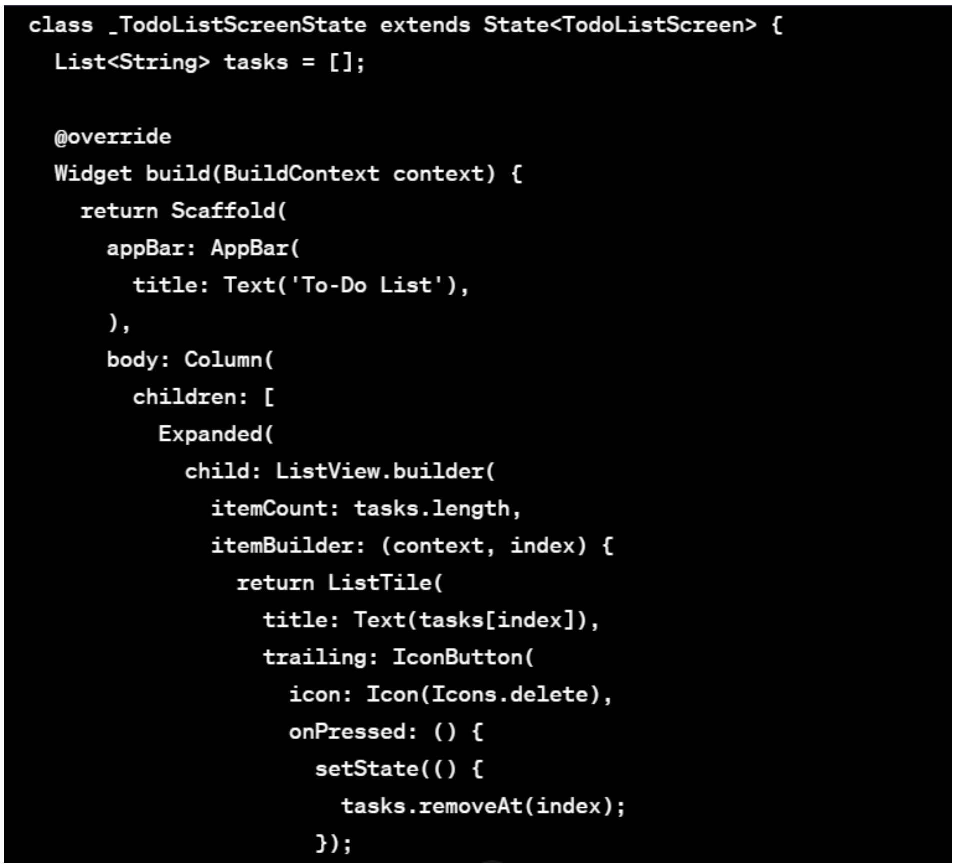
Step 1: Create a New Flutter Project


```
PS C:\Dev> flutter create todo_list_app
Creating project todo_list_app...
Resolving dependencies in todo_list_app... (1.9s)
Got dependencies in todo_list_app.
Wrote 129 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 todo_list_app
  $ flutter run
```

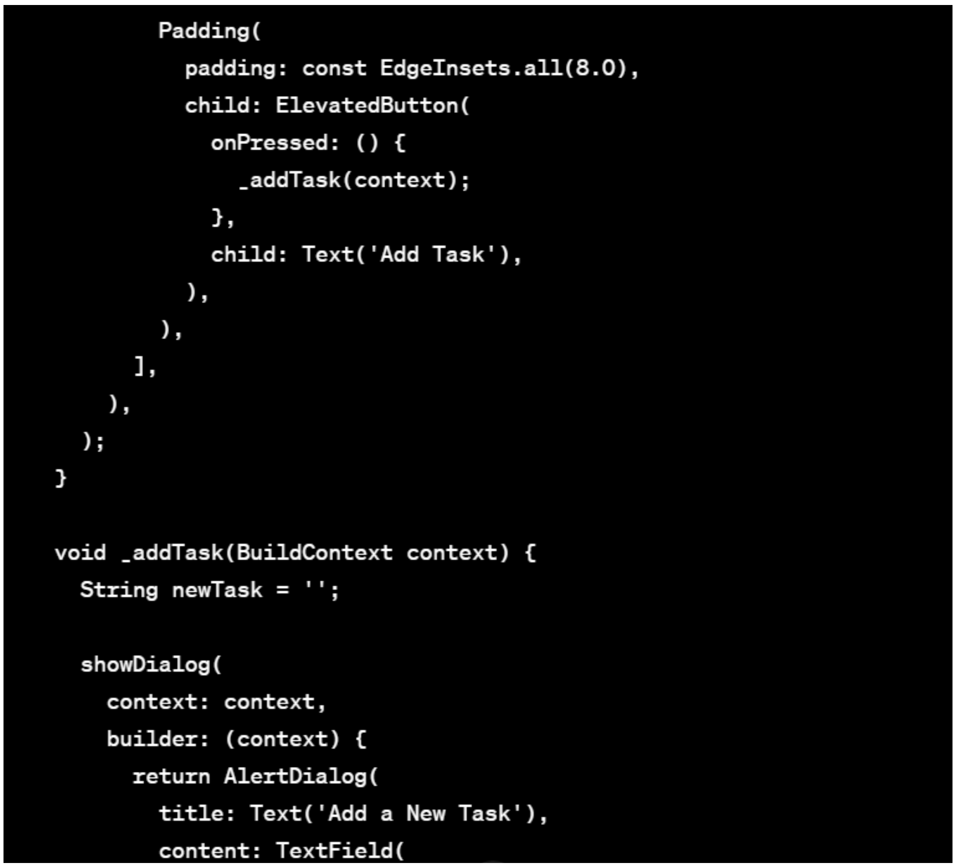
Step 2: Update lib/main.dart

ChatGPT provided the following code snippets to replace the content of the main.dart file in our Flutter project.

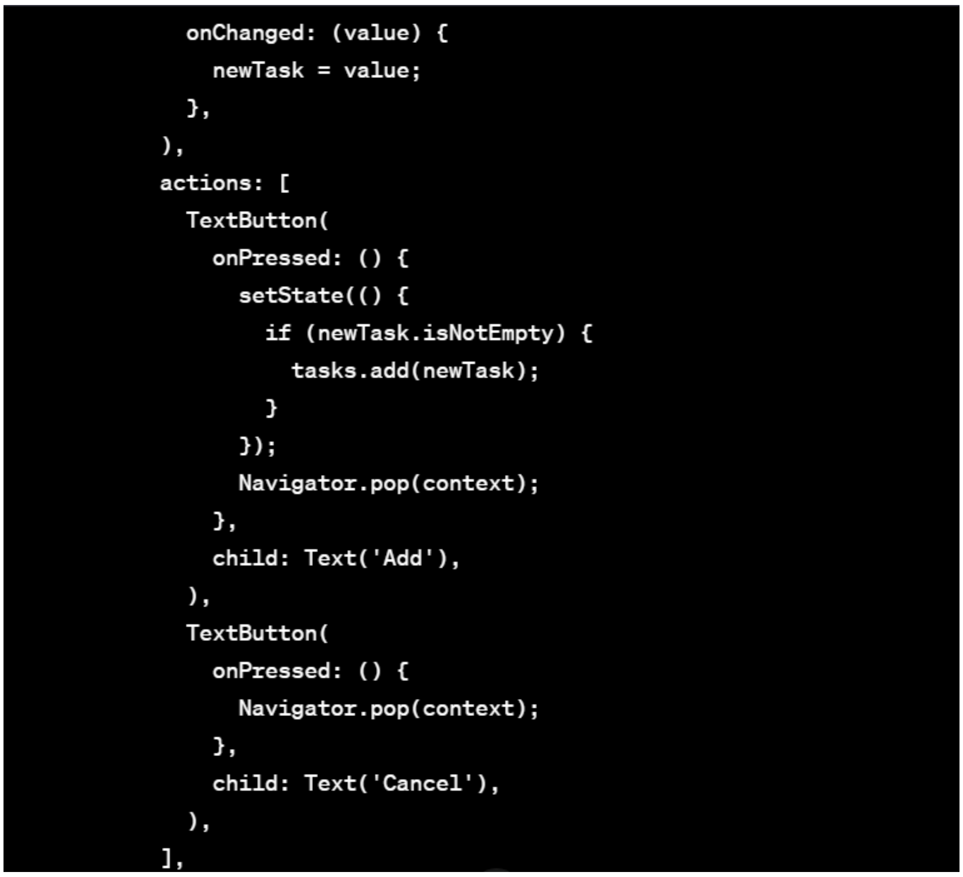

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'To-Do List App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: TodoListScreen(),
    );
  }
}

class TodoListScreen extends StatefulWidget {
  @override
  _TodoListScreenState createState() => _TodoListScreenState();
}
```



```
class _TodoListScreenState extends State<TodoListScreen> {
  List<String> tasks = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('To-Do List'),
      ),
      body: Column(
        children: [
          Expanded(
            child: ListView.builder(
              itemCount: tasks.length,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text(tasks[index]),
                  trailing: IconButton(
                    icon: Icon(Icons.delete),
                    onPressed: () {
                      setState(() {
                        tasks.removeAt(index);
                      });
```



```
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: ElevatedButton(
              onPressed: () {
                _addTask(context);
              },
              child: Text('Add Task'),
            ),
          ),
        ],
      ),
    );
  }

  void _addTask(BuildContext context) {
    String newTask = '';

    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: Text('Add a New Task'),
          content: TextField(
```

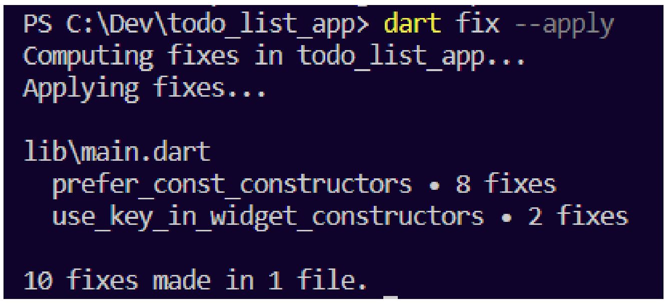


```
              onChanged: (value) {
                newTask = value;
              },
            ),
            actions: [
              TextButton(
                onPressed: () {
                  setState(() {
                    if (newTask.isNotEmpty) {
                      tasks.add(newTask);
                    }
                  });
                  Navigator.pop(context);
                },
                child: Text('Add'),
              ),
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: Text('Cancel'),
              ),
            ],
```

Step 3: Run the App

Before running the code, it's essential to fetch the dependencies required for the project. In this case, we can run flutter pub get in the terminal to fetch the necessary packages and ensure they are up to date.

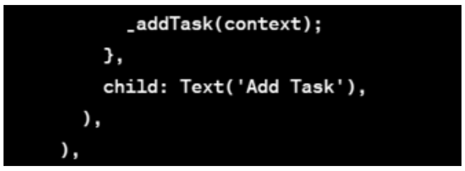
During the execution of ‘flutter pub get’, Visual Studio Code (VSC) prompted us to run ‘dart fix’ to address some issues in the code. ‘dart fix’ command automatically analyzes the code and suggests fixes for any detected issues.



```
PS C:\Dev\todo_list_app> dart fix --apply
Computing fixes in todo_list_app...
Applying fixes...

lib\main.dart
  prefer_const_constructors • 8 fixes
  use_key_in_widget_constructors • 2 fixes

10 fixes made in 1 file.
```

By running dart fix, we can ensure that the code is in line with the latest best practices and conventions.

Before dart fix:



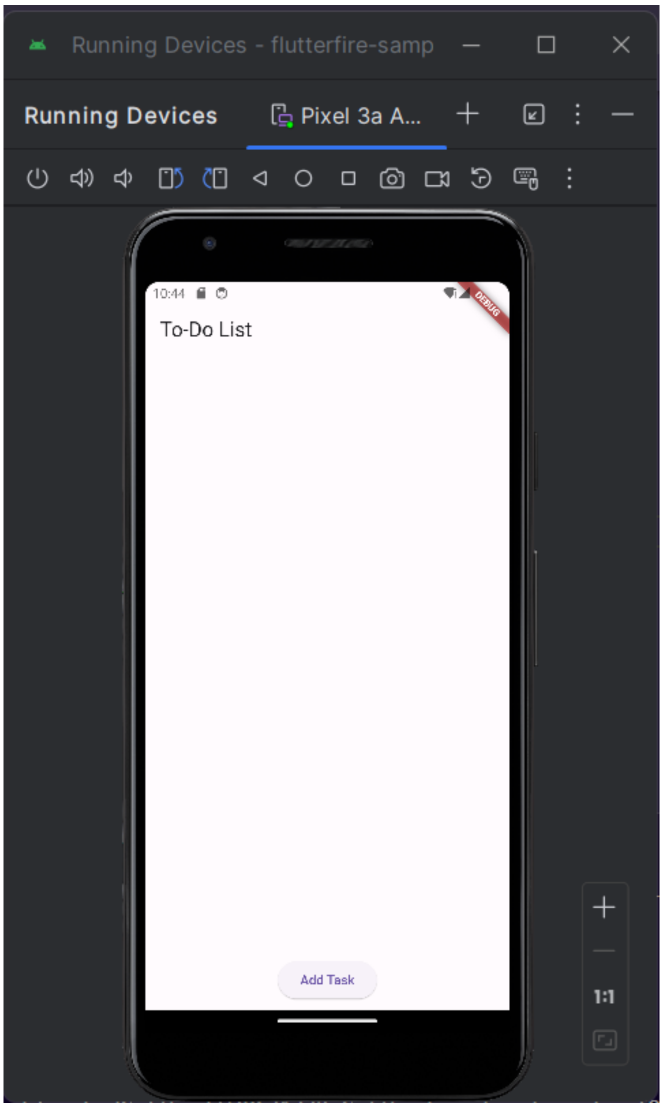
```
          _addTask(context);
        },
        child: Text('Add Task'),
      ),
    ),
```

After dart fix:

```
child: const Text('Add Task'),
```

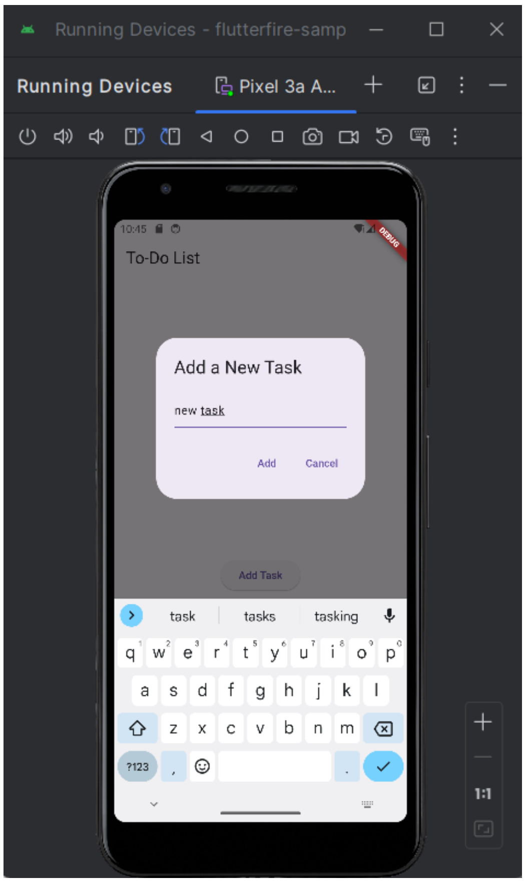
It seems that the dart fix command automatically added the const keyword before each parameter in the code. This behavior is a result of the Dart language's effort to promote the use of immutable objects and improve performance by utilizing compile-time constants.

Once the dependencies have been fetched and any suggested fixes have been applied, we can proceed to run the code by executing the flutter run command in the terminal. This command launches the Flutter app on the connected device or emulator, allowing us to test and interact with the to-do list app.

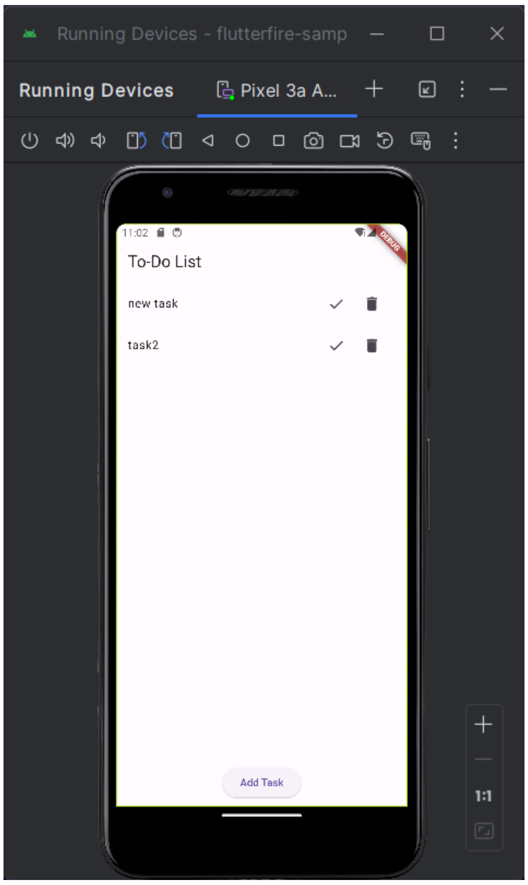


During the execution of the app, we can verify that the functionality aligns with the suggested features and expectations outlined by ChatGPT. We can add tasks, mark them as completed, and delete them from the list, ensuring that the app behaves as intended.

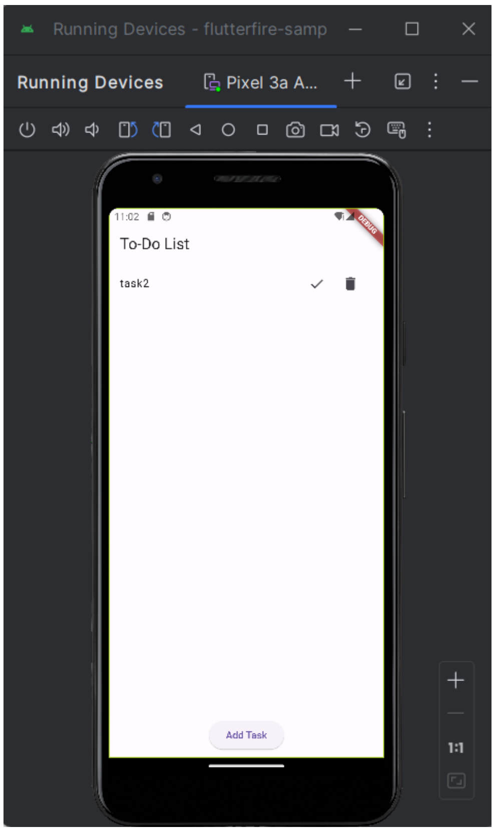
1. Add task



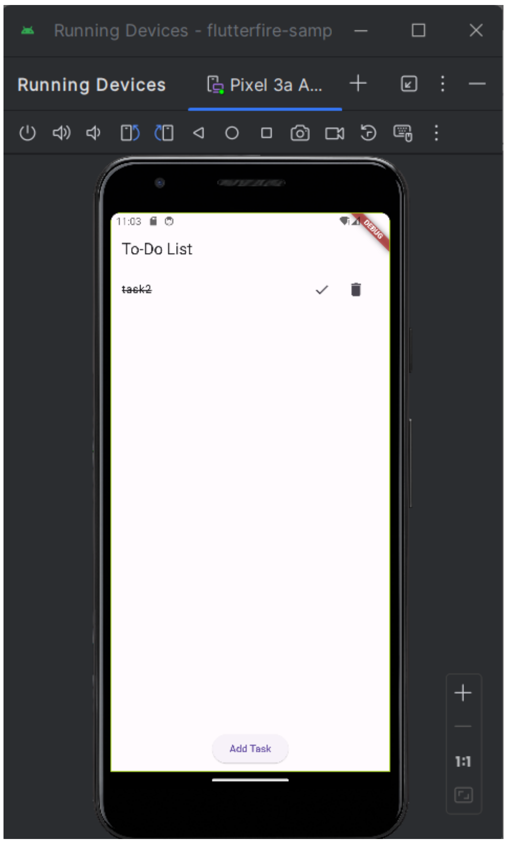
2. Display a list of task



3. Delete task



4. Mark as completed



Conclusion

In conclusion, with the help of ChatGPT, we successfully created a to-do list app by following the provided guidance and code snippets. While the design could be further improved, the core functionality and logic have been correctly implemented. ChatGPT proved to be a valuable resource for information and code generation, accelerating the development process. However, it's important to review and validate the generated code for correctness and adherence to project requirements.

Reference

You can find the chat here:

https://chat.openai.com/share/46aec7a6-be99-4466-874c-f3d074efb33e