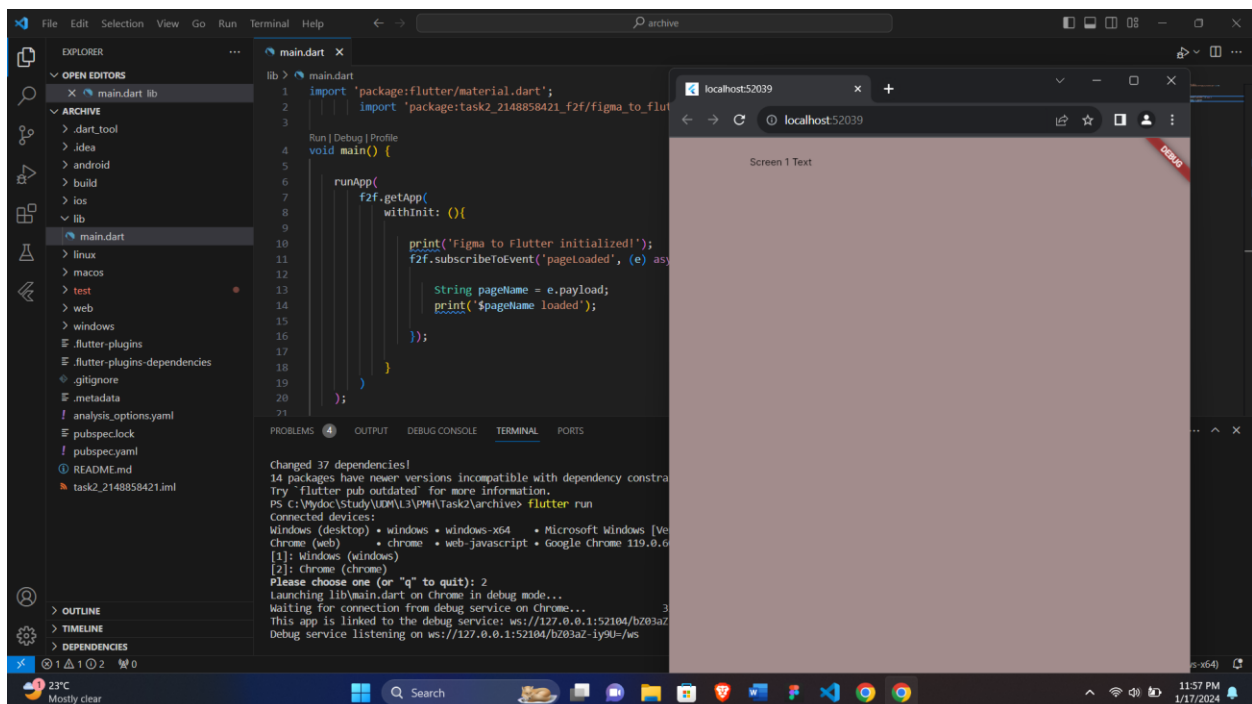


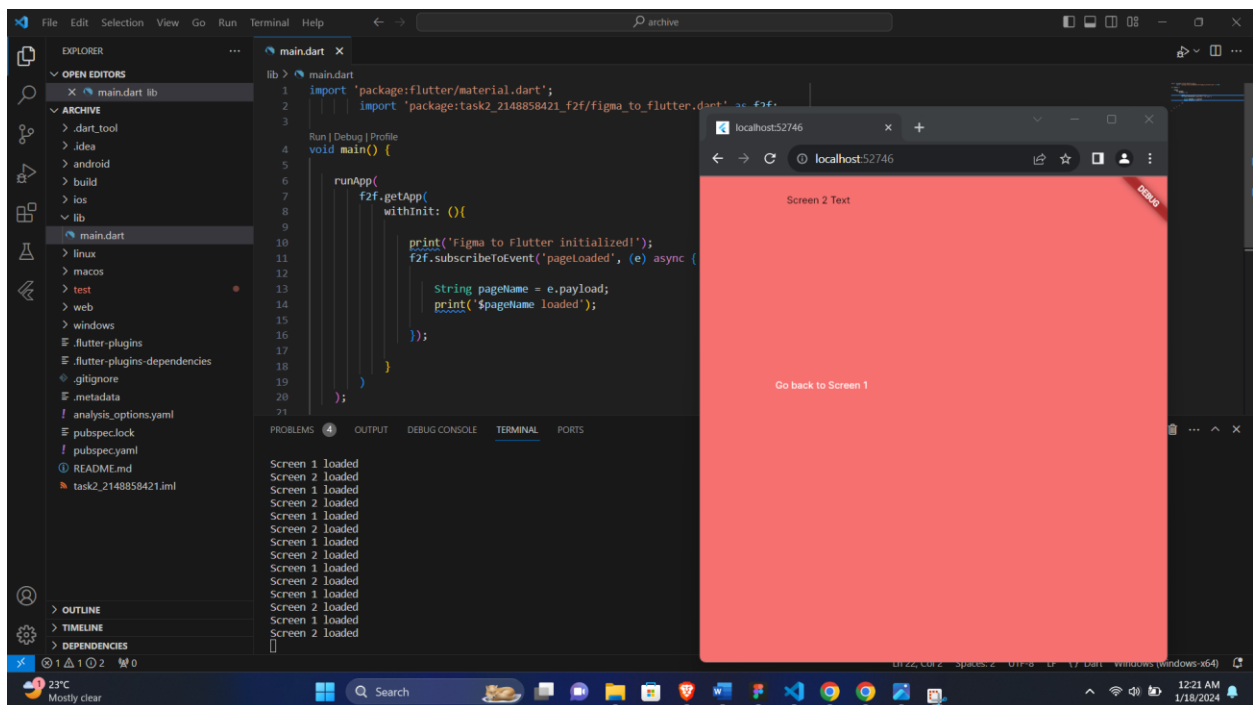
RAKOTONDRAMANANA A.A. LAFATRA

INFO 3 – UNIVERSITE DES MASCAREIGNES

## TP2 session 2: UI Design and Creating Flutter Apps

### Task-1: Figma2Flutter





## Task-2: Take your Flutter app from boring to beautiful

1- Setting up a Flutter Environment To participate in this workshop, you need two essential software components: the Flutter SDK and an editor.

You have the option to run the programming workshop on one of the following devices:

- A physical Android or iOS device connected to your computer and configured in developer mode.
- The iOS simulator (requires the installation of Xcode tools).
- The Android emulator (must be configured in Android Studio).
- A web browser (Chrome is required for debugging).

It is also possible to develop your application as a desktop application on Windows, Linux, or macOS operating systems. It is crucial to develop your application on the platform where you plan to deploy it. For instance, if you intend to create a Windows desktop application, you must do so under Windows to access the appropriate compilation chain. Additionally, make sure to review the specific operating system requirements, detailed on [flutter.dev/desktop](https://flutter.dev/desktop).

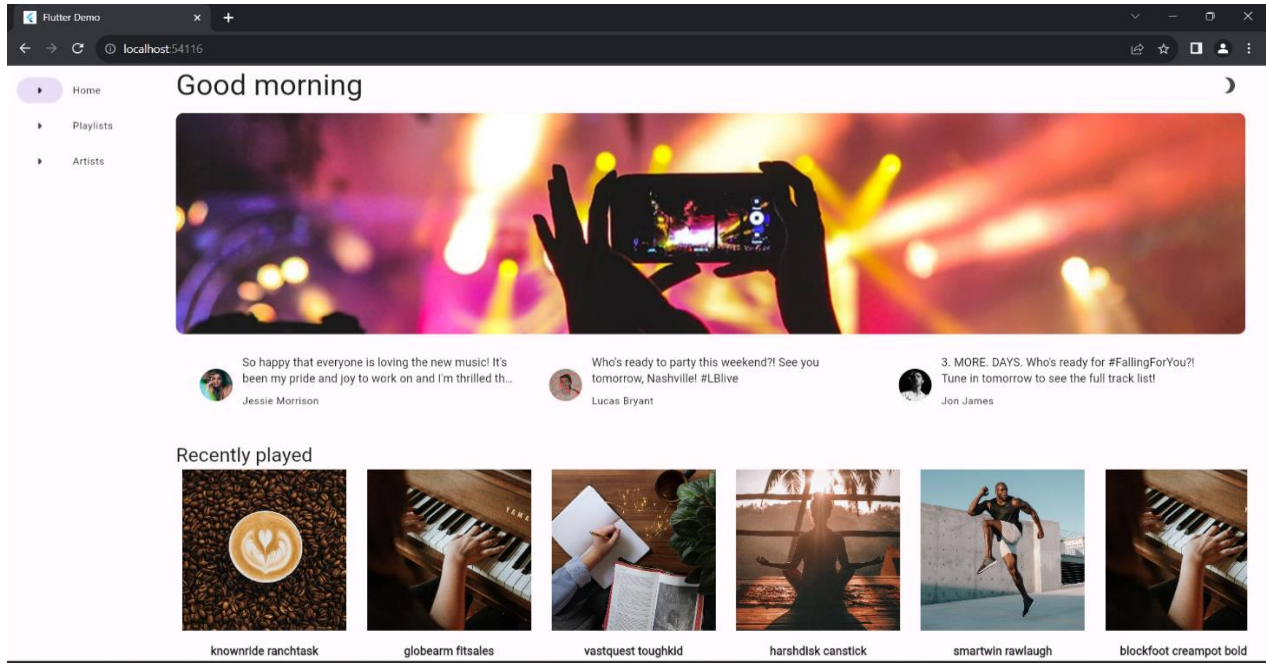
2- Obtaining the Workshop's Starter Application Clone the application from GitHub To obtain a copy of this programming workshop from GitHub, please execute the following commands:

**git clone** <https://github.com/flutter/codelabs.git>

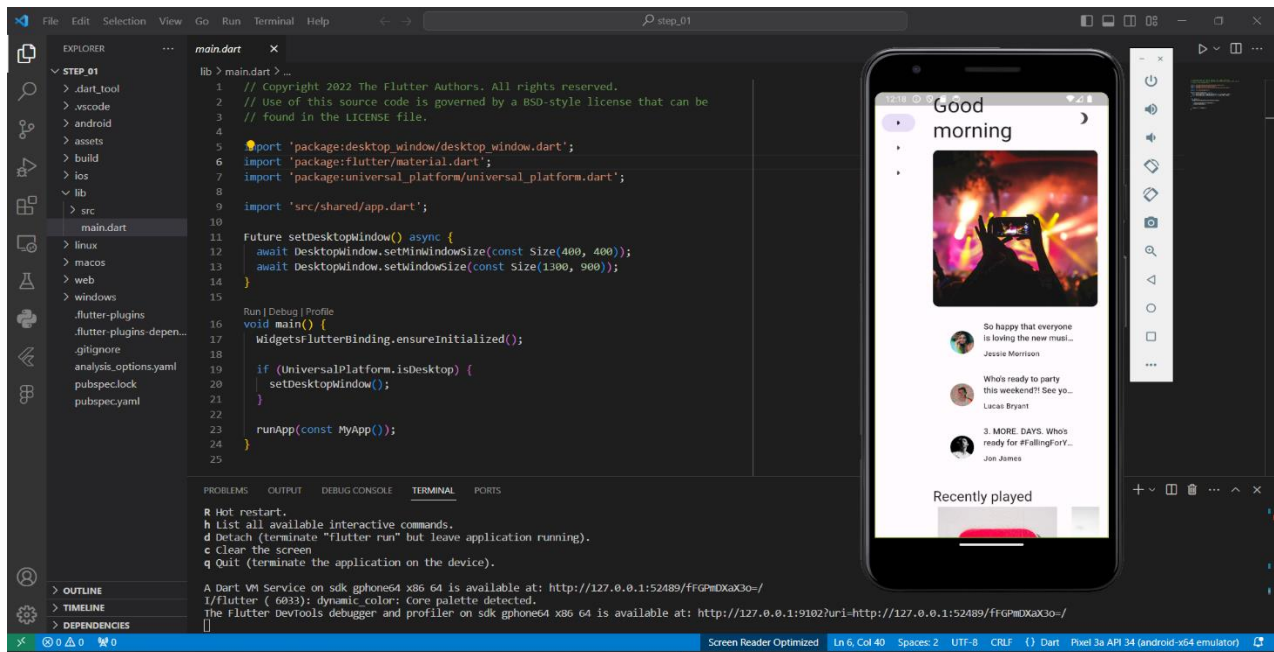
To make sure everything is working, run the Flutter application as a desktop application as shown below. Alternatively, open this project in your IDE, and use its tooling to run the application.

Launch the application. The initial code for MyArtist's home screen is now active, and you should observe the MyArtist home screen. While it appears satisfactory on desktop, the mobile view needs improvement. One noticeable issue is that it does not accommodate the notch.

- Chrome

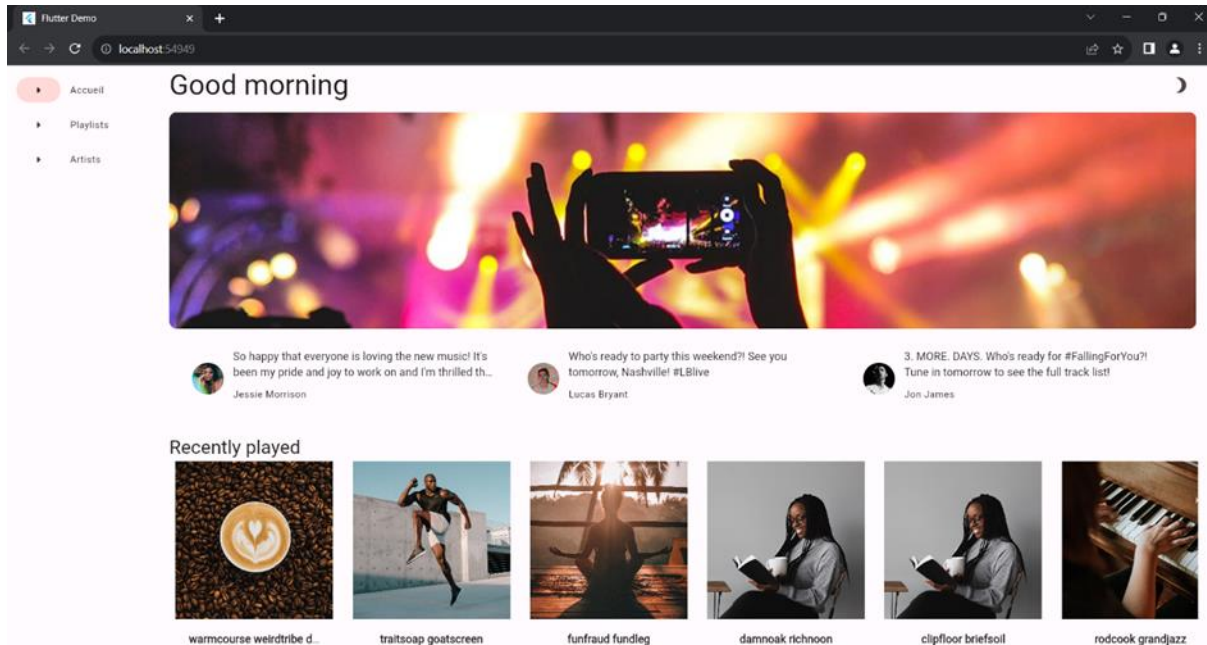


- Mobile



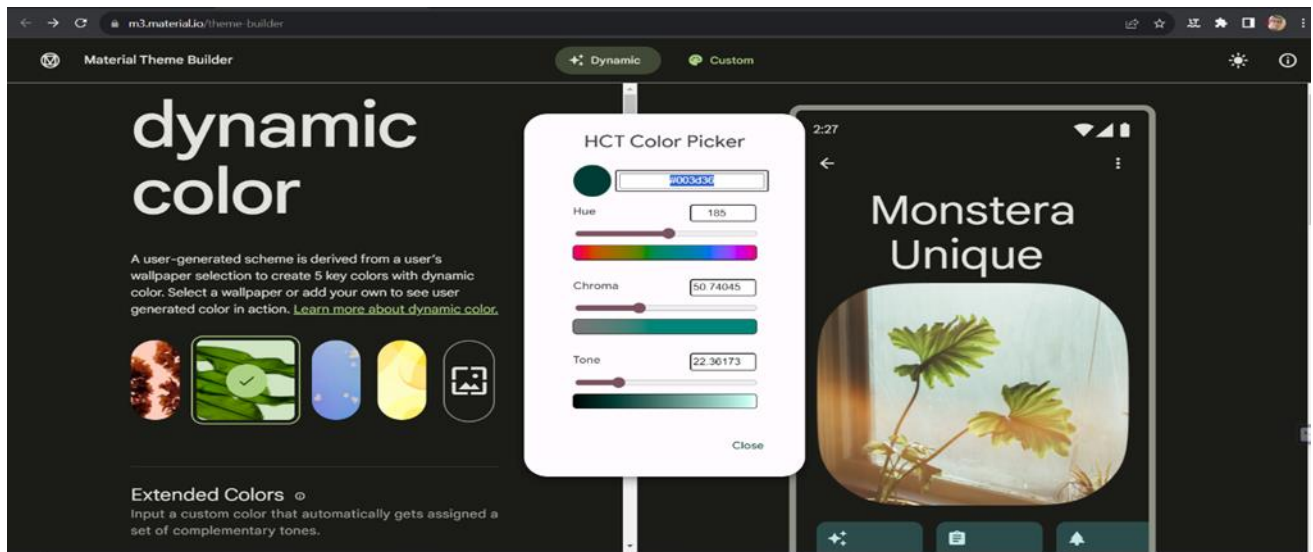
### 3- Leveraging Typography

- To incorporate unique launch icons for each navigation destination, we will make modifications to the code within "lib/src/shared/router.dart." As an example, I changed "Home" to "Accueil" to test the functionality.

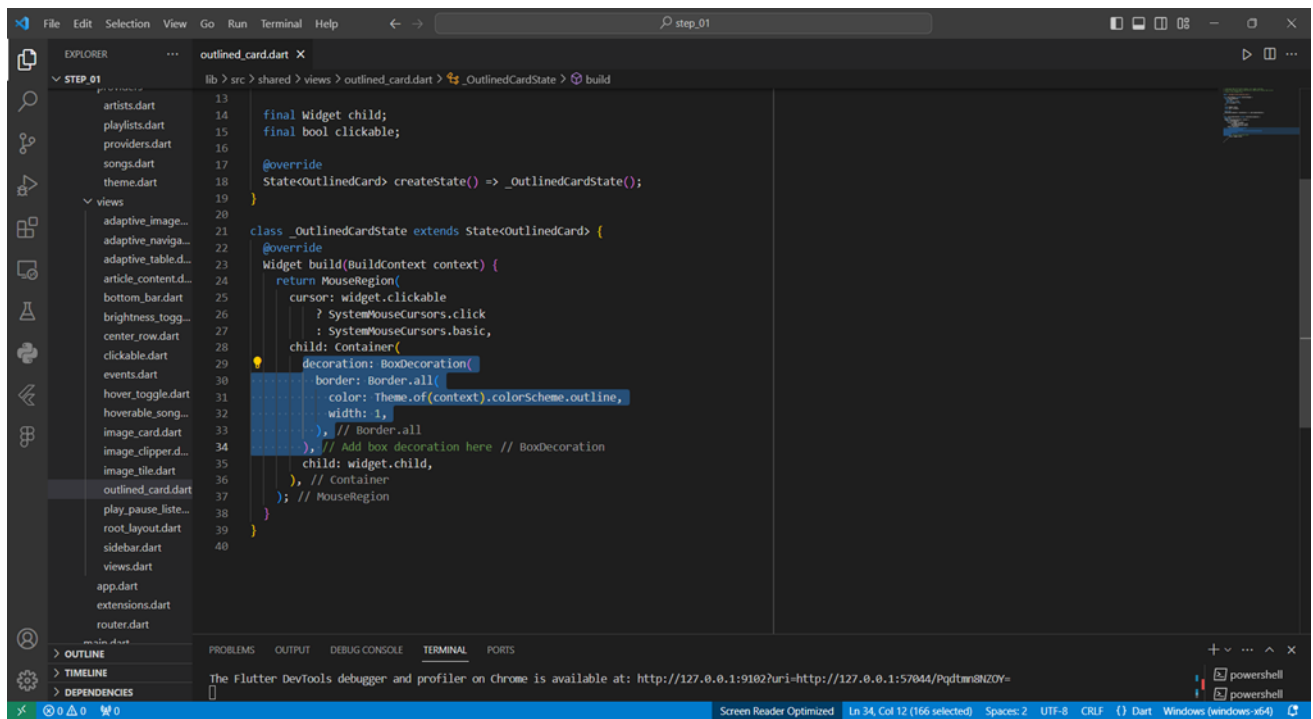


### 4- Defining the Theme

- To establish coherent widgets and colors associated with a theme throughout the application, navigate to the path: "lib/src/shared/providers/theme.dart."
- To utilize the provider, create an instance and pass it to the theme-defined object in the "lib/src/shared/app.dart" file, within the MaterialApp class. All nested theme objects will inherit from this instance. By adding "final theme = ThemeProvider.of(context);" and "theme: theme.light(settings.value.sourceColor)," in the code.
- Visit the Material Theme Builder website to set the color using the provided dialogue box, then copy the hexadecimal code.



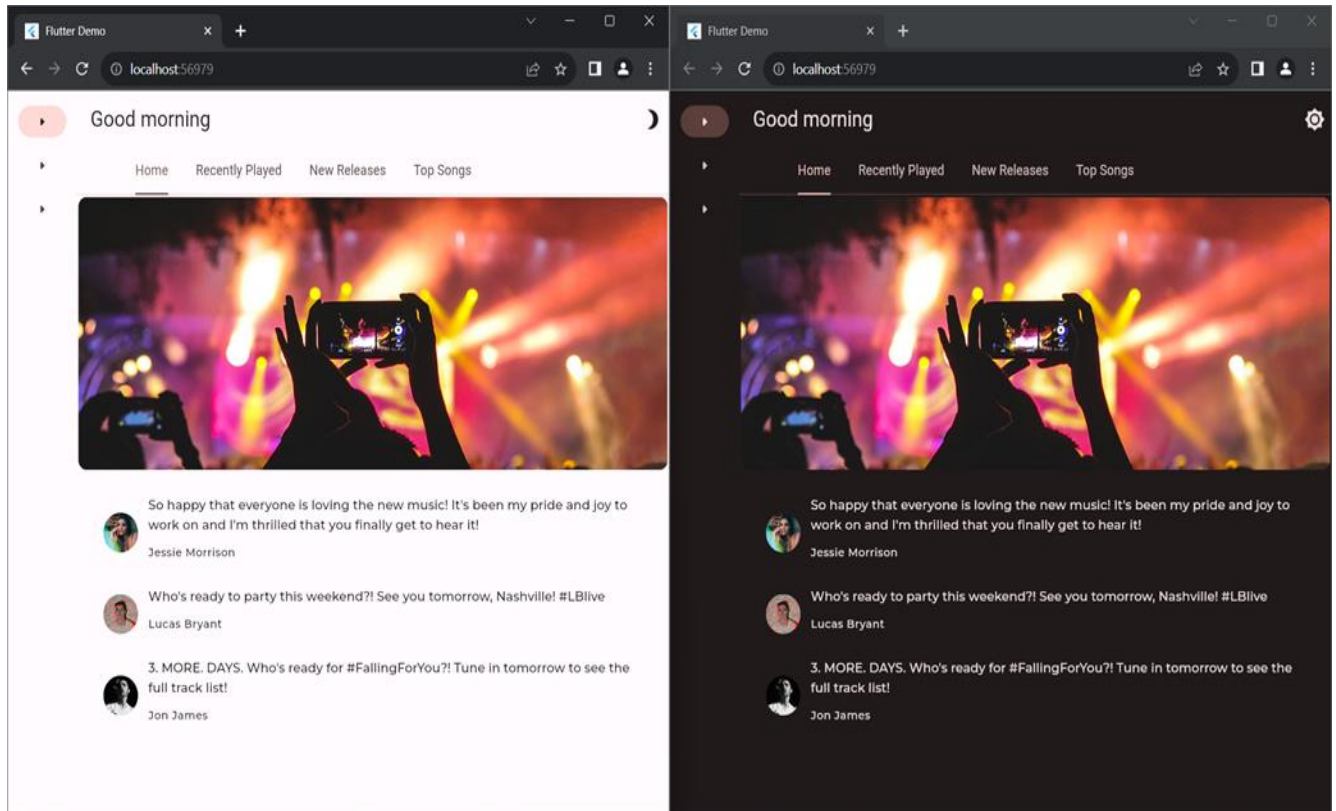
- To utilize a specific color, navigate to the path "lib/src/shared/views/outlined\_card.dart," and add the following code.



- To adjust the application's brightness in the device's system settings, navigate to the path "lib/src/shared/app.dart," and add the following code: "darkTheme:



`theme.dark(settings.value.sourceColor),"` and `"themeMode: theme.themeMode(),"` within the `"return MaterialApp.router()"` line.



### Task-3: Flutter Authentication

Fingerprint authentication has become a crucial necessity in our mobile applications, representing the latest and most reliable innovation in identity and access management. This biometric technology allows users to access information or services through fingerprint images, verifying their identity based on a stored fingerprint template.

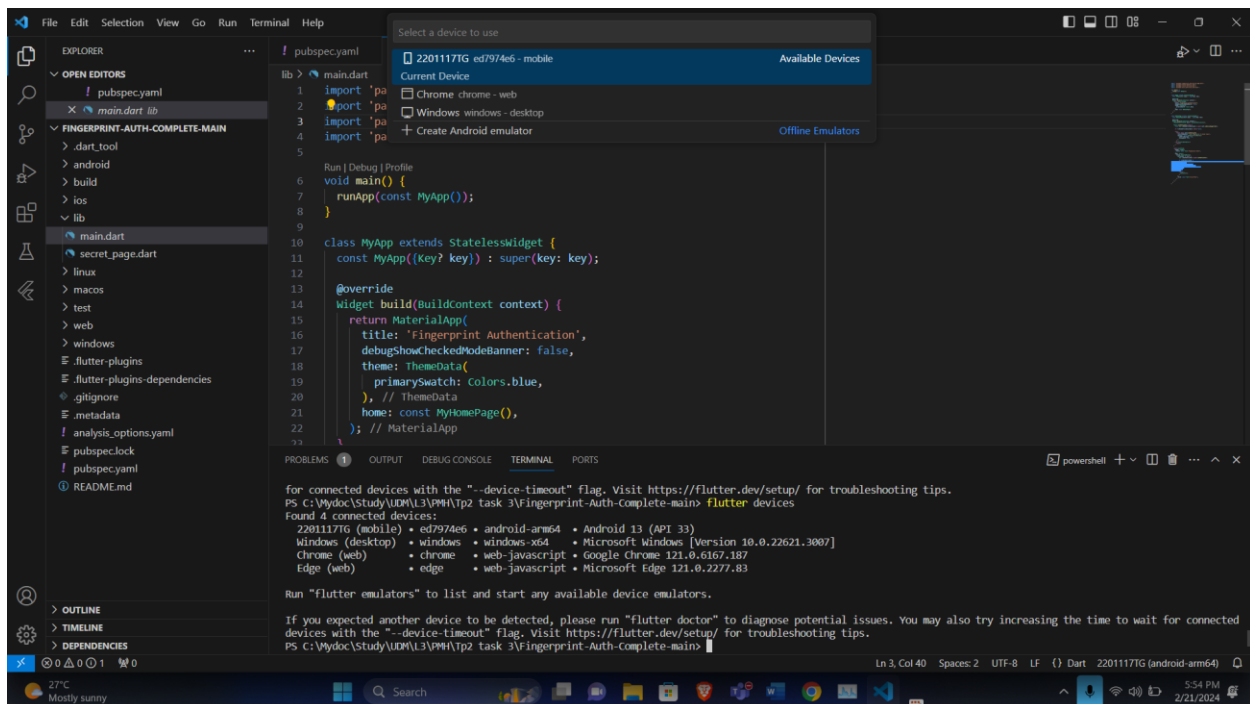
or implementing this functionality in a Flutter application, the `local_auth` package is available. This Flutter plugin facilitates on-device authentication of the user. To set up the Flutter project,

clone the starter files provided for this project, and execute the commands **flutter create .** and **flutter pub add local\_auth** to add the necessary dependencies.

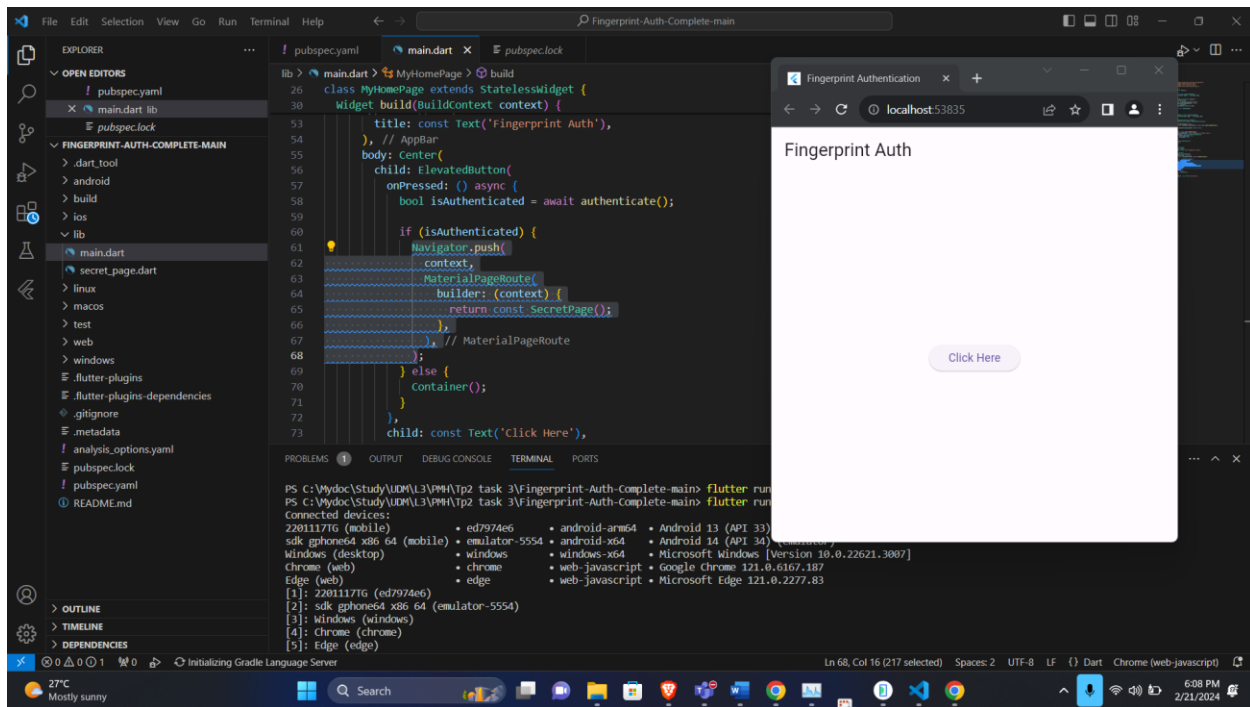
Once the configuration is complete, proceed to add functionalities. Import the `local_auth` packages and follow these steps:

1. Check if the device supports biometrics; otherwise, default to device credentials.
2. Authenticate the user with the available biometrics using the **authenticate()** method, allowing for device authentication methods like pin, pattern, and passcode.
3. The **useErrorDialogs** parameter determines whether the system handles user-fixable issues encountered during authentication.
4. The **stickyAuth** parameter, when set to true, ensures that authentication resumes when the application is resumed after going into the background during the authentication process.

If the authentication returns true, the user is directed to the `SecretPage` screen. To test the application, run the command **flutter run** in your terminal.







## Task-4 : Use ChatGPT to create a Flutter Mobile App.

### Step 1: Set Up Flutter

Ensure Flutter is installed on your machine by following the instructions on the official Flutter website: [Flutter Installation Guide](#).

### Step 2: Create a New Flutter Project

View → command Palette → Flutter New Project

### Step 3: Open the Project in an IDE

Use your preferred IDE, such as Visual Studio Code, to open the **todo\_app** folder.

### Step 4: Define the App Structure

In the **lib/main.dart** file, define the basic app structure. This includes creating a **MyApp** class and setting up a **MaterialApp** with a title and the initial screen, **TodoListScreen**.

### Step 5: Create the Todo List Screen

Create a new file, `screens/todo_list_screen.dart`, which contains the code for the app's main screen. This screen includes a simple app bar and the **ToDoList** widget.

## Step 6: Create the Todo List Widget

Inside a new folder named **widgets**, create a file named **todo\_list.dart**. This file contains the **ToDoList** widget, a stateful widget that manages a list of tasks. It also includes functionality for adding and deleting tasks.

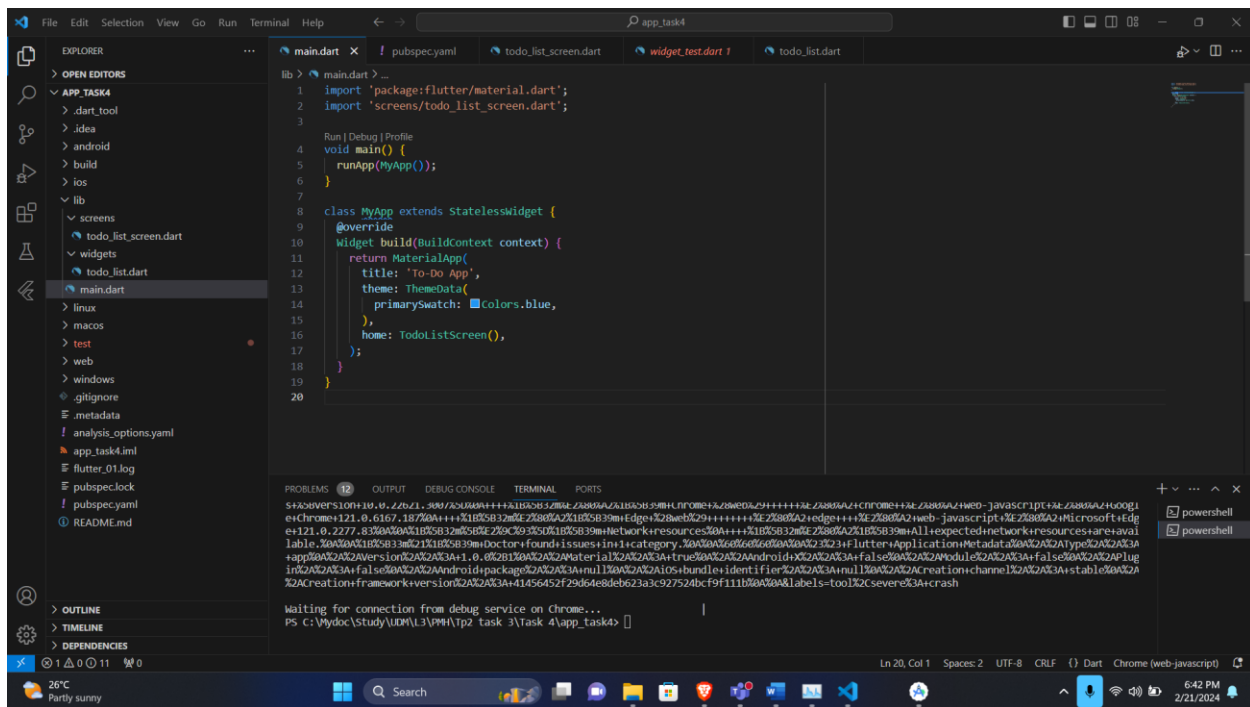
## Step 7: Run the App

Save your files and run the app using the command:

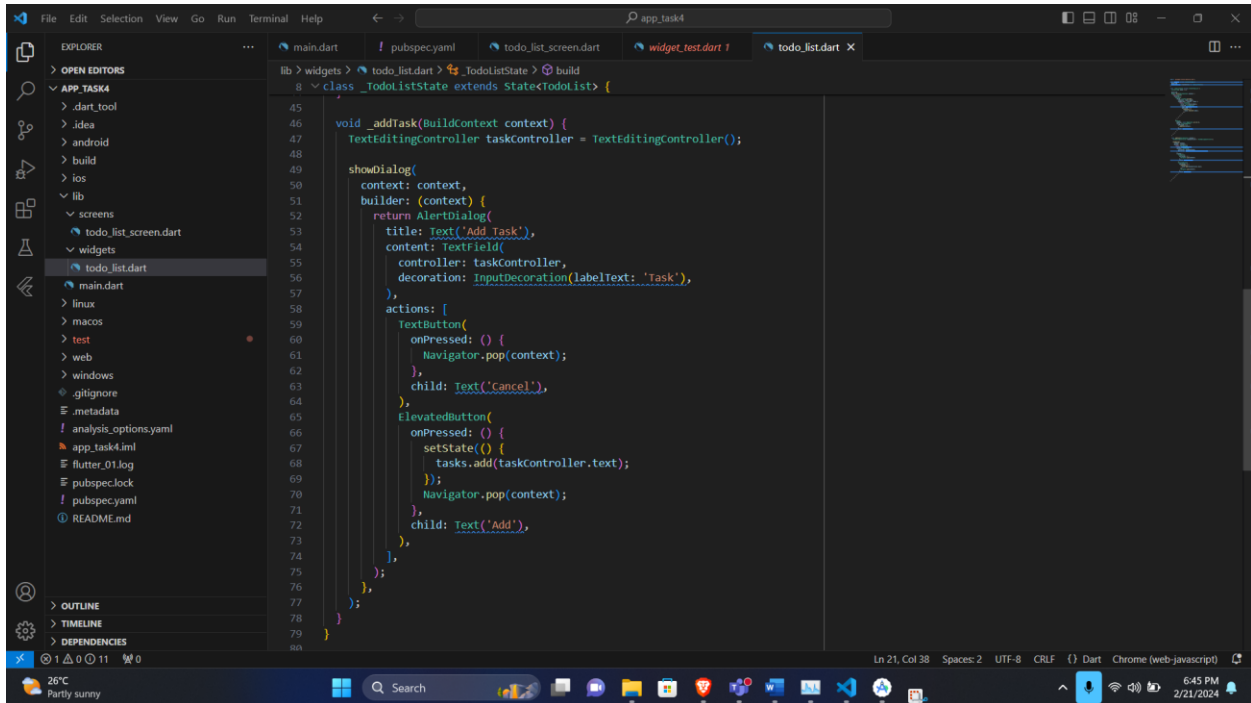
flutter run

### Screenshots:

### App Main Screen:

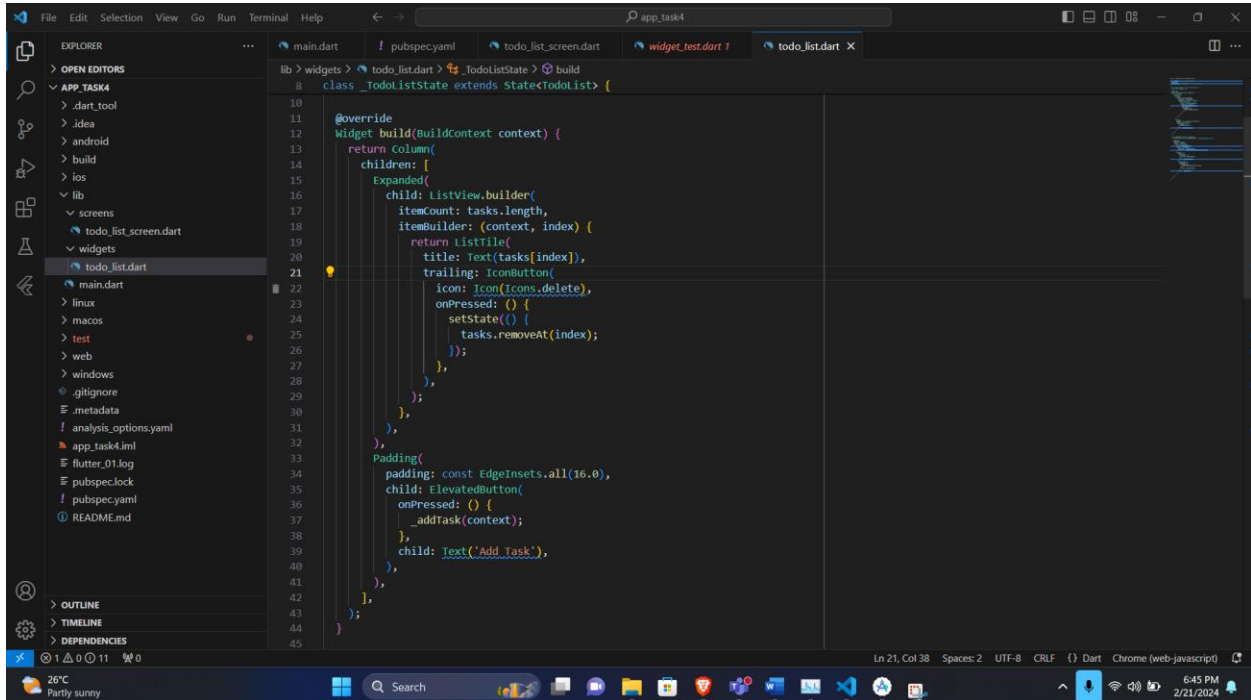


## Adding a Task:



```
lib > widgets > todo_list.dart > _TodoListState > build
8 class _TodoListState extends State<TodoList> {
45
46
47 void _addTask(BuildContext context) {
48   TextEditingController taskController = TextEditingController();
49
50   showDialog(
51     context: context,
52     builder: (context) {
53       return AlertDialog(
54         title: Text('Add Task'),
55         content: TextField(
56           controller: taskController,
57           decoration: InputDecoration(labelText: 'Task'),
58         ),
59         actions: [
60           TextButton(
61             onPressed: () {
62               Navigator.pop(context);
63             },
64             child: Text('Cancel'),
65           ),
66           ElevatedButton(
67             onPressed: () {
68               setState(() {
69                 tasks.add(taskController.text);
70               });
71               Navigator.pop(context);
72             },
73             child: Text('Add'),
74           ),
75         ],
76       );
77     },
78   );
79 }
80 }
```

## Deleting a Task:



```
lib > widgets > todo_list.dart > _TodoListState > build
8 class _TodoListState extends State<TodoList> {
10
11 @override
12 Widget build(BuildContext context) {
13   return Column(
14     children: [
15       Expanded(
16         child: ListView.builder(
17           itemCount: tasks.length,
18           itemBuilder: (context, index) {
19             return ListTile(
20               title: Text(tasks[index]),
21               trailing: IconButton(
22                 icon: Icon(Icons.delete),
23                 onPressed: () {
24                   setState(() {
25                     tasks.removeAt(index);
26                   });
27                 },
28               ),
29             ),
30           ],
31         ),
32       ),
33       Padding(
34         padding: const EdgeInsets.all(16.0),
35         child: ElevatedButton(
36           onPressed: () {
37             _addTask(context);
38           },
39           child: Text('Add Task'),
40         ),
41       ),
42     ],
43   );
44 }
45 }
```

## Demonstration:

