

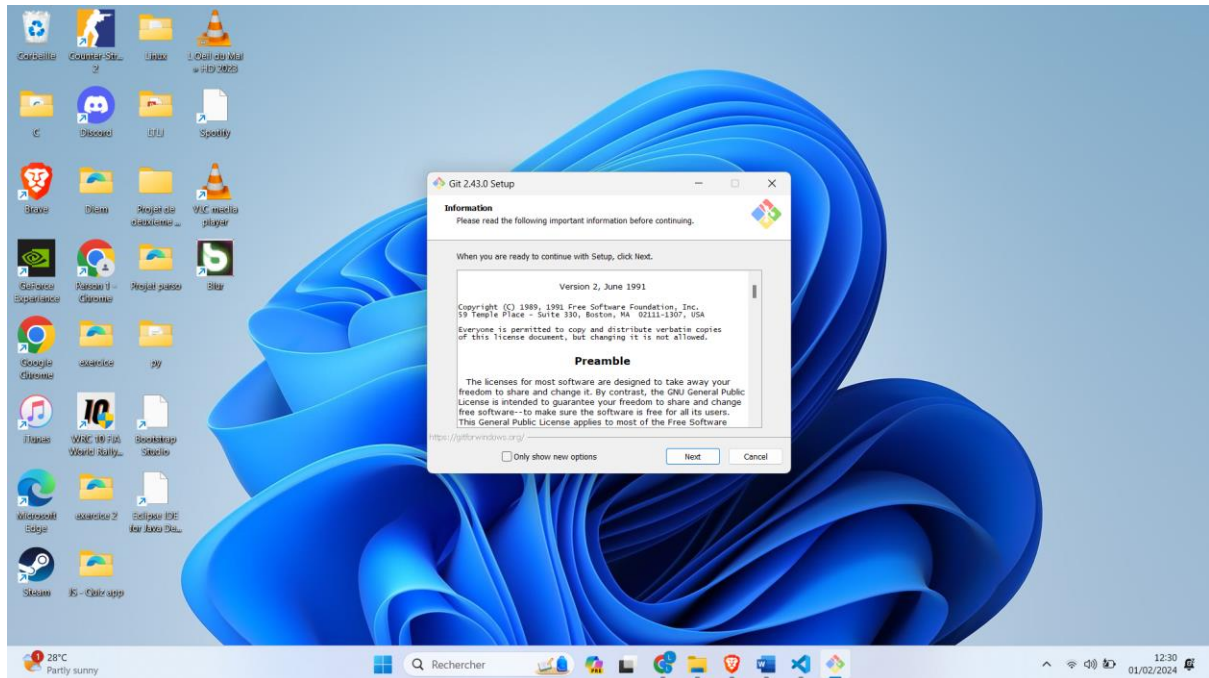
RAKOTONDRAMANANA A.A. LAFATRA

INFO 3 – UNIVERSITE DES MASCAREIGNES

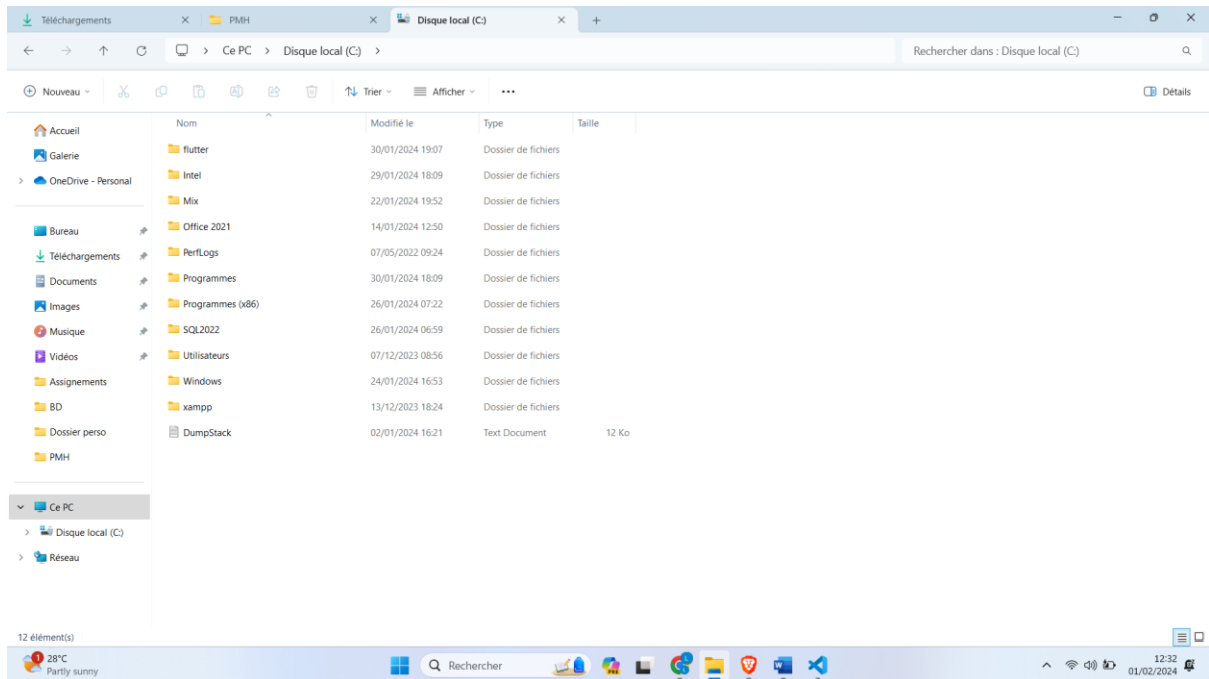
TP Session 1 : Programmation Mobile Hybride

Task 1 : Installation de l'environnement flutter :

- 1/ Installation de GIT

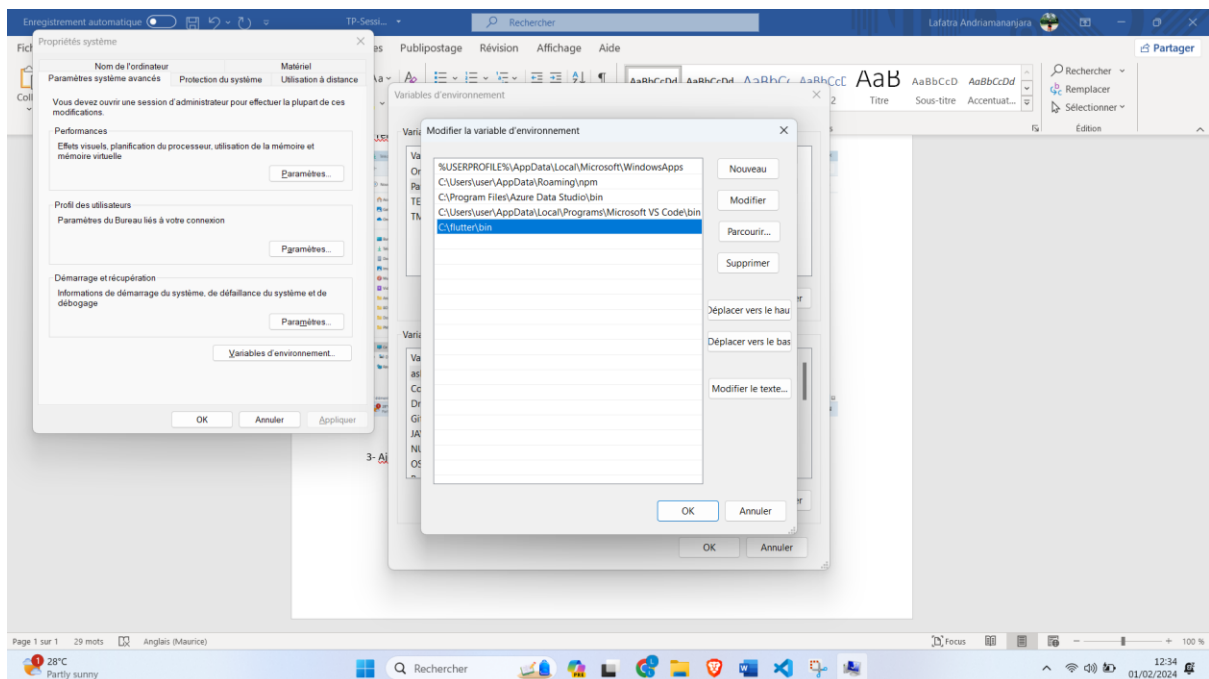


- 2/ Téléchargement, Extraction et déplacement de flutter

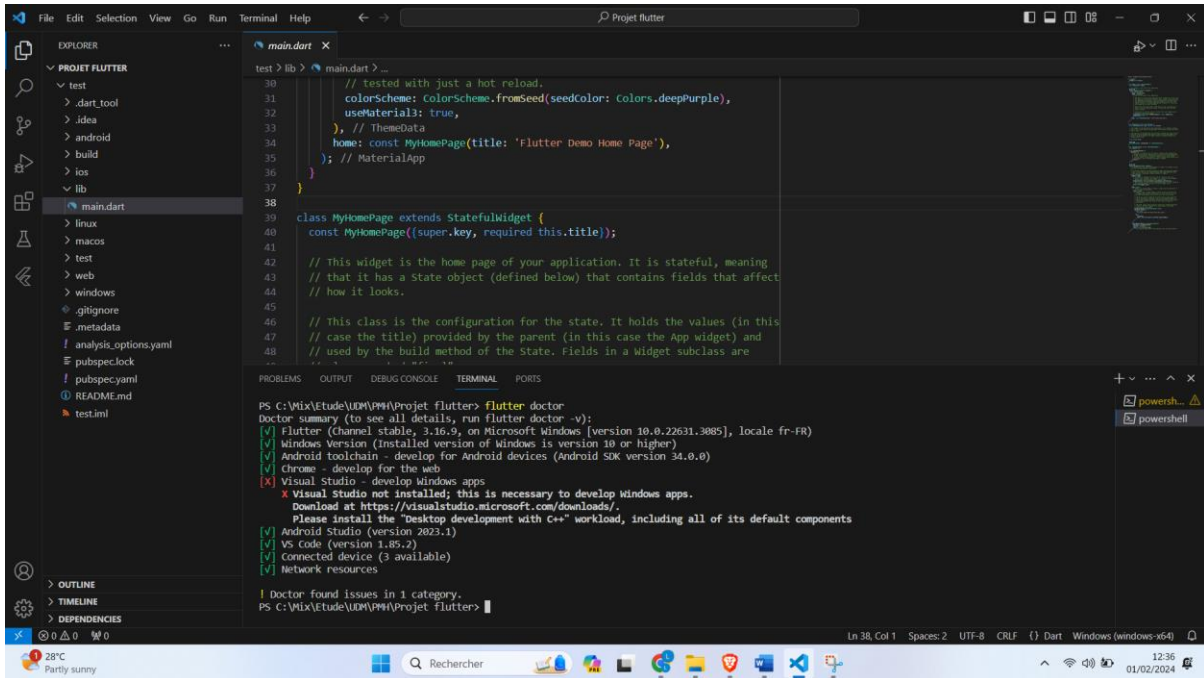


- 3/ Ajout de flutter dans les variables d'environnements de système

Ajouter le chemin du bin du dossier flutter dans "path"



- 4/ Installation de Android studio, de vs code, et réglage de SDK + test avec la commande 'flutter doctor -v'



The screenshot shows the Visual Studio Code interface with a Flutter project open. The Explorer panel on the left shows the project structure, including the 'lib' directory and the 'main.dart' file. The main editor displays the 'main.dart' file, which contains the following code:

```
test > lib > main.dart
30 // tested with just a hot reload.
31 colorScheme: ColorsScheme.fromSeed(seedColor: Colors.deepPurple),
32 useMaterial3: true,
33 }, // ThemeData
34 home: const MyHomePage(title: 'Flutter Demo Home Page'),
35 ); // MaterialApp
36 }
37
38 class MyHomePage extends StatefulWidget {
39   const MyHomePage({super.key, required this.title});
40
41   // This widget is the home page of your application. It is stateful, meaning
42   // that it has a state object (defined below) that contains fields that affect
43   // how it looks.
44
45   // This class is the configuration for the state. It holds the values (in this
46   // case the title) provided by the parent (in this case the App widget) and
47   // used by the build method of the State. Fields in a Widget subclass are
48   // ...
```

The Output panel at the bottom shows the results of the 'flutter doctor' command:

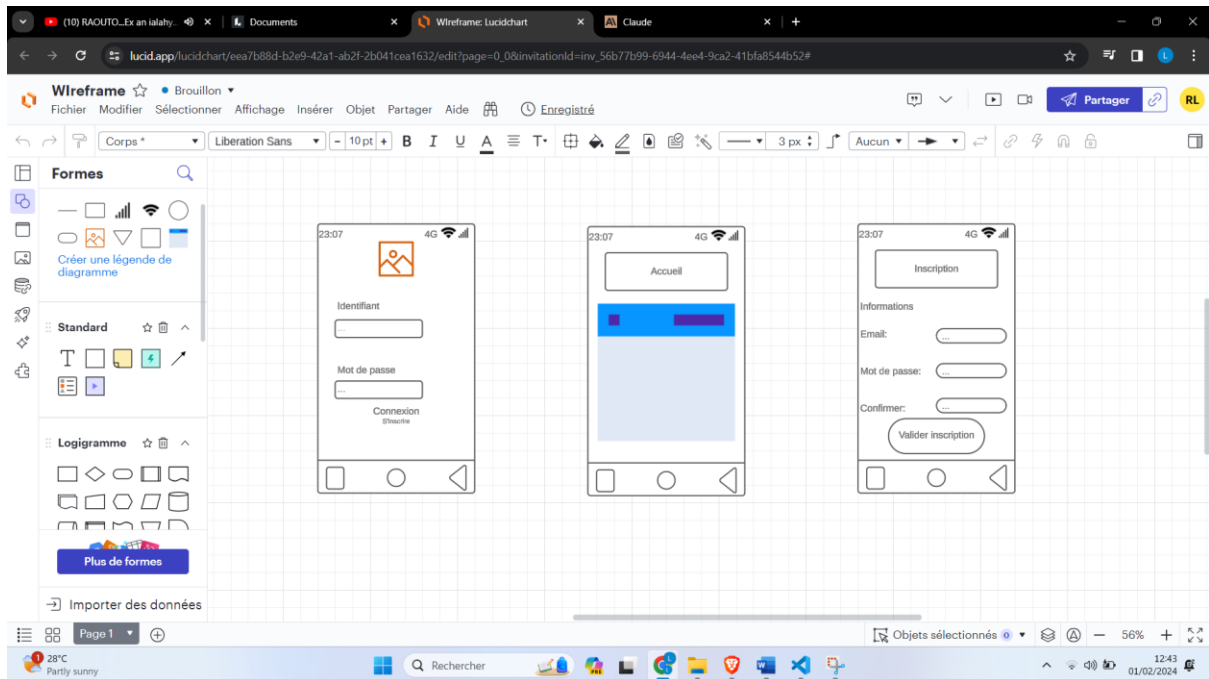
```
PS C:\Mix\étude\UDM\PMH\Projet flutter> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.16.9, on Microsoft Windows [version 10.0.22631.3885], locale fr-FR)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[✓] Android Studio (version 2023.1)
[✓] VS Code (version 1.85.2)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 1 category.
PS C:\Mix\étude\UDM\PMH\Projet flutter>
```

Task 2 : Création d'un Wireframe flutter avec un outil en ligne

Outil utilisée : Lucidchart.

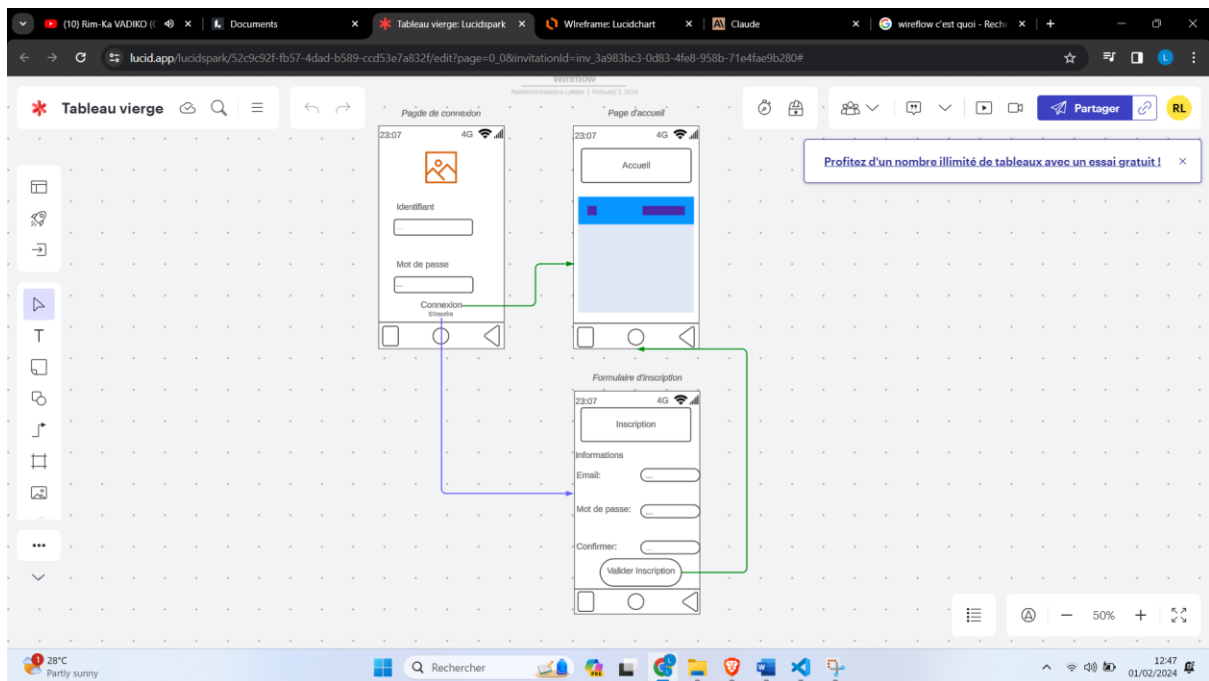
Lucidchart est un logiciel de diagrammes en ligne qui permet aux utilisateurs de créer visuellement différents types de diagrammes, comme des organigrammes, des diagrammes de flux, des cartes heuristiques, des maquettes de sites web, des schémas UML et bien d'autres.



Task 3 : Wireflow pour une application flutter

Outil utilisée : Lucidchart

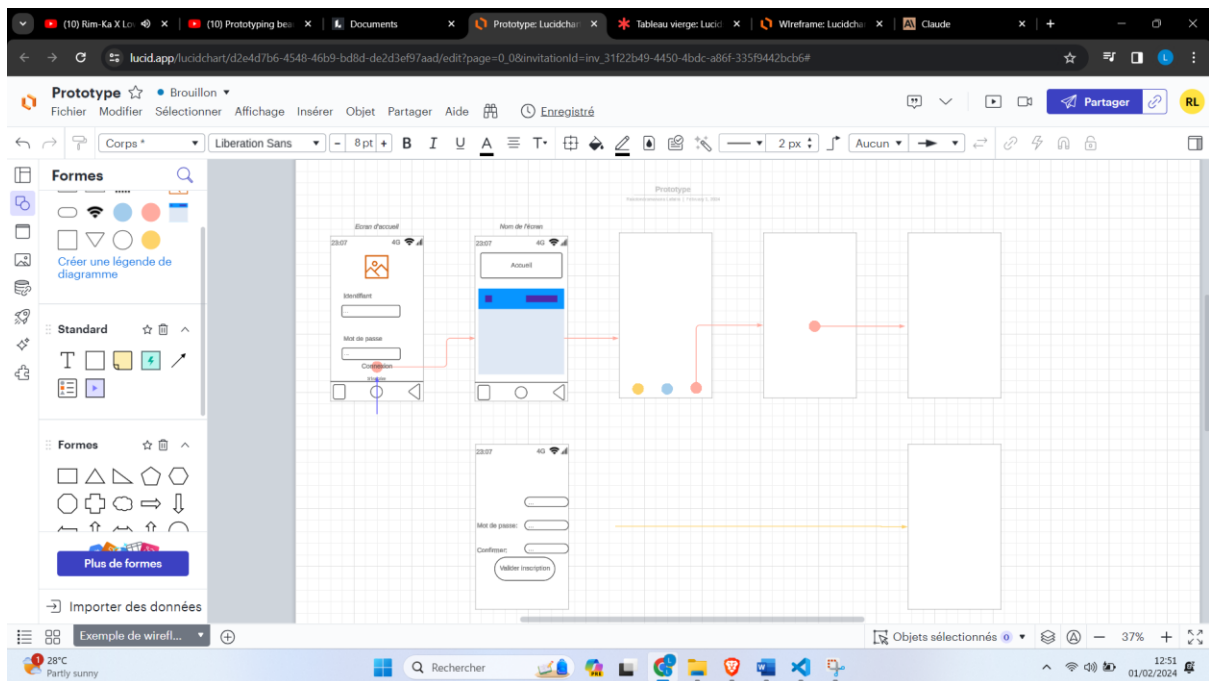
Lucidchart est un logiciel de diagrammes en ligne qui permet aux utilisateurs de créer visuellement différents types de diagrammes, comme des organigrammes, des diagrammes de flux, des cartes heuristiques, des maquettes de sites web, des schémas UML et bien d'autres.



Task 4 : Prototype d'une application flutter

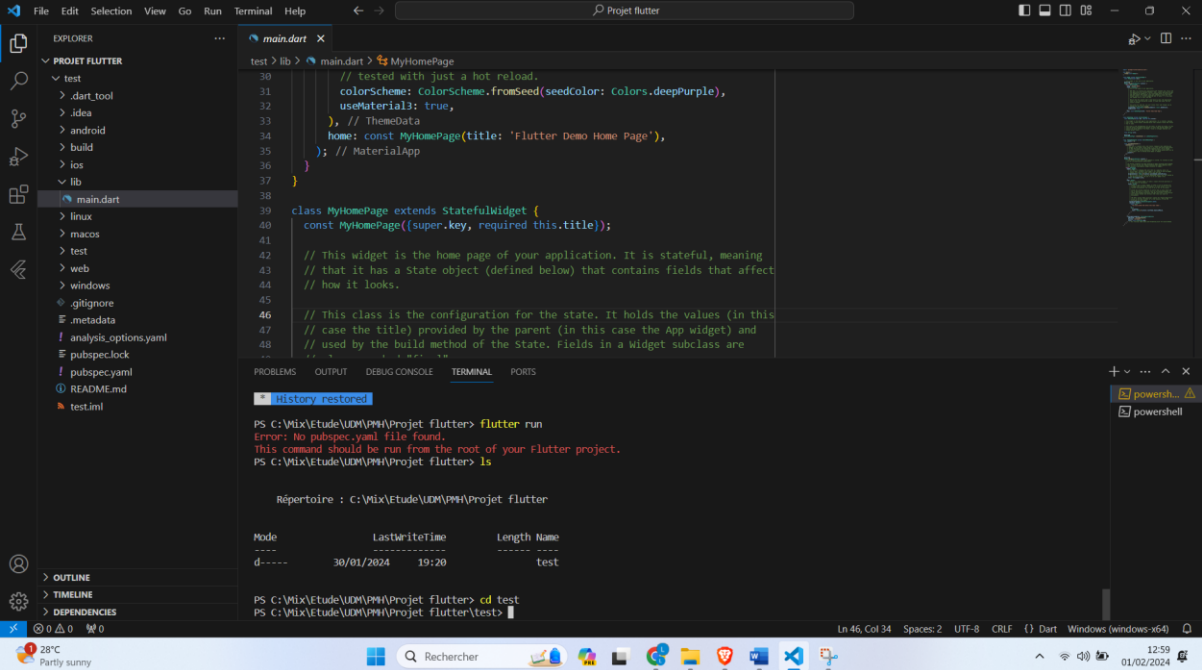
Outil utilisée : Lucidchart

Lucidchart est un logiciel de diagrammes en ligne qui permet aux utilisateurs de créer visuellement différents types de diagrammes, comme des organigrammes, des diagrammes de flux, des cartes heuristiques, des maquettes de sites web, des schémas UML et bien d'autres.



Task 5 : First flutter app in vs code

- 1/ Sur le terminal de vs code, exécuter la commande “flutter create + nom du projet” pour créer l’application.
- 2/ Exécution de la commande “flutter run” pour run l’application (il faut s’assurer que l’on se trouve dans le dossier de l’application que l’on vient de créer pour pouvoir exécuter cette commande. Ex: “cd test” => test est le nom du projet flutter)



```
test > lib > main.dart > MyHomePage
30 // tested with just a hot reload.
31 colorsScheme: ColorsScheme.fromSeed(seedColor: Colors.deepPurple),
32 useMaterial3: true,
33 ), // ThemeData
34 home: const MyHomePage(title: 'Flutter Demo Home Page'),
35 ); // MaterialApp
36 }
37 }
38
39 class MyHomePage extends StatefulWidget {
40   const MyHomePage({super.key, required this.title});
41
42   // This widget is the home page of your application. It is stateful, meaning
43   // that it has a State object (defined below) that contains fields that affect
44   // how it looks.
45
46   // This class is the configuration for the state. It holds the values (in this
47   // case the title) provided by the parent (in this case the App widget) and
48   // used by the build method of the State. Fields in a Widget subclass are
```

History restored

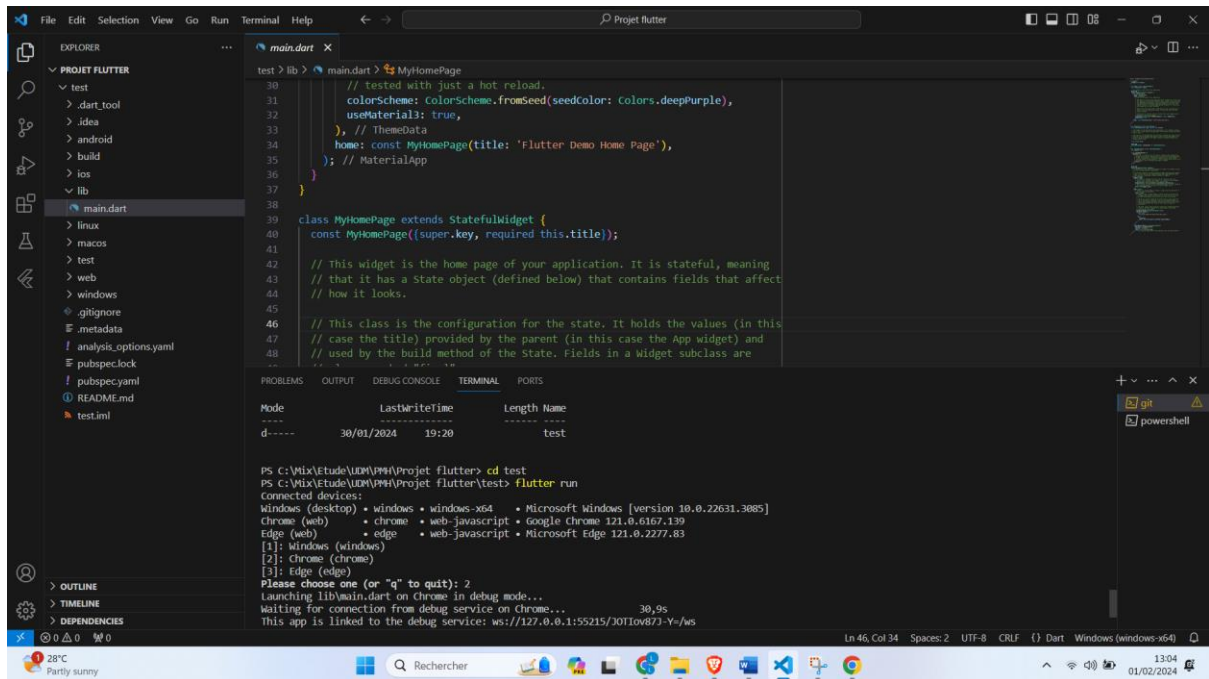
PS C:\Vix\Etude\UDM\PPHA\Projet flutter> flutter run
Error: No pubspec.yaml file found.
This command should be run from the root of your Flutter project.
PS C:\Vix\Etude\UDM\PPHA\Projet flutter> ls

Répertoire : c:\Vix\Etude\UDM\PPHA\Projet flutter

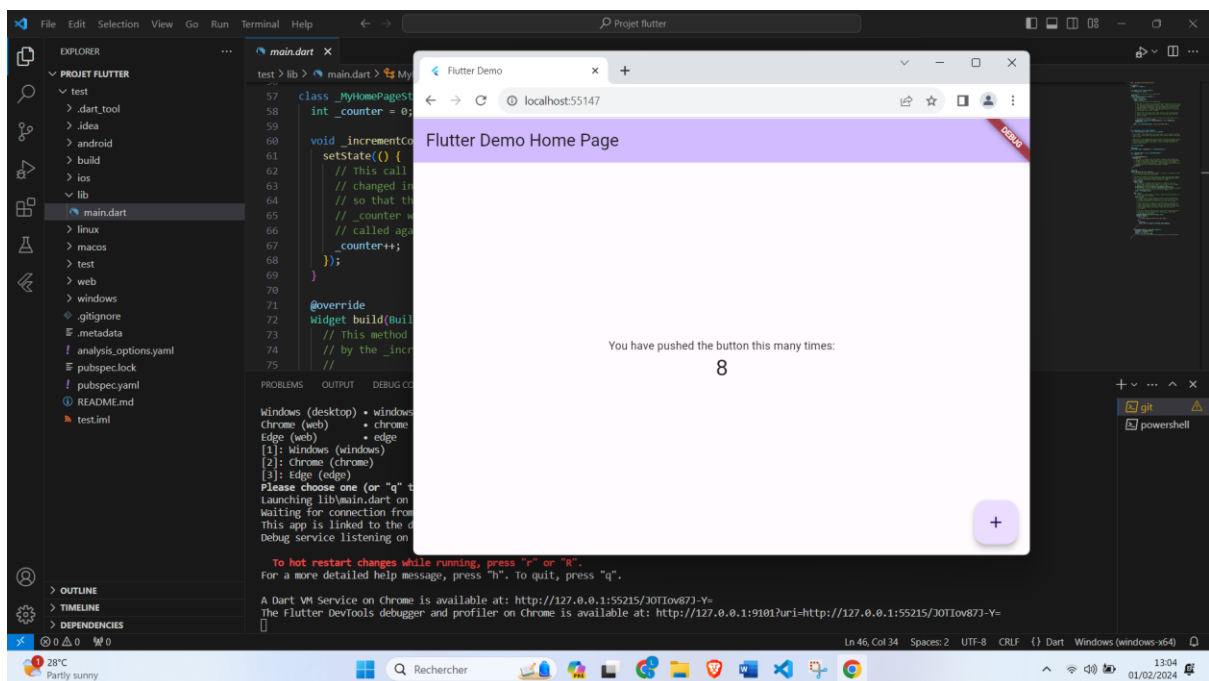
Mode	LastWriteTime	Length	Name
d----	30/01/2024 19:20		test

PS C:\Vix\Etude\UDM\PPHA\Projet flutter> cd test
PS C:\Vix\Etude\UDM\PPHA\Projet flutter\test>

- 3/ choix du navigateur pour exécuter l’application

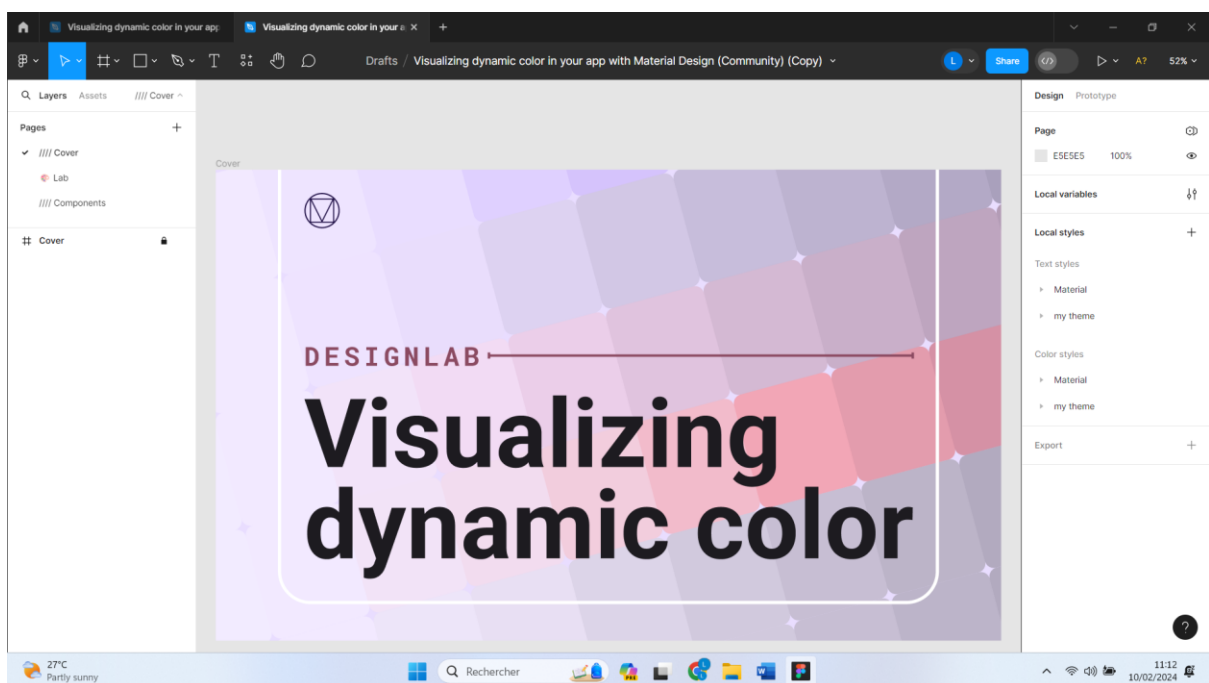


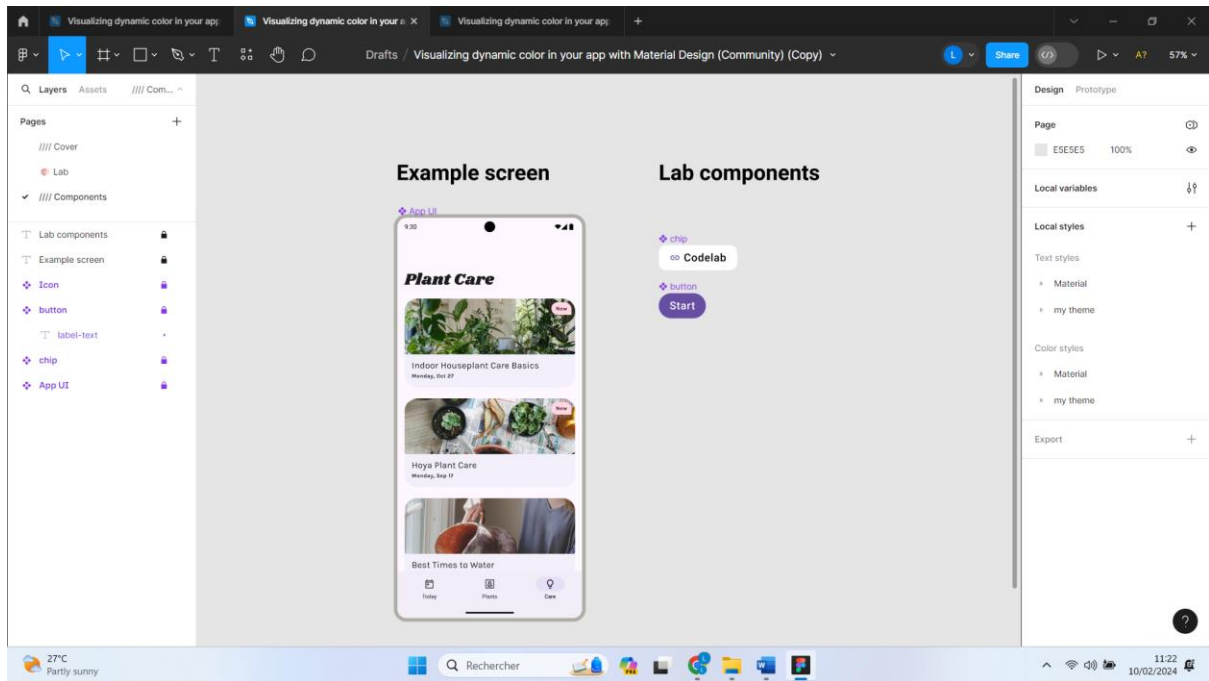
- 4/ Affichage de l'application exécuter par le navigateur chois



Task 6 : Visualisation des couleurs dynamiques dans votre application – tutoriel

1. Installer l'application Figma et créer un compte
2. Créer un nouveau projet Figma
3. Ajouter des rectangles de différentes couleurs pour représenter les écrans de l'application
4. Utiliser le "Variant" dans Figma pour créer différentes variantes de couleurs pour les écrans
5. Définir une couleur primaire et une couleur secondaire qui changeront entre les variantes
6. Animer la transition entre les différentes variantes de couleurs dans le prototype Figma
7. Exporter le fichier Figma et l'intégrer dans un projet d'application Android Studio
8. Utiliser la bibliothèque Palette dans l'application pour récupérer les couleurs du thème Figma
9. Appliquer les couleurs dynamiques aux vues de l'application en fonction du thème sélectionné

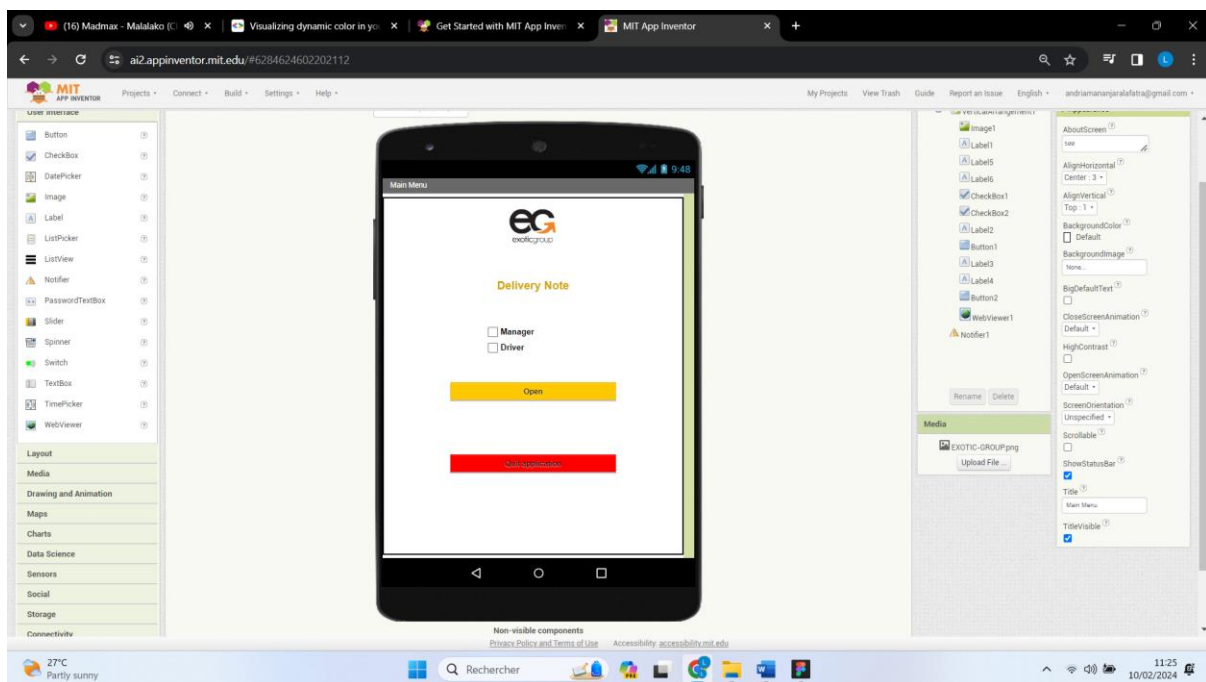




Task 7 : Application en utilisant MIT AppInventor

MIT App Inventor est une plateforme open-source qui permet de créer des applications Android sans avoir à coder. Elle fonctionne avec un système de glisser-déposer d'éléments visuels et de blocs de code.

- L'interface visuelle : On glisse et dépose des éléments d'interface comme des boutons, texte, images, etc. Cela crée l'UI de l'application.
- La logique : On utilise des blocs de code prédéfinis comme des blocs "si/alors" ou des boucles pour définir le comportement souhaité lorsqu'on interagit avec l'interface. On connecte ces blocs entre eux.
- Génération de l'app : Une fois l'interface et la logique définies, MIT App Inventor génère le code source de l'application Android et produit un fichier APK exécutable sur n'importe quel appareil Android.
- Test et débogage : On peut tester l'app en temps réel sur émulateur ou périphérique connecté pour déboguer et itérer dessus.



MIT App Inventor interface showing a project named "DELIVERY_NOTE". The interface includes a "Blocks" panel on the left, a "Viewer" panel on the right, and a "Media" panel at the bottom left. The "Blocks" panel lists various components like Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures, and Screen1. The "Viewer" panel displays a visual representation of the app's logic, including blocks for "when CheckBox1 Changed", "when CheckBox2 Changed", "when Button1 Click", "when Button2 Click", and "when Notifier1 AfterChoosing". The "Media" panel shows a file named "COTIC-GROUP.png" and an "Upload File" button. The interface also features a "Show Warnings" button and a "Privacy Policy and Terms of Use" link.

The logic blocks in the Viewer panel are as follows:

- when CheckBox1 Changed**
 - do if **CheckBox1** **Checked** **to** **false**
 - then **set CheckBox2** **Checked** **to** **false**
- when CheckBox2 Changed**
 - do if **CheckBox2** **Checked** **to** **false**
 - then **set CheckBox1** **Checked** **to** **false**
- when Button1 Click**
 - do if **CheckBox1** **Checked** **is** **true**
 - then **open another screen** **screenName** **Manager**
 - else if **CheckBox2** **Checked** **is** **true**
 - then **open another screen** **screenName** **Driver**
- when Button2 Click**
 - do **cat Notifier1** **ShowChooseDialog**
 - message** **Do You Want to Quit Delivery Note App?**
 - title** **Quit**
 - button1Text** **Yes**
 - button2Text** **No**
 - cancelable** **true**
- when Notifier1 AfterChoosing**
 - do **choice** **get choice** **is** **Yes**
 - then **close application**