Name- Aditya Kundu

Div-D15A

Roll No-31

**EXPERIMENT NO.2**

**AIM:- To design Flutter UI by including common widgets.**

**THEORY:-**

**Designing a Flutter UI involves creating a visually appealing and functional user interface by leveraging Flutter's rich set of widgets. Here's a theoretical approach to designing Flutter UI by including common widgets:**

**1. Understand User Requirements:**

**- Clearly understand the user requirements and the purpose of the application.**

**- Identify key features and functionalities the UI needs to support.**

**2. Wireframing:**

**- Create a basic wireframe of the app's layout. This helps in visualizing the structure and organization of the UI.**

**- Identify the main components and their relationships.**

**3. Identify Common Widgets:**

**- Recognize common UI elements required for most applications, such as buttons, text fields, images, lists, and navigation bars.**

**- Leverage Flutter's extensive set of pre-built widgets to save time and maintain consistency.**

**4. Structural Layout with Containers:**

**- Use `Container` widgets to define the structural layout of the UI.**

**- Utilize properties like `padding`, `margin`, and `alignment` to control spacing and positioning.**

**5. AppBar for Top Navigation:**

**- Include an `AppBar` at the top for navigation and displaying the app's title.**

**- Utilize `actions` for additional icons or buttons.**

**6. Bottom Navigation Bar:**

**- If the app requires bottom navigation, use the `BottomNavigationBar` widget.**

**- Assign each tab a specific function and navigate between them.**

**7. Card and ListTile for Information Display:**

**- Employ `Card` widgets to display information in a structured and visually appealing manner.**

**- For lists, use `ListView` along with `ListTile` for efficient and responsive information display.**

**8. Buttons and Gestures:**

**- Implement various button types (`ElevatedButton`, `TextButton`, `OutlinedButton`) depending on the design requirements.**

**- Utilize gesture detectors for more complex interactions.**

**9. Form Elements:**

**- If the app involves user input, incorporate form elements like `TextField`, `Checkbox`, `Radio`, and `DropDownButton`.**

**10. Images and Icons: Include `Image` widgets to display images.**

**- Integrate icons using `Icon` or `IconButton` for a more intuitive UI.**

**11. Theming:**

**- Apply consistent theming using `Theme` and ` ThemeData` to maintain a cohesive look and feel across the app.**

**- Customize colors, fonts, and other styling parameters.**

**12. Responsive Design:**

**- Design UI elements to adapt to different screen sizes and orientations.**

**- Utilize responsive layout techniques such as `MediaQuery` and `LayoutBuilder`.**

**13. Testing:**

**- Regularly test the UI on different devices and screen sizes to ensure a consistent user experience.**

**14. Refinement:**

**- Gather feedback from users or stakeholders and refine the UI based on their input.**

**- Optimize performance and address any usability issues.**

**By following these theoretical steps, you can efficiently design a Flutter UI by incorporating common widgets, resulting in a visually appealing and functional application.**

**CODE:-**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: VideoCallScreen(),

);

}

}

class VideoCallScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Microsoft Teams'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Placeholder(

// Video stream or preview placeholder

fallbackHeight: 200.0,

),

SizedBox(height: 20.0),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

IconButton(

icon: Icon(Icons.videocam),

onPressed: () {

// Toggle video on/off

},

),

SizedBox(width: 20.0),

IconButton(

icon: Icon(Icons.mic), // Corrected icon name

onPressed: () {

// Toggle microphone on/off

},

),

],

),

],

),

),

floatingActionButton: FloatingActionButton(

onPressed: () {

// End the call

},

child: Icon(Icons.call\_end),

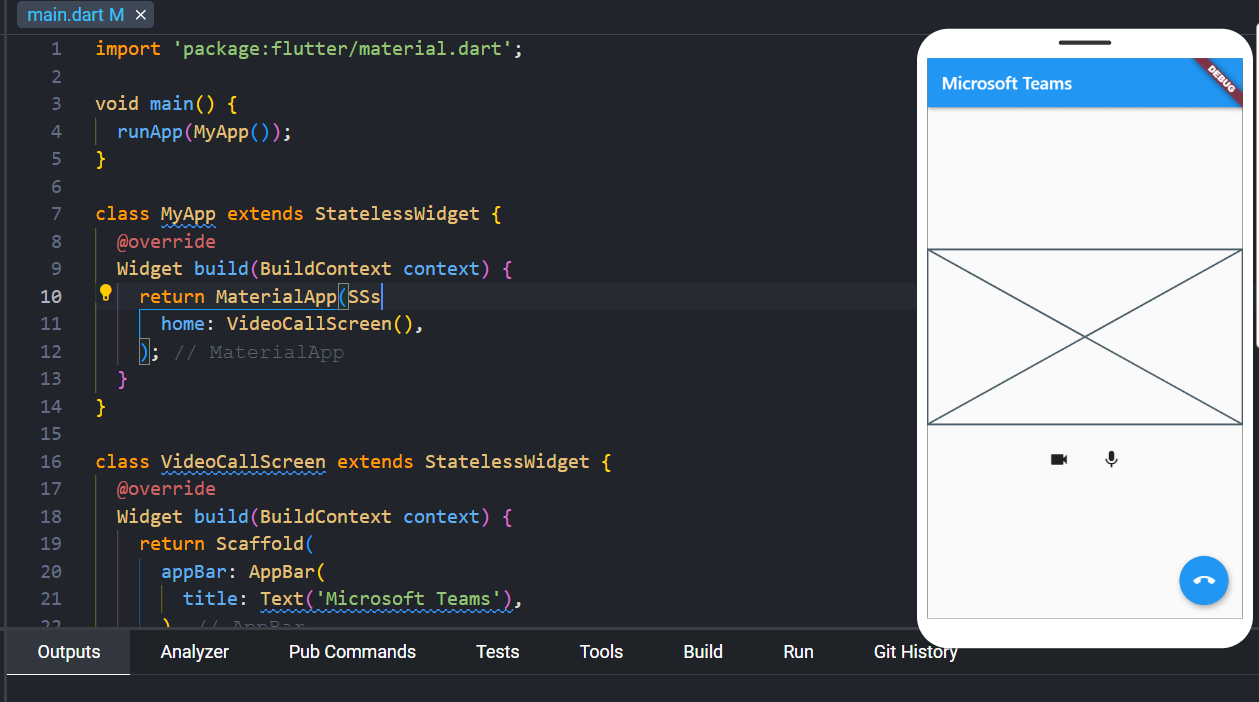
),

);

}

}

**OUTPUT:-**



**CONCLUSION:-**

From the above experiment, the widgets and syntax in flutter has been understood.