Experiment 2:
Name:Khushi Jeswani
Div: D15A
Roll no: 26

Aim: To design Flutter UI by including common widgets.

Theory: In Flutter, widgets are the building blocks of the user interface, and several common widgets play crucial roles in creating engaging and interactive applications. Here's a brief overview of some fundamental Flutter widgets:

Container: The most basic building block, a container is a box model that can contain other widgets, allowing you to customize its dimensions, padding, and decoration.

Row and Column: These widgets help organize children widgets horizontally (Row) or vertically (Column), facilitating the creation of flexible and responsive layouts.

AppBar: AppBar is a material design widget providing a top app bar that typically includes the app's title, leading and trailing icons, and actions.

ListView: Used to create scrollable lists of widgets, ListView is versatile for displaying a large number of items efficiently.

TextField: Enables users to input text, providing a text editing interface with options for validation, styling, and interaction.

ElevatedButton is a Flutter widget used to create a button with a raised appearance. It typically represents the primary action in a user interface. The button has a background color, elevation, and responds to user interactions with visual feedback

Image: The Image widget displays images from various sources, supporting both local and network images.

Scaffold: A top-level container for an app's visual elements, Scaffold provides a structure that includes an AppBar, body, and other optional features like drawers and bottom navigation.

Card: Representing a material design card, this widget displays information in a compact and visually appealing format, often used for grouping related content.

GestureDetector: Allows detection of various gestures like taps, drags, and long presses, enabling interactive responses to user input.

Stack: A widget that allows children widgets to be overlaid, facilitating complex UI designs by layering widgets on top of each other.

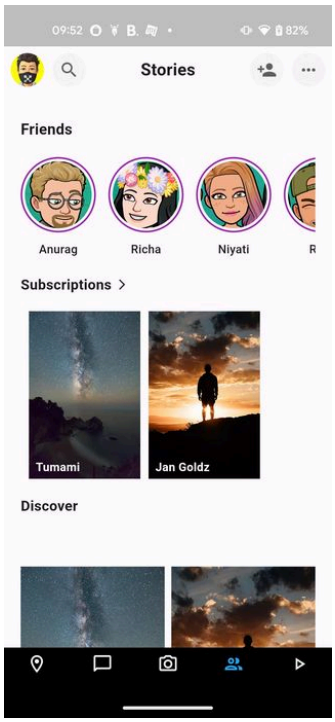
FutureBuilder: Ideal for handling asynchronous operations, FutureBuilder simplifies the management of UI updates based on the completion of a Future, making it valuable for fetching and displaying data.

These are just a few of the many widgets available in Flutter, each serving a unique purpose in crafting dynamic and user-friendly interfaces.

```
import 'dart:io';
import 'package:flutter/material.dart';
// import 'package:slicing_snapchat/firebase_options.dart';
// import 'package:slicing_snapchat/page/home_page.dart';
// import 'package:slicing_snapchat/page/initial_page.dart';
// import 'package:slicing_snapchat/page/login_page.dart';
// import 'package:slicing_snapchat/page/register_page.dart';

import 'package:firebase_core/firebase_core.dart';
import 'package:snapchatfinal/firebase_options.dart';
import 'package:snapchatfinal/page/home_page.dart';
import 'package:snapchatfinal/page/initial_page.dart';
import 'package:snapchatfinal/page/login_page.dart';
import 'package:snapchatfinal/page/register_page.dart';
// import 'package:get/get.dart';

void main() async{
  WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(
        options: DefaultFirebaseOptions.currentPlatform);

  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primaryColor: Color(0xFF838486),
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => InitialPage(),
```

```
        '/login_page': (context) => LoginPage(),
        '/register_page': (context) => RegisterPage(),
        '/home_page': (context) => HomePage(),
      },
    );
  }
}
```

App UI:



Widgets used: Icons, font, bottom navigation bar, image

Conclusion: Thus, understood the use of basic common widgets used in Mobile App Development and used some of them to create the login page for the chosen mini project application.