**Experiment 2**

**Shashwat Tripathi**

**D15A Batch C**

**Roll No: 64**

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

**THEORY:**

**Flutter Widgets: An Overview**

Flutter widgets are the basic building blocks that construct the user interface of a Flutter application. They are responsible for defining the structure, appearance, and behavior of the app. Here are some fundamental widgets:

**StatelessWidget and StatefulWidget:**

**StatelessWidget:**

Represents an immutable part of the user interface.

It does not change over time and does not depend on any mutable state.

**StatefulWidget:**

Represents a mutable part of the user interface.

Can change over time based on user interactions or other factors.

**1. Flutter Scaffold:**

The Scaffold widget is the basic structure for a Flutter app, providing a layout for the visual elements.

It includes an AppBar, BottomNavigationBar, and a body for the main content.

**2. Flutter Container:**

The Container widget is a versatile box model that can contain other widgets.

It's used for layout, padding, margin, decoration, and constraints.

**3. Flutter Row & Column:**

Row and Column widgets help in arranging child widgets horizontally (Row) or vertically (Column).

Useful for creating flexible and responsive layouts.

**4. Flutter Text:**

The Text widget is used to display text on the screen.

It supports various styling options like font size, color, and alignment.

**5. Flutter TextField:**

TextField is a widget for capturing user input, such as text, numbers, or passwords.

The onChanged property is commonly used for dynamic updates based on user input.

**6. Flutter Buttons:**

Button widgets, such as ElevatedButton or TextButton, trigger actions when pressed.

Provide a way for users to interact with the app.

**7. Flutter Forms:**

The Form widget helps in managing a group of TextFormField widgets.

Facilitates the validation and submission of user input.

**8. Flutter Icons:**

The Icon widget displays icons from various icon libraries, such as Material Icons or custom icons.

Enhances visual elements and conveys meaning through symbols.

**Key Design Principles:**

**Consistency:** Use of common widgets fosters a consistent design language throughout the app.

**Responsive Layouts:** Row and Column help create responsive and flexible layouts, adapting to different screen sizes.

**User Input Handling:** TextField and Form widgets facilitate user input handling, ensuring data integrity and validation.

**Interactive Elements:** Buttons and icons contribute to the interactivity and user engagement of the app.

**Visual Styling:** Container and styling properties of widgets allow for visual customization and theming.

**Code:**

**main.dart : Main entry point of our flutter app**

import 'package:flutter/material.dart';

import 'package:tiktok\_shashwat/constants.dart';

import 'package:tiktok\_shashwat/views/screens/auth/login\_screen.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

const MyApp({super.key});

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

// Application name

title: 'TikTok Clone',

theme: ThemeData.dark().copyWith(

scaffoldBackgroundColor: backgroundColor,

),

// A widget which will be started on application startup

home: LoginScreen(),

);

}

}

**constants.dart : Here the colors are defined to maintain consistency throughout the app.**

import 'package:flutter/material.dart';

// COLORS

const backgroundColor = Colors.black;

var buttonColor = Colors.red[400];

const borderColor = Colors.grey;

**add\_video\_screen.dart :**

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:tiktok\_tutorial/constants.dart';

import 'package:tiktok\_tutorial/views/screens/confirm\_screen.dart';

class AddVideoScreen extends StatelessWidget {

const AddVideoScreen({Key? key}) : super(key: key);

pickVideo(ImageSource src, BuildContext context) async {

final video = await ImagePicker().pickVideo(source: src);

if (video != null) {

Navigator.of(context).push(

MaterialPageRoute(

builder: (context) => ConfirmScreen(

videoFile: File(video.path),

videoPath: video.path,

),

),

);

}

}

showOptionsDialog(BuildContext context) {

return showDialog(

context: context,

builder: (context) => SimpleDialog(

children: [

SimpleDialogOption(

onPressed: () => pickVideo(ImageSource.gallery, context),

child: Row(

children: const [

Icon(Icons.image),

Padding(

padding: EdgeInsets.all(7.0),

child: Text(

'Gallery',

style: TextStyle(fontSize: 20),

),

),

],

),

),

SimpleDialogOption(

onPressed: () => pickVideo(ImageSource.camera, context),

child: Row(

children: const [

Icon(Icons.camera\_alt),

Padding(

padding: EdgeInsets.all(7.0),

child: Text(

'Camera',

style: TextStyle(fontSize: 20),

),

),

],

),

),

SimpleDialogOption(

onPressed: () => Navigator.of(context).pop(),

child: Row(

children: const [

Icon(Icons.cancel),

Padding(

padding: EdgeInsets.all(7.0),

child: Text(

'Cancel',

style: TextStyle(fontSize: 20),

),

),

],

),

),

],

),

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: Center(

child: InkWell(

onTap: () => showOptionsDialog(context),

child: Container(

width: 190,

height: 50,

decoration: BoxDecoration(color: buttonColor),

child: const Center(

child: Text(

'Add Video',

style: TextStyle(

fontSize: 20,

color: Colors.black,

fontWeight: FontWeight.bold,

),

),

),

),

),

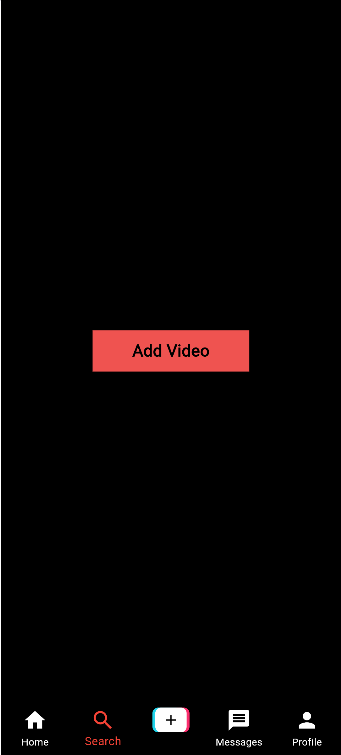
),

);

}

}

**OUTPUT :**



**CONCLUSION:** Thus, we have used some common widgets like Scaffold, Icon, Container, Button, etc. to create our login page of the application.