Sujal Taktani **Batch :- C**
D15A_60

Experiment no. :- 03

Aim :- To include icons, images, fonts in Flutter app

Theory :-

1) Button: the Button widget is not a specific widget, but rather a category of widgets that are used to handle user interaction by triggering actions when pressed. Some commonly used button widgets include: Elevated Button , Textfield Button, Outlined button etc

2.) Textfield with Icon: In Flutter, a TextField widget is used to allow users to input text. It is a fundamental part of many forms and input-based user interfaces. TextField provides a text input area where users can enter and edit text, and it comes with various customization options.

3.) Image : This widget holds the image which can fetch it from multiple sources like from the asset folder or directly from the URL. To add an image in the project, you need first to create an assets folder where you keep your images and then add the below line in pubspec.yaml file.

4.) Gesture Detection: To make an image interactive like a button, you need to detect user gestures such as taps. Flutter provides gesture detection widgets like GestureDetector .These widgets allow you to listen for various touch events like taps, swipes, and drags.I used this widget to make image as a button.

5.) Icon Button: Flutter provides the IconButton widget, which combines an icon with a tappable area, making it easy to create interactive icons that respond to user taps. The IconButton widget is commonly used for actions like navigation, opening menus, submitting forms, etc.

Code :-

```
import 'package:animated_login/animated_login.dart';
import 'package:async/async.dart';
import 'package:flutter/material.dart';

import 'auth.dart';



/// Example app widget.
class MyAppLogin extends StatelessWidget {
 /// Main app widget.
 const MyAppLogin({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Animated Login',
   theme: ThemeData(
    primaryColor: Colors.blue,
    colorScheme: const ColorScheme.light(primary: Colors.blue),
    // useMaterial3: true,
    textTheme: Theme.of(context).textTheme.apply(
     // Note: The below line is required due to a current bug in Flutter:
     // https://github.com/flutter/flutter/issues/129553
      decorationColor: Colors.blue),
    inputDecorationTheme: const InputDecorationTheme(
     prefixIconColor: Colors.black54,
     suffixIconColor: Colors.black54,
     iconColor: Colors.black54,
     labelStyle: TextStyle(color: Colors.black54),
     hintStyle: TextStyle(color: Colors.black54),
    ),
   ),
   debugShowCheckedModeBanner: false,
```

```
    initialRoute: '/login',
    routes: {
      '/login': (BuildContext context) => const LoginScreen(),
      '/forgotPass': (BuildContext context) => const ForgotPasswordScreen(),
    },
   );
 }

// static const Map<int, Color> color = {
//   50: Color.fromRGBO(4, 131, 184, .1),
//   100: Color.fromRGBO(4, 131, 184, .2),
//   200: Color.fromRGBO(4, 131, 184, .3),
//   300: Color.fromRGBO(4, 131, 184, .4),
//   400: Color.fromRGBO(4, 131, 184, .5),
//   500: Color.fromRGBO(4, 131, 184, .6),
//   600: Color.fromRGBO(4, 131, 184, .7),
//   700: Color.fromRGBO(4, 131, 184, .8),
//   800: Color.fromRGBO(4, 131, 184, .9),
//   900: Color.fromRGBO(4, 131, 184, 1),
// };
}

/// Example login screen
class LoginScreen extends StatefulWidget {
  /// Simulates the multilanguage, you will implement your own logic.
  /// According to the current language, you can display a text message
  /// with the help of [LoginTexts] class.
  const LoginScreen({Key? key}) : super(key: key);

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  /// Example selected language, default is English.
```

```
LanguageOption language = _languageOptions[1];

/// Current auth mode, default is [AuthMode.login].
AuthMode currentMode = AuthMode.login;

CancelableOperation? _operation;

@override
Widget build(BuildContext context) {
  return AnimatedLogin(
    onLogin: (LoginData data) async =>
        _authOperation(LoginFunctions(context).onLogin(data)),
    onSignup: (SignUpData data) async =>
        _authOperation(LoginFunctions(context).onSignup(data)),
    onForgotPassword: _onForgotPassword,
    logo: Image.asset('assets/images/logo.gif'),
    // backgroundImage: 'images/background_image.jpg',
    signUpMode: SignUpModes.both,
    socialLogins: _socialLogins(context),
    loginDesktopTheme: _desktopTheme,
    loginMobileTheme: _mobileTheme,
    loginTexts: _loginTexts,
    emailValidator: ValidatorModel(
        validatorCallback: (String? email) => 'What an email! $email'),
    changeLanguageCallback: (LanguageOption? _language) {
      if (_language != null) {
        DialogBuilder(context).showResultDialog(
            'Successfully changed the language to: ${_language.value}.');
        if (mounted) setState(() => language = _language);
      }
    },
    changeLangDefaultOnPressed: () async => _operation?.cancel(),
    languageOptions: _languageOptions,
    selectedLanguage: language,
    initialMode: currentMode,
```

```
    onAuthModeChange: (AuthMode newMode) async {
      currentMode = newMode;
      await _operation?.cancel();
    },
  );
}

Future<String?> _authOperation(Future<String?> func) async {
  await _operation?.cancel();
  _operation = CancelableOperation.fromFuture(func);
  final String? res = await _operation?.valueOrCancellation();
  if (_operation?.isCompleted == true) {
    DialogBuilder(context).showResultDialog(res ?? 'Successful.');
  }
  return res;
}

Future<String?> _onForgotPassword(String email) async {
  await _operation?.cancel();
  return await LoginFunctions(context).onForgotPassword(email);
}

static List<LanguageOption> get _languageOptions => const <LanguageOption>[
  LanguageOption(
    value: 'Turkish',
    code: 'TR',
    iconPath: 'assets/images/tr.png',
  ),
  LanguageOption(
    value: 'English',
    code: 'EN',
    iconPath: 'assets/images/en.png',
  ),
];
```

```
/// You can adjust the colors, text styles, button styles, borders
/// according to your design preferences for DESKTOP view.
/// You can also set some additional display options such as [showLabelTexts].
LoginViewTheme get _desktopTheme => _mobileTheme.copyWith(
  // To set the color of button text, use foreground color.
  actionButtonStyle: ButtonStyle(
    foregroundColor: MaterialStateProperty.all(Colors.white),
  ),
  dialogTheme: const AnimatedDialogTheme(
    languageDialogTheme: LanguageDialogTheme(
        optionMargin: EdgeInsets.symmetric(horizontal: 80)),
  ),
  loadingSocialButtonColor: Colors.blue,
  loadingButtonColor: Colors.white,
  privacyPolicyStyle: const TextStyle(color: Colors.black87),
  privacyPolicyLinkStyle: const TextStyle(
      color: Colors.blue, decoration: TextDecoration.underline),
);

/// You can adjust the colors, text styles, button styles, borders
/// according to your design preferences for MOBILE view.
/// You can also set some additional display options such as [showLabelTexts].
LoginViewTheme get _mobileTheme => LoginViewTheme(
  // showLabelTexts: false,
  backgroundColor: Colors.blue, // const Color(0xFF6666FF),
  formFieldBackgroundColor: Colors.white,
  formWidthRatio: 60,
  actionButtonStyle: ButtonStyle(
    foregroundColor: MaterialStateProperty.all(Colors.blue),
  ),
  animatedComponentOrder: const <AnimatedComponent>[
    AnimatedComponent(
      component: LoginComponents.logo,
      animationType: AnimationType.right,
    ),
```

```
    AnimatedComponent(component: LoginComponents.title),
    AnimatedComponent(component: LoginComponents.description),
    AnimatedComponent(component: LoginComponents.formTitle),
    AnimatedComponent(component: LoginComponents.socialLogins),
    AnimatedComponent(component: LoginComponents.useEmail),
    AnimatedComponent(component: LoginComponents.form),
    AnimatedComponent(component: LoginComponents.notHaveAnAccount),
    AnimatedComponent(component: LoginComponents.forgotPassword),
    AnimatedComponent(component: LoginComponents.policyCheckbox),
    AnimatedComponent(component: LoginComponents.changeActionButton),
    AnimatedComponent(component: LoginComponents.actionButton),
  ],
  privacyPolicyStyle: const TextStyle(color: Colors.white70),
  privacyPolicyLinkStyle: const TextStyle(
      color: Colors.white, decoration: TextDecoration.underline),
);

LoginTexts get _loginTexts => LoginTexts(
  nameHint: _username,
  login: _login,
  signUp: _signup,
  // signupEmailHint: 'Signup Email',
  // loginEmailHint: 'Login Email',
  // signupPasswordHint: 'Signup Password',
  // loginPasswordHint: 'Login Password',
);

/// You can adjust the texts in the screen according to the current language
/// With the help of [LoginTexts], you can create a multilanguage scren.
String get _username => language.code == 'TR' ? 'Kullanıcı Adı' : 'Username';

String get _login => language.code == 'TR' ? 'Giriş Yap' : 'Login';

String get _signup => language.code == 'TR' ? 'Kayıt Ol' : 'Sign Up';
```

```
  /// Social login options, you should provide callback function and icon path.
  /// Icon paths should be the full path in the assets
  /// Don't forget to also add the icon folder to the "pubspec.yaml" file.
  List<SocialLogin> _socialLogins(BuildContext context) => <SocialLogin>[
    SocialLogin(
        callback: () async => _socialCallback('Google'),
        iconPath: 'assets/images/google.png'),
    SocialLogin(
        callback: () async => _socialCallback('Facebook'),
        iconPath: 'assets/images/facebook.png'),
    SocialLogin(
        callback: () async => _socialCallback('LinkedIn'),
        iconPath: 'assets/images/linkedin.png'),
  ];

  Future<String?> _socialCallback(String type) async {
    await _operation?.cancel();
    _operation = CancelableOperation.fromFuture(
        LoginFunctions(context).socialLogin(type));
    final String? res = await _operation?.valueOrCancellation();
    if (_operation?.isCompleted == true && res == null) {
      DialogBuilder(context)
          .showResultDialog('Internal Server Error in with $type.');
    }
    return res;
  }
}

/// Example forgot password screen
class ForgotPasswordScreen extends StatelessWidget {
  /// Example forgot password screen that user is navigated to
  /// after clicked on "Forgot Password?" text.
  const ForgotPasswordScreen({Key? key}) : super(key: key);

  @override
```

```
Widget build(BuildContext context) {
  return const Scaffold(
    body: Center(
      child: Text('FORGOT PASSWORD'),
    ),
  );
 }
}
```

Output :-