

## MAD & PWA Lab

### Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	37
Name	Nihal Nagdev
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

**Aim:**

To design Flutter UI by including common widgets.

**Theory:**

Widgets are the building blocks of a Flutter application. In Flutter, everything is a widget, from simple elements like text and images to complex structures like entire layouts and navigations. Flutter provides a rich set of predefined widgets to create various UI components, which can be combined to build a complex UI.

**Common Flutter Widgets:**

1. Text Widget:
  - Used to display text on the screen. You can customize its style, size, color, and alignment.
2. Container Widget:
  - A box that can contain other widgets. It is used for styling, adding padding, margin, alignment, and background color to widgets.
3. Row and Column Widgets:
  - Row: Arranges widgets horizontally.
  - Column: Arranges widgets vertically.
  - These widgets are fundamental for creating flexible layouts and positioning UI elements.
4. Image Widget:
  - Used to display images in the app, either from assets, network, or file system.
5. Button Widgets:
  - Flutter offers several button widgets like RaisedButton, FlatButton, ElevatedButton, and IconButton that are used for interaction. These buttons are essential for handling user input and triggering actions.
6. TextField Widget:
  - Used for user input. It provides an editable field where the user can type text.
7. Scaffold Widget:

- This is a top-level container that holds the structure of the UI. It includes the app bar, body, drawer, and bottom navigation bar. It provides a standard layout for the app.

#### 8. ListView Widget:

- A scrolling widget that allows the display of a long list of items. It is used for displaying dynamic content efficiently.

#### Layouts in Flutter:

- **Padding:** Adds space around a widget.
- **Align:** Aligns a widget within its parent.
- **Expanded:** Makes a widget expand to fill available space in a Row, Column, or Flex.
- **Stack:** Used for placing widgets on top of one another.

#### **Conclusion:**

We learned how to design a basic Flutter UI by utilizing common widgets such as Text, Container, Row, Column, Image, Button, TextField, ListView, and Scaffold. These widgets provide a flexible and powerful way to create UIs that are visually appealing and functional. By combining these widgets, you can build complex layouts that cater to your app's design needs.

- Text and Container are fundamental for displaying text and styling.
- Row and Column are essential for structuring layouts, either horizontally or vertically.
- Buttons allow users to interact with the app.
- Scaffold provides a basic structure for the app, including app bars, bodies, and drawers.
- ListView is ideal for displaying lists of items, especially when the content is dynamic or long.

Mastering these common widgets enables developers to design clean and efficient UIs that cater to the needs of modern mobile applications.

#### **Code:**

```
import 'package:crimetrack/validation/validator.dart'; import
'package:flutter/material.dart'; import
'package:firebase_auth/firebase_auth.dart'; import
'package:fluttertoast/fluttertoast.dart'; // Import FlutterToast
import 'verify_page.dart'; // Import VerifyEmailScreen import
'../app_colors.dart'; // Import AppColors
```

```
class RegScreen extends StatefulWidget {  const
RegScreen({Key? key}) : super(key: key);
```

```
  @override
  _RegScreenState createState() => _RegScreenState();
}
```

```
class _RegScreenState extends State<RegScreen> {
  final _formKey = GlobalKey<FormState>();  final
  TextEditingController _nameController =
  TextEditingController();
  final TextEditingController _emailController =
  TextEditingController();
  final TextEditingController _passwordController =
  TextEditingController();
  final TextEditingController _confirmPasswordController =
  TextEditingController();
```

```
  bool _isPasswordVisible = false;  bool
  _isConfirmPasswordVisible = false;
  bool _isLoading = false;
```

```
  FocusNode _nameFocusNode = FocusNode();
  FocusNode _emailFocusNode = FocusNode();
  FocusNode _passwordFocusNode = FocusNode();
  FocusNode _confirmPasswordFocusNode = FocusNode();
```

```
  @override
  void dispose() {
    _nameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
```

```

        _confirmPasswordController.dispose();
        _nameFocusNode.dispose();
        _emailFocusNode.dispose();
        _passwordFocusNode.dispose();
        _confirmPasswordFocusNode.dispose();
        super.dispose();
    }

```

```

Future<void> _registerUser() async {
  if (_formKey.currentState?.validate() ?? false) {
    setState(() => _isLoading = true);
    try {
      UserCredential userCredential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
          email: _emailController.text, password:
            _passwordController.text);

```

```

      // Set display name after user is created
      await
        userCredential.user?.updateDisplayName(_nameController.text);

```

```

      // Send email verification
      await
        userCredential.user?.sendEmailVerification();

```

```

      // Show success toast
      Fluttertoast.showToast(
        msg: "Registration
        Successful! Please verify your email.",
        toastLength:
        Toast.LENGTH_SHORT,
        gravity:
        ToastGravity.BOTTOM,
        timeInSecForIosWeb: 1,
        backgroundColor: AppColors.successColor,
        textColor:
        AppColors.textColor,
        fontSize: 16.0,
      );

```

```

      // Navigate to VerifyEmailScreen
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => VerifyEmailScreen()),
      );
    } catch (e) {

```



```

        Padding(padding: const
EdgeInsets.only(top: 200.0), child:
Container(decoration: const
BoxDecoration(borderRadius:
BorderRadius.only(topLeft: Radius.circular(40),
topRight: Radius.circular(40)), color:
AppColors.backgroundColor,
),
height: double.infinity, width: double.infinity,
child: SingleChildScrollView(padding: const
EdgeInsets.symmetric(horizontal: 18.0, vertical: 30),
child: Form(
key: _formKey,
child: Column(
children: [
_buildTextField('Full Name', _nameController, false,
Validator.validateName, _nameFocusNode),
const SizedBox(height: 10),
_buildTextField('Email', _emailController,
false, Validator.validateEmail, _emailFocusNode),
const SizedBox(height: 10),
_buildTextField('Password', _passwordController, true,
Validator.validatePassword, _passwordFocusNode),
const SizedBox(height: 10),
_buildTextField('Confirm Password',
_confirmPasswordController, true, (value) { return
Validator.validateConfirmPassword(value ?? "",
_passwordController.text);
}, _confirmPasswordFocusNode),
const SizedBox(height: 50),
GestureDetector(onTap: _isLoading ? null
: _registerUser, child: Container(
height: 55, width: 300,
decoration: BoxDecoration(
borderRadius: BorderRadius.circular(30),
gradient: const LinearGradient(colors:
[AppColors.primaryColor,
AppColors.secondaryColor],

```

```

        ),
      child: Center(
        child: _isLoading
          ? const CircularProgressIndicator(color:
Colors.white)
          : const Text(
              'SIGN UP',
              style:
TextStyle(fontWeight: FontWeight.bold, fontSize: 20, color:
AppColors.buttonTextColor),
            ),
      ),
    ),
  ),
  const SizedBox(height: 50),
],
),
),
),
),
),
],
),
);
}

```

// Updated text field method

```

Widget _buildTextField(String label, TextEditingController
controller, bool isPassword, String? Function(String?) validator,
FocusNode focusNode) {  return TextFormField(    controller:
controller,    focusNode: focusNode, // Assign focus node
obscureText: isPassword ? (!_isPasswordVisible &&
!_isConfirmPasswordVisible) : false,    cursorColor:
AppColors.primaryColor, // Set cursor color to primary
    // Set selection color to primary
    decoration: InputDecoration(
      labelText: label,    labelStyle:
TextStyle(    fontWeight:
FontWeight.bold,    color:

```



```

focusNode.hasFocus ?
AppColors.primaryColor :
AppColors.secondaryColor, // Change label color on focus
    ),
    suffixIcon: isPassword ? IconButton( icon: Icon(
label == 'Password' ? (_isPasswordVisible ? Icons.visibility :
Icons.visibility_off)
: (_isConfirmPasswordVisible ? Icons.visibility
: Icons.visibility_off), color: Colors.grey,
    ),
    onPressed: () {
setState(() { if (label ==
'Password') {
_isPasswordVisible = !_isPasswordVisible;
} else {
_isConfirmPasswordVisible = !_isConfirmPasswordVisible;
}
});
},
    )
    : null,
    ),
    validator: validator, style:
TextStyle(color: focusNode.hasFocus ?
AppColors.primaryColor : AppColors.textColor), // Text color on focus
    );
}
}

```

Login Screen:

```

import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart'; // Import
FirebaseAuth package
import 'package:crimetrack/screens/forgot_password.dart'; // Import
ForgotPasswordScreen
import 'package:crimetrack/screens/home_screen.dart'; // Import
HomeScreen

```

```
import 'package:crimetrack/screens/register_screen.dart'; // Import
RegisterScreen import 'package:crimetrack/validation/validator.dart'; //
Import Validator class import '../app_colors.dart'; // Import AppColors
class import 'package:fluttertoast/fluttertoast.dart'; // Import fluttertoast
```

```
class LoginScreen extends StatefulWidget {  const
LoginScreen({Key? key}) : super(key: key);
```

```
  @override
  _LoginScreenState createState() => _LoginScreenState();
}
```

```
class _LoginScreenState extends State<LoginScreen> {  final
_formKey = GlobalKey<FormState>(); // Key to identify the form
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
```

```
  bool _isPasswordVisible = false; // Boolean variable to track password
visibility  bool _isLoading = false; // Boolean to track loading state
```

```
  @override
  Widget build(BuildContext context) {
return Scaffold(  appBar: AppBar(
leading: IconButton(
  icon: const Icon(Icons.arrow_back),
onPressed: () {
  Navigator.pop(context);
},
  color: AppColors.backgroundColor, // Set the back button color
to white
),
  backgroundColor: AppColors.primaryColor, // Use primaryColor
from AppColors
  elevation: 0,
),
```

```

        body: Stack(
children: [
    // Background gradient
    Container(
        height:
double.infinity,
        width:
double.infinity,
        decoration:
BoxDecoration(
            gradient:
LinearGradient(
                colors: [
                    AppColors.primaryColor, // Dark Blue from AppColors
                    AppColors.secondaryColor, // Light Blue from AppColors
                ],
            ),
        ),
        child: const Padding(
            padding:
EdgeInsets.only(top: 60.0, left: 22),
            child:
Text(
                'Hello\nSign in!',
                style:
TextStyle(
                    fontSize: 30,
                    color: AppColors.backgroundColor, // White color from
AppColors
                    fontWeight: FontWeight.bold,
                ),
            ),
        ),
        // Login form container
        Padding(
padding: const EdgeInsets.only(top: 200.0),
child: Container(
            decoration:
BoxDecoration(
                borderRadius: const
BorderRadius.only(
                    topLeft:
Radius.circular(40),
                    topRight:
Radius.circular(40),
                ),
            color: AppColors.backgroundColor, // White background for
the login form
        ),
        height: double.infinity,
        width:
double.infinity,
        child: Padding(
            padding:
const EdgeInsets.only(left: 18.0, right: 18),
            child:
Form(
                key: _formKey, // Attach the form key

```

```

child: Column(
    mainAxisAlignment:
MainAxisAlignment.center,
    children: [
        // Email input field with validation
        TextFormField(
            controller:
_emailController,
            decoration:
InputDecoration(
                suffixIcon:
const Icon(
                    Icons.check,
                    color: Colors.grey,
                ),
            label: Text(
                'Gmail',
                style:
TextStyle(
                    fontWeight: FontWeight.bold,
                    color:
AppColors.secondaryColor, // Set
the label color to secondary color
                ),
            ),
            enabledBorder: UnderlineInputBorder(
borderSide: BorderSide(color:
AppColors.primaryColor), // Set the border color to primary color
            ),
            focusedBorder: UnderlineInputBorder(
borderSide: BorderSide(color:
AppColors.primaryColor), // Set the border color to primary color when
focused
            ),
            hintStyle: TextStyle(
color: AppColors.primaryColor, // Set the hint text color to primary
color
            ),
            style: TextStyle(
color: AppColors.primaryColor, // Set the input text color to
primary color
            ),
            validator: (value) {
                return Validator.validateEmail(value); // Using
Validator class to validate email
            },
        ),
        const SizedBox(height: 20),
    ],
),

```

```

// Password input field with validation and visibility
toggle
    TextFormField(
        controller:
        _passwordController,
        obscureText: !_isPasswordVisible,
        // Toggle the visibility based on the boolean value
        decoration: InputDecoration(
            suffixIcon: IconButton(
                icon: Icon(
                    _isPasswordVisible
                    ? Icons.visibility
                    : Icons.visibility_off,
                color: AppColors.primaryColor,
            ),
            onPressed: () {
                setState(() {
                    _isPasswordVisible = !_isPasswordVisible; //
                Toggle password visibility
                });
            },
            label: Text(
                'Password',
                style: TextStyle(
                    fontWeight: FontWeight.bold,
                    color:
                    AppColors.secondaryColor, // Set the label color to secondary color
                ),
            ),
            enabledBorder: UnderlineInputBorder(
                borderSide: BorderSide(color:
                AppColors.primaryColor), // Set the border color to primary color
            ),
            focusedBorder: UnderlineInputBorder(
                borderSide: BorderSide(color:
                AppColors.primaryColor), // Set the border color to primary color when
                focused
            ),
            hintStyle: TextStyle(
                color: AppColors.primaryColor, // Set the hint text color to primary
                color
            ),
            style: TextStyle(
                color: AppColors.primaryColor, // Set the input text color to
                primary color

```

```

    ),
    validator: (value) {
      return Validator.validatePassword(value); // Using
Validator class to validate password
    },
  ),
  const SizedBox(height: 20),
  // Forgot password text
  Align(
    alignment:
  Alignment.centerRight,
    child:
  GestureDetector(
    onTap: () {
      // Navigate to ForgotPasswordScreen when tapped
  Navigator.push(
    context,
    MaterialPageRoute(
  builder: (context) => const
  ForgotPasswordScreen(),
    ),
  );
    },
    child: const Text(
  'Forgot Password?',
    style: TextStyle(
  fontWeight: FontWeight.bold,
    fontSize: 17,
  color: AppColors.secondaryColor, // Darker color for the text
    ),
  ),
  ),
  ),
  const SizedBox(height: 70),
  // Sign In button with loading indicator
  GestureDetector(
    onTap: () async {
  if (_formKey.currentState!.validate()) {
  setState(() {
    _isLoading = true; // Set loading state to true
  });

  // Attempt login using Firebase Authentication
  try {
    UserCredential userCredential = await
  FirebaseAuth.instance

```

```

        .signInWithEmailAndPassword(
email: _emailController.text,
password: _passwordController.text,
        );

        // Check if the email is verified
if (userCredential.user != null &&
userCredential.user!.emailVerified) {
        // Show success toast
Fluttertoast.showToast(
                                msg: "Login
Successful", // Toast message
toastLength: Toast.LENGTH_SHORT,
gravity: ToastGravity.BOTTOM,
timeInSecForIosWeb: 1,
backgroundColor: Colors.green,
                                textColor:
Colors.white,
                                fontSize: 16.0,
        );

        Navigator.push(
            context,
            MaterialPageRoute(
builder: (context) => const HomeScreen(),
            ),
        );
    } else {
        // Show email verification message as toast
Fluttertoast.showToast(
                                msg: "Please verify your
email before logging in.",
                                toastLength:
Toast.LENGTH_SHORT,
                                gravity:
ToastGravity.BOTTOM,
                                timeInSecForIosWeb: 1,
                                textColor:
Colors.orange,
                                fontSize: 16.0,
                                backgroundColor: Colors.white,
        );
    }
} catch (e) {
    String errorMessage = 'Error: Invalid Credentials';

    // Handle specific Firebase errors
if (e is FirebaseAuthException) {
        if (e.code ==

```

```

        'user-not-found') {
            found for that email.
        } else if (e.code ==
        'wrong-password') {
            password.
        } else if (e.code == 'invalid-email')
        {
            errorMessage = 'Invalid email address.';
        }
    }

    // Show error toast
    Fluttertoast.showToast(
        Display the error message in toast
        msg: errorMessage, //
        toastLength:
        Toast.LENGTH_SHORT,
        ToastGravity.BOTTOM,
        gravity:
        timeInSecForIosWeb: 1,
        backgroundColor: Colors.red,
        textColor: Colors.white,
        fontSize:
        16.0,
    );
} finally {
    setState() {
        _isLoading = false; // Set loading state to false
    });
}
},
child: _isLoading
? const CircularProgressIndicator() // Show loading
indicator while processing
: Container(
    height: 55,
    width: 300,
    decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(30),
        gradient: LinearGradient(
            colors: [
                AppColors.primaryColor,
                AppColors.secondaryColor,
            ],
        ),
    ),
    child: const Center(
        child: Text(
            'SIGN IN',
            style: TextStyle(

```



```


fontWeight: FontWeight.bold,
fontSize: 20,
                color: AppColors.backgroundColor,
            ),
        ),
    ),
),
    ),
    const SizedBox(height: 150),
    // Sign up link
    Align(
alignment: Alignment.bottomRight,
        child:
Column(
            crossAxisAlignment:
CrossAxisAlignment.end,
            children: [
const Text(
                "Don't have an account?",
style: TextStyle(
                    fontWeight:
FontWeight.bold,
                    color:
Colors.grey,
                ),
            ),
            GestureDetector(
onTap: () {
                // Navigate to the Register screen
Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => const RegScreen(), //
Navigate to RegisterScreen
                    ),
                );
            },
        child: const Text(
            "Sign up",
style: TextStyle(
                fontWeight:
FontWeight.bold,
                fontSize:
17,
                color:
AppColors.primaryColor, // Black color for
"Sign up"
            ),
        ),
    ),
),


```


$$\left\{ \begin{array}{l} \\ \end{array} \right\}$$

Output:

# Authentication



 Email

 Password

Login

Signup