MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
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 include icons, images, and fonts in a Flutter app.

Theory

1. Icons in Flutter

Icons are essential elements in modern mobile apps, visually representing actions, features, or categories. In Flutter, they enhance the UI's clarity and intuitiveness.

Types of Icons:

- Material Icons:
  - Built-in set of icons following Material Design guidelines.
  - Scalable and optimized for various devices and screen sizes.
- Custom Icons:
  - You can add and use your own icons as image assets. ○ Useful for branding and specific UI needs.

Benefits of Using Icons:

- Provide visual cues for actions.
- Improve app aesthetics.
- Offer consistent and recognizable symbols (e.g., home, settings).

2. Images in Flutter

Images enhance UI by making it more engaging and visually descriptive. Flutter supports multiple image sources.

Types of Images:

- Asset Images:
  - Stored locally in the app’s directory.
  - Bundled during the build process.

- Network Images:
    - Fetched from the internet (e.g., URLs, cloud-hosted content).
    - Great for displaying dynamic and real-time data.
- File Images:
    - Reside on the device's storage.
    - Often used for user-uploaded content.

Benefits of Using Images:

- Add visual context and appeal.
- Communicate ideas and features effectively. ● Support branding (e.g., logos, banners).

3. Fonts in Flutter

Fonts define the typographic style and play a vital role in an app's branding and readability.

Types of Fonts:

- Custom Fonts:
    - Support .ttf and .otf formats.
    - Defined in pubspec.yaml for app-wide use.
- Default Fonts:
    - Flutter provides system default fonts.

Benefits of Using Custom Fonts:

- Enhance aesthetic appeal.
- Ensure brand consistency.
- Enable flexible text styling for different elements (headings, paragraphs, etc.).

Code Implementation :

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

```
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);

await
userCredential.user?.updateDisplayName(_nameController.text);
await userCredential.user?.sendEmailVerification();

       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,
       );

       Navigator.pushReplacement(
         context,
         MaterialPageRoute(builder: (context) => VerifyEmailScreen()),
       );
     } catch (e) {
       Fluttertoast.showToast(           msg:
"Error: ${e.toString()}",           toastLength:
Toast.LENGTH_SHORT,           gravity:
ToastGravity.BOTTOM,
timeInSecForIosWeb: 1,
backgroundColor: AppColors.errorColor,
textColor: AppColors.textColor,
fontSize: 16.0,
       );
     } finally {         setState(() =>
_isLoading = false);
     }
```

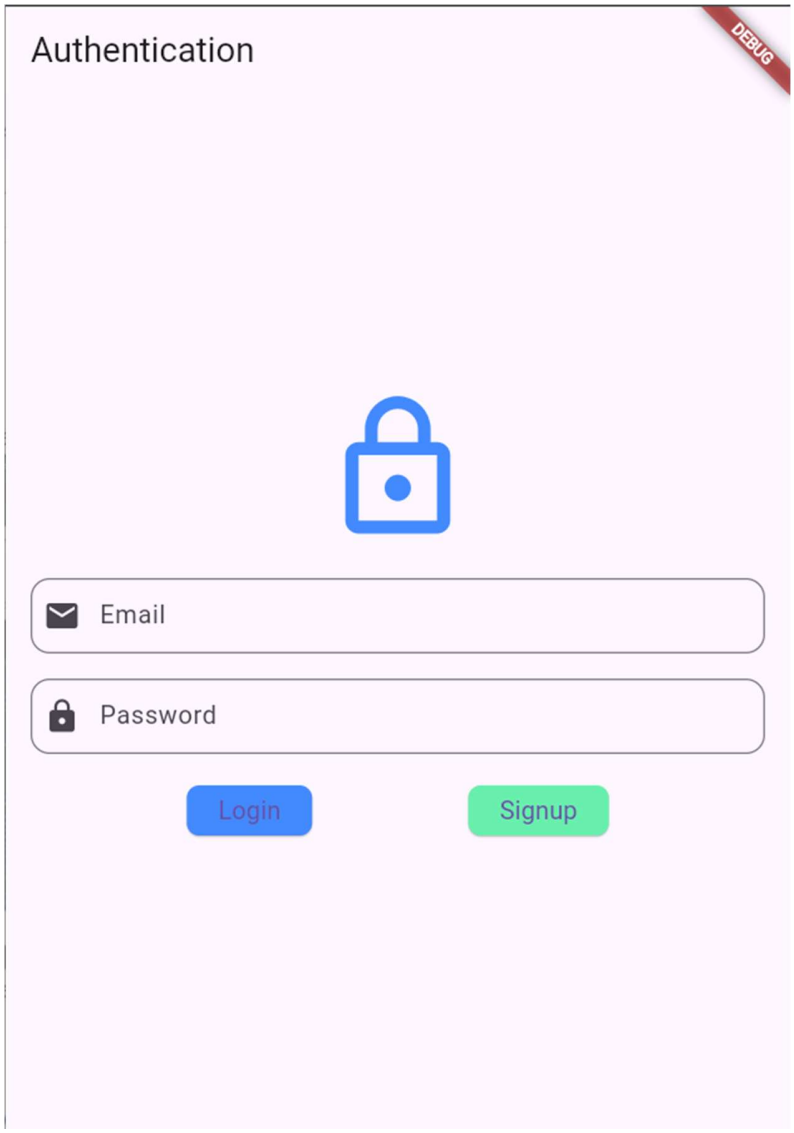
```
  }
 }

 @override
 Widget build(BuildContext context) {
return Scaffold(       body: Stack(
children: [
        // Background Gradient
Container(             height: double.infinity,
width: double.infinity,             decoration:
const BoxDecoration(               gradient:
LinearGradient(                colors:
[AppColors.primaryColor,
AppColors.secondaryColor],
            ),
          ),
          child: const Padding(               padding:
EdgeInsets.only(top: 60.0, left: 22),               child:
Text(
              'Create Your\nAccount',
style: TextStyle(                   fontSize:
30,                 color: Colors.white,
fontWeight: FontWeight.bold),
            ),
          ),
        ),
        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),
              ],
            ),
          ),
        ),
      ),
    ),
  ],
 ),
 );
 }
}
```

Output:



Conclusion:

We learned how to include icons, images, and custom fonts in a Flutter app to elevate the visual appeal and user experience.

Key Takeaways:

- Icons provide intuitive navigation and reinforce meaning.
- Images enhance design and communicate content visually.
- Custom Fonts support brand identity and text styling flexibility.

Mastering these assets is crucial for creating visually polished and user-friendly Flutter apps.