EXPERIMENT NO: - 03

Name:- Himesh Pathai Class:- D15A Roll:No: - 34

AIM: - To include icons, images, fonts in Flutter app.

Theory: -

Incorporating Visual Elements in Flutter: Icons, Images, and Custom Fonts

Flutter is a powerful open-source UI framework that enables developers to build natively compiled applications for mobile, web, and desktop platforms—all from a single codebase. One of Flutter's greatest strengths is its flexibility in crafting highly customizable UIs.

This practical guide focuses on integrating essential visual elements—icons, images, and custom fonts—into a Flutter application. These elements enhance visual appeal, improve usability, and create a more engaging user experience.

Importance of Visual Elements in App Development

- Enhanced User Experience – Icons and images make applications more visually appealing and userfriendly.
- Efficient Communication – Well-designed icons convey information quickly, reducing the need for lengthy text.
- Brand Identity – Custom icons and images reinforce branding, making an app more memorable.

Managing Assets in a Flutter App

When a Flutter app is built, it consists of both code and assets. Assets include static files such as images, icons, fonts, and configuration files, which are deployed and available at runtime. Flutter supports multiple image formats, including JPEG, WebP, PNG, GIF, BMP, and WBMP.

Adding Icons in Flutter

Flutter provides built-in Material Design icons through the Icons class. Custom icons can also be integrated using third-party packages like flutter_launcher_icons and font_awesome_flutter. Example (Built-in Material Icons):

Code :
Main-app/dashboard

```
import 'package:flutter/material.dart';
import 'package:get/get.dart'; // Get for dark mode support
import 'package:inventory/lumina/src/features/authentication/controllers/componentController.dart';
import 'package:inventory/lumina/src/features/main_app/dashboard/classScreen.dart';

class ClassContainer extends StatelessWidget {
  final String label;

  ClassContainer({required this.label});

  final ComponentController componentController =
      Get.put(ComponentController());

  @override
  Widget build(BuildContext context) {
    bool isDarkMode = Get.isDarkMode;

    return InkWell(
      onTap: () {
        componentController.ClassName.value = label;
        Navigator.of(context).push(
          MaterialPageRoute(builder: (context) => Classscreen(title: label)),
        );
      },
      splashColor: const Color(0xFFD3DCF2),
      child: Container(
        margin: const EdgeInsets.all(5),
        padding: const EdgeInsets.symmetric(horizontal: 10),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(10),
          gradient: isDarkMode
              ? const LinearGradient(
                  colors: [Color(0xFF507DBC), Color(0xFF70A1D7)],
                  begin: Alignment.topLeft,
                  end: Alignment.bottomRight,
                )
              : const LinearGradient(
                  colors: [
                    Color.fromARGB(255, 154, 210, 255),
                    Color.fromARGB(255, 213, 245, 252),
                  ],
                  begin: Alignment.topLeft,
                  end: Alignment.bottomRight,
                ),
          boxShadow: [
            BoxShadow(
              color: isDarkMode
                  ? Color(0xFF70A1D7).withOpacity(0.5)
```

```
                : Colors.black.withOpacity(0.2),
            offset: Offset(2, 3),
            blurRadius: isDarkMode ? 12 : 6,
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment:
            MainAxisAlignment.center, // Center contents vertically
        crossAxisAlignment:
            CrossAxisAlignment.center, // Center contents horizontally
        children: [
          // Centered Icon
          Icon(
            Icons.memory,
            color: isDarkMode ? Colors.white : Colors.black,
            size: 40, // Adjust size as needed
          ),
          const SizedBox(height: 8), // Spacing between icon and label
          // Label at the bottom
          Text(
            label,
            style: Theme.of(context).textTheme.titleLarge?.copyWith(
                  color: isDarkMode ? Colors.white : Colors.black,
                  fontWeight: FontWeight.bold,
                  fontSize: 19,
                ),
            textAlign: TextAlign.center, // Center text
            maxLines: 2, // Allow up to two lines
            overflow: TextOverflow.ellipsis, // Show "..." if text is too long
          )
        ],
      ),
    ),
  );
 }
}
```

Profile Screen:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:image_picker/image_picker.dart';
import 'package:inventory/lumina/src/features/authentication/controllers/emailcontroller.dart';
import 'package:inventory/lumina/src/features/authentication/screens/log_out_widget.dart';
import 'package:qr_flutter/qr_flutter.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'dart:convert';
import 'package:inventory/lumina/src/features/authentication/screens/verify_old_pass.dart';

class ProfileScreen extends StatefulWidget {
  const ProfileScreen({super.key});

  @override
  State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  final _supabase = Supabase.instance.client;
  final Emailcontroller emailGet = Get.put(Emailcontroller());
  File? _image;
  final ImagePicker _picker = ImagePicker();

  @override
  void initState() {
    super.initState();
    naamkaran();
    _loadImage();
  }

  Future<void> naamkaran() async {
    while (emailGet.emailget.value.isEmpty) {
      print("Waiting for email to be set...");
      await Future.delayed(Duration(milliseconds: 200)); // Small wait loop
    }

    try {
      final response = await _supabase
          .from('Members')
          .select()
          .eq('Email Id', emailGet.emailget.value)
          .maybeSingle();

      if (response != null) {
        emailGet.Namefrommail.value = response['Name'] ?? 'Guest';
        emailGet.idfrommail.value = response['ISA Login ID'] ?? '';
```

```
      emailGet.qrData.value = jsonEncode({
        'member_id': response['ISA Login ID'] ?? '',
        'email_id': response['Email Id'] ?? '',
        'division': response['Division'] ?? '',
        'name': response['Name'] ?? 'Guest',
        'phone_number': response['Phone Number'] ?? '',
      });

      print("User data fetched successfully!");
    } else {
      print("No user data found.");
    }
  } catch (e) {
    print('Error in naamkaran: $e');
  }
}

Future<void> _loadImage() async {
  final user = _supabase.auth.currentUser;
  if (user == null) return;

  try {
    // Query the database for the profile image URL
    final response = await _supabase
        .from('profiles')
        .select('profile_image_url')
        .eq('id', user.id)
        .maybeSingle();

    // Check if response contains data
    if (response != null && response['profile_image_url'] != null) {
      final imageUrl = response['profile_image_url'] as String;
      emailGet.profileImageUrl.value =
          '$imageUrl?timestamp=${DateTime.now().millisecondsSinceEpoch}';
    } else {
      // No profile image URL found, set default image
      emailGet.profileImageUrl.value =
          ''; // Or assign a default image URL if you have one
    }
  } catch (e) {
    print('Error loading profile image: $e');
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Failed to load profile image.')),
    );
  }
}

Future<void> _pickImage() async {
  final user = _supabase.auth.currentUser;
  if (user == null) {
    ScaffoldMessenger.of(context).showSnackBar(
```

```
    const SnackBar(content: Text('No user is currently logged in.')),
   );
   return;
  }

  final pickedFile = await _picker.pickImage(source: ImageSource.gallery);

  if (pickedFile != null) {
   File file = File(pickedFile.path);

   try {
    // First, get the ISA Login ID from Members table
    final memberResponse = await _supabase
      .from('Members')
      .select('"ISA Login ID"')
      .eq('Email Id', emailGet.emailget.value)
      .single();

    final String isaLoginId = memberResponse['ISA Login ID'] as String;

    // Upload the file with the same filename
    String fileName = 'profile_${user.id}.jpg';

    // Delete the old file (optional but recommended)
    await _supabase.storage.from('profile-images').remove([fileName]);

    // Upload the new file
    await _supabase.storage.from('profile-images').upload(fileName, file);

    // Get the public URL
    final imageUrl =
      _supabase.storage.from('profile-images').getPublicUrl(fileName);

    // Update the database with both image URL and member_id
    await _supabase.from('profiles').upsert({
     'id': user.id,
     'profile_image_url': imageUrl,
     'member_id': isaLoginId, // Adding the member_id from ISA Login ID
    });

    // Update the observable and invalidate the cache by appending a timestamp
    emailGet.profileImageUrl.value =
      '$imageUrl?timestamp=${DateTime.now().millisecondsSinceEpoch}';

    setState(() {
     _image = file;
    });

    ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(content: Text('Profile image updated successfully!')),
    );
```

```
    } catch (e) {
      print('Error: $e');
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('An unexpected error occurred.')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  final theme = Theme.of(context);
  bool isDarkMode = Get.isDarkMode;

  return Scaffold(
    appBar: AppBar(
      backgroundColor: const Color(0xff2196F3),
      title: Text(
        'My Profile',
        style: GoogleFonts.lato(
          fontSize: 22,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
      ),
      leading: IconButton(
        icon: const Icon(Icons.arrow_back, color: Colors.white),
        onPressed: () => Navigator.pop(context),
      ),
    ),
    body: SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 30.0),
        child: Column(
          children: [
            Center(
              child: Obx(() {
                String profileImageUrl = emailGet.profileImageUrl.value;
                return CircleAvatar(
                  backgroundImage: profileImageUrl.isNotEmpty
                      ? NetworkImage(profileImageUrl) as ImageProvider
                      : const AssetImage(
                          "assets/images/default_profile_image.png"),
                  radius: 80,
                );
              }),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: _pickImage,
              style: ElevatedButton.styleFrom(
```

```
    backgroundColor: const Color(0xff2196F3),
    shape: RoundedRectangleBorder(
     borderRadius: BorderRadius.circular(12),
    ),
    padding: const EdgeInsets.symmetric(
     horizontal: 20,
     vertical: 12,
    ),
   ),
   child: Text(
    'Add/Change Profile Image',
    style: GoogleFonts.lato(
     color: Colors.white,
     fontSize: 16,
     fontWeight: FontWeight.w600,
    ),
   ),
  ),
 ),
 const SizedBox(height: 5),
 Obx(
  () {
    String name = emailGet.Namefrommail.value.isNotEmpty
        ? emailGet.Namefrommail.value
        : 'Guest';
    return Container(
     height: 55,
     width: 500,
     child: Center(
        child: Text(
       name,
       style: theme.textTheme.bodyLarge?.copyWith(
        fontSize: 21,
        fontWeight: FontWeight.bold,
        color: isDarkMode ? Colors.white : Colors.black,
       ),
      )),
    );
  },
 ),
 const SizedBox(height: 1),
 Text(
  "ISA Member",
  style: GoogleFonts.lato(
   fontSize: 18,
   fontWeight: FontWeight.w400,
   color: Colors.grey,
  ),
 ),
 const SizedBox(height: 30),
 Obx(
  () => QrImageView(
```

```
      data: emailGet.qrData.value,
      version: QrVersions.auto,
      size: 180.0,
      backgroundColor: Colors.white,
      padding: const EdgeInsets.all(10),
    ),
  ),
  const SizedBox(height: 40),
  ElevatedButton.icon(
    onPressed: () {
      final userEmail = emailGet.emailget.value;
      if (userEmail.isNotEmpty) {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => VerifyOldPasswordScreen(
              email: userEmail,
            ),
          ),
        );
      } else {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text('No email found. Please try again.'),
          ),
        );
      }
    },
    style: ElevatedButton.styleFrom(
      backgroundColor: const Color(0xff2196F3),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(12),
      ),
      padding: const EdgeInsets.symmetric(
        horizontal: 20,
        vertical: 12,
      ),
    ),
    icon: const Icon(Icons.lock_reset, color: Colors.white),
    label: Text(
      'Reset Password',
      style: GoogleFonts.lato(
        color: Colors.white,
        fontSize: 16,
        fontWeight: FontWeight.w600,
      ),
    ),
  ),
  const SizedBox(height: 20),
  const LogOutWidget(),
],
```

```
          ),
        ),
      ),
    );
  }
}
```

ScreenShots: