MPL EXPERIMENT-3

Name: Ansh Sarfare **Class/Roll No :** D15A-49

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 user friendly.
- **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):

```
Icon(
 Icons.home,
 size: 40,
);
```

Adding Images in Flutter
Flutter supports images from three primary sources: Assets, Network, and Local Storage (Memory or File
System).

1. Using Asset Images (Local Project Files)
To use an image stored in the project folder:

Place the image inside the assets/images/ folder.
Declare it in pubspec.yaml:
flutter:
 assets:
 - assets/images/sample.png
Display it in the app:
Image.asset('assets/images/sample.png');

2. Using Network Images (Fetched from the Internet)
Flutter simplifies loading images from the web using Image.network. Additional properties like height,
width, fit, and color can be specified.
Example:
Image.network('https://example.com/sample.jpg');

3. Using Local Storage (Memory or File System)
Images stored on the user's device can also be displayed using packages like image_picker or file_picker.
Adding Custom Fonts in Flutter
By default, Flutter uses the Roboto font. However, custom fonts can be added to create a unique visual
Identity.

Steps to Add a Custom Font:
1. Download the font and place it in the assets/fonts/ folder.

2. Declare the font in pubspec.yaml:
yaml
flutter:
 fonts:
 - family: CustomFont
 fonts:
 - asset: assets/fonts/CustomFont.ttf

3. Use the font in your app
Text(
 'Custom Font Example',
 style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),
);

Code:

Profile_page:

```
import 'package:flutter/material.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';
import
'package:firebase_auth/firebase_auth.dart';
import
'../services/firebase_auth_service.dart';
import
'../screens/liked_recipes_screen.dart';
import
'../screens/saved_recipes_screen.dart';
import '../screens/orders_page.dart';
import '../screens/address_page.dart';
import
'../screens/community_post_page.dart'; //
Import CommunityPostsPage

class ProfilePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Container(
          alignment: Alignment.topLeft,
          child: const Text("Profile"),
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.location_on,
color: Colors.white),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => AddressPage()),
              );
            },
          ),
          IconButton(
            icon: const Icon(Icons.image_sharp,
color: Colors.white), // Community Posts
icon
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => CommunityPostsPage()),
              );
            },
          ),
          IconButton(
            icon: const Icon(Icons.logout, color:
Colors.white),
            onPressed: () async {
              final authService = AuthService();
              await authService.signOut();

Navigator.pushReplacementNamed(context,
'/login');
            },
          ),
        ],
      ),
      body:
FutureBuilder<DocumentSnapshot>(
        future:
FirebaseFirestore.instance.collection('users'
).doc(FirebaseAuth.instance.currentUser!.ui
d).get(),
        builder: (BuildContext context,
AsyncSnapshot<DocumentSnapshot>
snapshot) {
          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return const Center(child:
CircularProgressIndicator());
          }
          if (!snapshot.hasData ||
!snapshot.data!.exists) {
```

```
          return const Center(child: Text("No
data found"));
        }

        var userData = snapshot.data!.data()
as Map<String, dynamic>;
        String username =
userData['username'] ?? 'User';

        return SingleChildScrollView(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            children: [
              Text(
                'Hello, $username',
                style: const TextStyle(fontSize:
20, fontWeight: FontWeight.bold),
                textAlign: TextAlign.center,
              ),
              const SizedBox(height: 20),
              GestureDetector(
                onTap: () {
                  Navigator.push(context,
MaterialPageRoute(builder: (context) =>
LikedRecipesScreen()));
                },
                child: Container(
                  height: 150,
                  width: double.infinity,
                  decoration: BoxDecoration(
                    image: const
DecorationImage(
                      image:
AssetImage('assets/liked_recipes.png'),
                      fit: BoxFit.cover,
                    ),
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
              ),
              const SizedBox(height: 20),
              GestureDetector(
                onTap: () {
                  Navigator.push(context,
MaterialPageRoute(builder: (context) =>
SavedRecipesScreen()));
                },
                child: Container(
                  height: 150,
                  width: double.infinity,
                  decoration: BoxDecoration(
                    image: const
DecorationImage(
                      image:
AssetImage('assets/saved_recipes.png'),
                      fit: BoxFit.cover,
                    ),
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
              ),
              const SizedBox(height: 20),
              GestureDetector(
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
OrdersPage(),
                    ),
                  );
                },
                child: Container(
                  height: 150,
                  width: double.infinity,
                  decoration: BoxDecoration(
                    image: const
DecorationImage(
                      image:
AssetImage('assets/orders.png'),
                      fit: BoxFit.cover,
                    ),
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
```

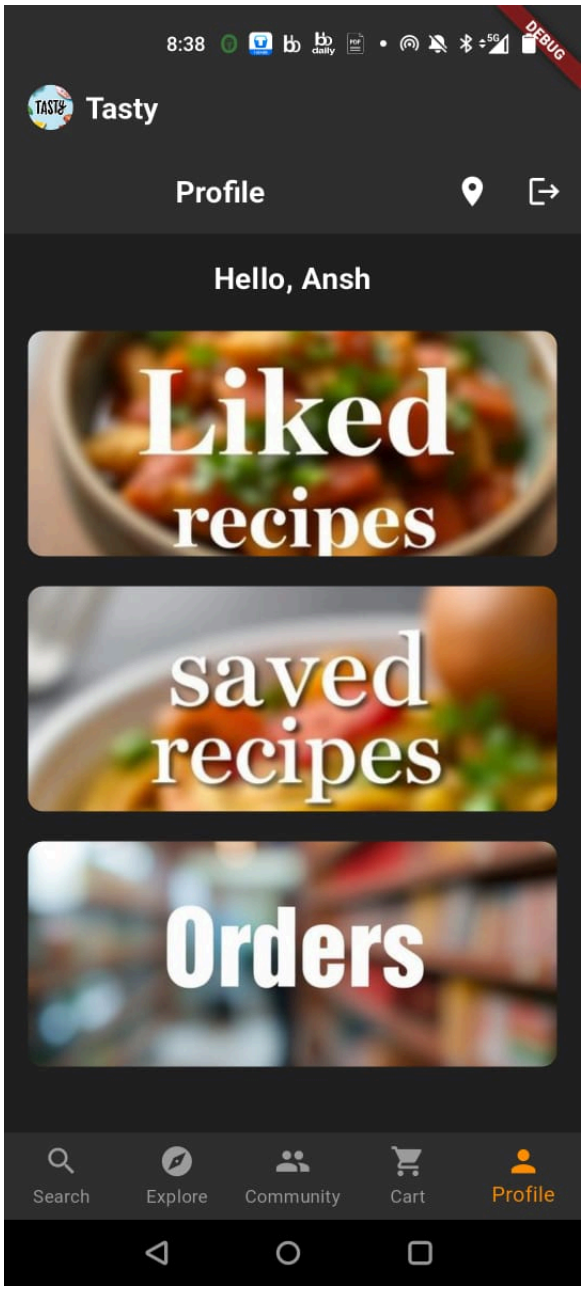
Explore_screen:

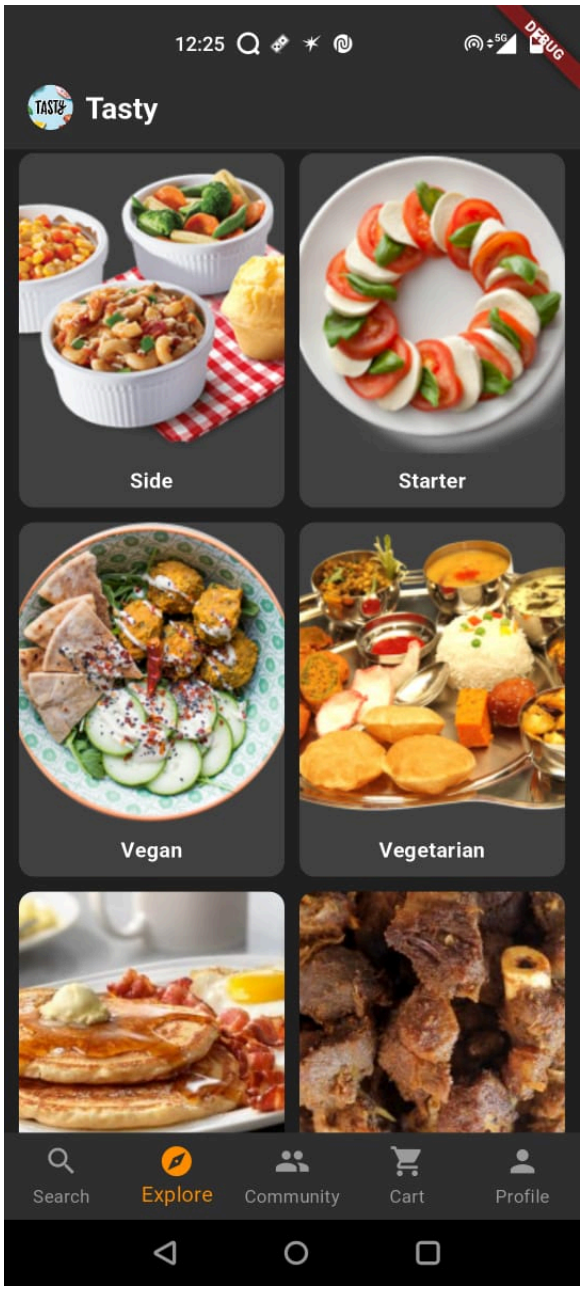
```
import 'package:flutter/material.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'widgets/recipe_detail_screen.dart';
import 'models/recipe.dart';

class ExploreScreen extends
StatefulWidget {
  @override
  _ExploreScreenState createState() =>
_ExploreScreenState();
}
class _ExploreScreenState extends
State<ExploreScreen> {
  List<Category> categories = [];
  bool _isLoading = true;

  @override
  void initState() {
    super.initState();
    fetchCategories();
  }
  Future<void> fetchCategories() async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/categories.php'));
    if (response.statusCode == 200) {
      final data =
json.decode(response.body);
      if (data['categories'] != null) {
        setState(() {
          categories = (data['categories'] as
List).map((json) =>
Category.fromJson(json)).toList();
          _isLoading = false;
        });
      }
    } else {
      print('Failed to load categories');
      setState(() {
        _isLoading = false;
  }

  Future<List<Recipe>>
fetchCategoryRecipes(String
categoryName) async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/filter.php?c=$categoryName
'));
    if (response.statusCode == 200) {
      final data =
json.decode(response.body);
      if (data['meals'] != null) {
        return (data['meals'] as List).map((json)
=> Recipe.fromJson(json)).toList();
      } else {
        return [];
      }
    } else {
      print('Failed to load recipes for category:
$categoryName');
      return [];
    }

  Future<Recipe?>
fetchRecipeDetails(String recipeName)
async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/search.php?s=$recipeName
'));
    if (response.statusCode == 200) {
      final data =
json.decode(response.body);
      if (data['meals'] != null &&
data['meals'].isNotEmpty) {
        return
Recipe.fromJson(data['meals'][0]);
      }
    } else {
      print('Failed to load recipe details for
$recipeName');
    }
    return null;
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[900],
      body: _isLoading
          ? Center(child:
CircularProgressIndicator())
          : GridView.builder(
        padding: EdgeInsets.all(10.0),
        gridDelegate:
SliverGridDelegateWithFixedCrossAxisCou
nt(
          crossAxisCount: 2,
          crossAxisSpacing: 10,
          mainAxisSpacing: 10,
          childAspectRatio: 0.75,
        ),
        itemCount: categories.length,
        itemBuilder: (context, index) {
          final category = categories[index];
          return InkWell(
            onTap: () async {
              List<Recipe> recipes = await
fetchCategoryRecipes(category.strCategory
);
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) =>
CategoryRecipeScreen(
                    categoryName:
category.strCategory,
                    recipes: recipes,
                  ),
            child: Container(
              decoration: BoxDecoration(
                color: Colors.grey[800],
                borderRadius:
BorderRadius.circular(10),
              ),
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.stretch,
                children: [
                  Expanded(
                    child: ClipRRect(
                      borderRadius:
BorderRadius.vertical(top:
Radius.circular(10)),
                      child: Image.network(
                        category.strCategoryThumb,
                        fit: BoxFit.cover,
                        errorBuilder: (context, error,
stackTrace) {
                          return Container(
                            color: Colors.grey[300],
                            child: Center(child:
Icon(Icons.error_outline)),
                          );
                        },
                      ),
                    ),
                  ),
                  Padding(
                    padding: const
EdgeInsets.all(8.0),
                    child: Text(
                      category.strCategory,
                      style: TextStyle(
                        color: Colors.white,
                        fontWeight: FontWeight.bold,
                      ),
                      textAlign: TextAlign.center,
                    ),
                  ),
class CategoryRecipeScreen extends
StatelessWidget {
  final String categoryName;
  final List<Recipe> recipes;

  CategoryRecipeScreen({required
this.categoryName, required this.recipes});

  Future<Recipe?>
fetchRecipeDetails(String recipeName)
async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/search.php?s=$recipeName
'));
    if (response.statusCode == 200) {
```

```
    final data =
json.decode(response.body);
    if (data['meals'] != null &&
data['meals'].isNotEmpty) {
      return
Recipe.fromJson(data['meals'][0]);
    }
  } else {
    print('Failed to load recipe details for
$recipeName');
  }
  return null;
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text(categoryName),
   ),
   backgroundColor: Colors.grey[900],
   body: recipes.isEmpty
      ? Center(
     child: Text(
      'No recipes found for this category.',
      style: TextStyle(color: Colors.white),
     ),
   )
      : ListView.builder(
     itemCount: recipes.length,
     itemBuilder: (context, index) {
      final recipe = recipes[index];
      return Card(
       color: Colors.grey[800],
       child: InkWell(
        onTap: () async {
         Recipe? detailedRecipe = await
fetchRecipeDetails(recipe.strMeal);
         if (detailedRecipe != null) {
          Navigator.push(
           context,
           MaterialPageRoute(
            builder: (context) =>
RecipeDetailScreen(recipe:
detailedRecipe),
         } else {
          print('Failed to load detailed
recipe');
        },
 {
          return Container(
           width: double.infinity,
           height: 250,
           color: Colors.grey[300],
           child: Center(
            child:
Icon(Icons.error_outline),
           ),
          SizedBox(height: 8),
          Align(
           alignment: Alignment.center,
           child: Text(
            recipe.strMeal,
            style: TextStyle(
             color: Colors.white,
             fontSize: 18,
             fontWeight:
FontWeight.bold,
           ),
}
class Category {
 final String strCategory;
 final String strCategoryThumb;
 final String idCategory;

 Category({
  required this.strCategory,
  required this.strCategoryThumb,
  required this.idCategory,
 });

 factory Category.fromJson(Map<String,
dynamic> json) {
  return Category(
   strCategory: json['strCategory'],
   strCategoryThumb:
json['strCategoryThumb'],
    idCategory: json['idCategory'],
  );
```

Output:


8:38
DEBUG
Tasty
Profile
Hello, Ansh
Liked recipes
saved recipes
Orders
Search
Explore
Community
Cart
Profile


12:25
DEBUG
Tasty
Side
Starter
Vegan
Vegetarian
Search
Explore
Community
Cart
Profile