MPL EXPERIMENT-5

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

Aim: To apply navigation, routing and gestures in Flutter App.

Theory: -

Navigation, Routing, and Gesture Handling in Flutter
In Flutter, screens or pages are referred to as routes, and each route is essentially a widget. This concept is similar to Activities in Android. Navigating between pages defines an app's workflow, and the mechanism for handling this is known as routing.
Flutter provides a built-in routing system using MaterialPageRoute, along with the Navigator.push() and Navigator.pop() methods to move between routes.
Additionally, gestures allow apps to respond to user interactions like taps, swipes, and drags, making applications more dynamic and user-friendly.

Navigation and Routing in Flutter,

1. Using the Navigator Widget

Flutter's Navigator widget manages a stack of routes, enabling seamless navigation between screens.
Pushing a Route: Moves to a new screen using Navigator.push().
Popping a Route: Returns to the previous screen using Navigator.pop().

Example:

```
ElevatedButton(
 onPressed: () {
 Navigator.push(
 context,
 MaterialPageRoute(builder: (context) => SecondScreen()),
 );
 },
 child: Text('Go to Second Screen'),
);
```

2. Using Named Routes

For larger applications, named routes provide a cleaner and more structured way to manage navigation.
Step 1: Define Routes in MaterialApp

```
MaterialApp(
```

```
 initialRoute: '/',
 routes: {
 '/': (context) => HomeScreen(),
 '/second': (context) => SecondScreen(),
 },
);
```

Step 2: Navigate Using Navigator.pushNamed()

```
Navigator.pushNamed(context, '/second');
```

Handling Gestures in Flutter

Gestures enable user interaction through taps, swipes, pinches, and drags. Flutter provides various widgets and gesture detectors to manage these interactions effectively.

1. Tap Gestures

Taps are one of the most common interactions and can be handled using:

GestureDetector

InkWell

ElevatedButton

Example (Tap Gesture using GestureDetector):

```
GestureDetector(
 onTap: () {
 print("Tapped!");
 },
 child: Container(
 padding: EdgeInsets.all(20),
 color: Colors.blue,
 child: Text('Tap Me'),
 ),
);
```

2. Long Press Gestures

Long-press interactions can be captured using the onLongPress callback in GestureDetector or InkWell.

```
InkWell(
 onLongPress: () {
 print("Long Pressed!");
 },
 child: Container(
 padding: EdgeInsets.all(20),
 color: Colors.red,
 child: Text('Long Press Me'),
 ),
);
```

3. Swipe and Drag Gestures

Flutter provides built-in methods like onHorizontalDragUpdate and onVerticalDragUpdate to detect
swipe and drag actions.
Example (Swipe Detection):

```
GestureDetector(
 onHorizontalDragUpdate: (details) {
 if (details.primaryDelta! > 0) {
 print("Swiped Right!");
 } else {
 print("Swiped Left!");
 }
 },
 child: Container(
 padding: EdgeInsets.all(20),
 color: Colors.green,
 child: Text('Swipe Me'),
 ),
);
```

Code:

Cart page:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'cart_provider.dart';
import 'personal_cart.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';
import '../models/cart_model.dart';

class CartPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Cart'),
        actions: [
          IconButton(
            icon: Icon(Icons.shopping_cart,
color: Colors.white),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => PersonalCartPage()),
              );
            },
          ),
        ],
```

```
      ),
      body: StreamBuilder<QuerySnapshot>(
        stream:
FirebaseFirestore.instance.collection('cart').
snapshots(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return Center(child:
CircularProgressIndicator());
          }

          return GridView.builder(
            padding: EdgeInsets.all(10),
            gridDelegate:
SliverGridDelegateWithFixedCrossAxisCou
nt(
              crossAxisCount: 2,
              crossAxisSpacing: 10,
              mainAxisSpacing: 10,
              childAspectRatio: 1,
            ),
            itemCount:
snapshot.data!.docs.length,
            itemBuilder: (context, index) {
              var category =
snapshot.data!.docs[index];
              return CategoryCard(category:
category);
            },
          );
        },
      ),
    );
  }
}

class CategoryCard extends
StatelessWidget {
  final QueryDocumentSnapshot category;

  CategoryCard({required this.category});

  @override
  Widget build(BuildContext context) {
    String categoryName = category.id;
    String imagePath =
'assets/${categoryName}.png'; // Category
image should be named categoryName.png
and present in assets.

    return GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) =>
ItemList(categoryName: categoryName),
          ),
        );
      },
      child: Card(
        elevation: 5,
        child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            Image.asset(
              imagePath,
              width: 130, // Increased image
width
              height: 130, // Increased image
height
              fit: BoxFit.cover,
            ),
            SizedBox(height: 10),
            Text(
              categoryName.toUpperCase(),
              style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.black), //
Black text
            ),
          ],
        ),
      ),
    );
  }
}
```

```
class ItemList extends StatelessWidget {
  final String categoryName;

  ItemList({required this.categoryName});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title:
Text('${categoryName.toUpperCase()}'),
      ),
      body:
StreamBuilder<DocumentSnapshot>(
        stream:
FirebaseFirestore.instance.collection('cart').
doc(categoryName).snapshots(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return Center(child:
CircularProgressIndicator());
          }

          var categoryData =
snapshot.data!.data() as Map<String,
dynamic>;
          List<dynamic> items =
categoryData['items'];

          return ListView.builder(
            itemCount: items.length,
            itemBuilder: (context, index) {
              Map<String, dynamic> item =
items[index];
              return ItemCard(item: item,
categoryName: categoryName);
            },
          );
        },
      ),
    );
  }
}
```

```
class ItemCard extends StatelessWidget {
  final Map<String, dynamic> item;
  final String categoryName;

  ItemCard({required this.item, required
this.categoryName});

  @override
  Widget build(BuildContext context) {
    final cartProvider =
Provider.of<CartProvider>(context);
    return Card(
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Row(
          children: [
            Image.network(
              item['imageURL'],
              width: 100,
              height: 100,
              fit: BoxFit.cover,
            ),
            SizedBox(width: 10),
            Expanded(
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  Text(
                    item['name'],
                    style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold, color:
Colors.black), // Black text
                  ),
                  Text(

'\₹${item['price'].toStringAsFixed(2)}',
                    style: TextStyle(color:
Colors.black), // Black text
                  ),
                ],
              ),
            ),
```

```
            QuantityControl(item: item,
cartProvider: cartProvider),
          ],
        ),
      ),
    );
  }
}

class QuantityControl extends
StatefulWidget {
  final Map<String, dynamic> item;
  final CartProvider cartProvider;

  QuantityControl({required this.item,
required this.cartProvider});

  @override
  _QuantityControlState createState() =>
_QuantityControlState();
}

class _QuantityControlState extends
State<QuantityControl> {
  late int quantity;

  @override
  void initState() {
    super.initState();
    // Initialize quantity based on what's in the
cart
    quantity =
widget.cartProvider.getQuantity(widget.item[
'name']);
  }

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        IconButton(
          icon: Icon(Icons.remove, color:
Colors.black), // Black icon
          onPressed: quantity > 0
              ? () {
            setState(() {
              quantity--;
            });
            widget.cartProvider.removeItem(
              CartItem(
                name: widget.item['name'],
                imageURL:
widget.item['imageURL'],
                price:
widget.item['price'].toDouble(),
                quantity: 1,
              ),
            );
          }
              : null,
        ),
        Text('$quantity', style: TextStyle(color:
Colors.black)), // Black text
        IconButton(
          icon: Icon(Icons.add, color:
Colors.black), // Black icon
          onPressed: () {
            setState(() {
              quantity++;
            });
            widget.cartProvider.addItem(
              CartItem(
                name: widget.item['name'],
                imageURL:
widget.item['imageURL'],
                price:
widget.item['price'].toDouble(),
                quantity: 1,
              ),
            );
          },
        ),
      ],
    );
  }
}
```

Community page:

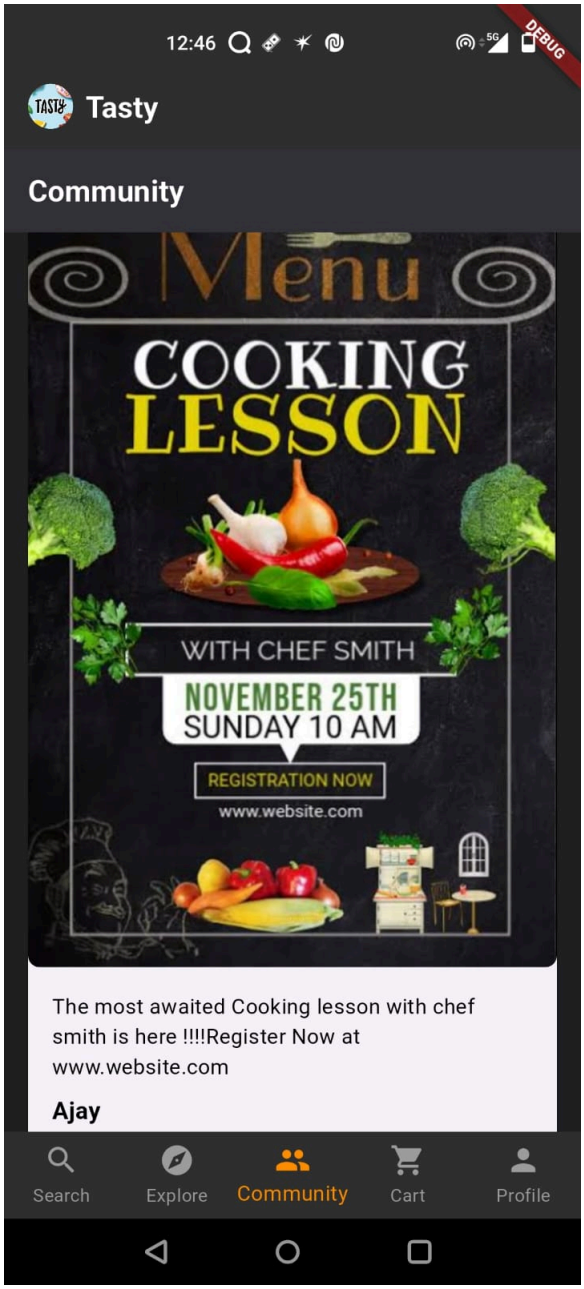
```
import 'package:flutter/material.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';
import
'package:firebase_auth/firebase_auth.dart';
import 'dart:io';
import '../services/cloudinary_service.dart';

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

  @override
  _CommunityPageState createState() =>
_CommunityPageState();
}

class _CommunityPageState extends
State<CommunityPage> {
  final _formKey = GlobalKey<FormState>();
  final _captionController =
TextEditingController();
  File? _image;
  bool _uploading = false;

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

  @override
  void dispose() {
    _captionController.dispose();
    super.dispose();
  }

  Future<void> _pickImage() async {
    if (!mounted) return;

    try {
      final pickedImage = await
CloudinaryService.pickImage();
      if (pickedImage != null && mounted) {
        setState(() {
          _image = pickedImage;
        });
      }
    } catch (e) {
      if (!mounted) return;

ScaffoldMessenger.of(context).showSnackB
ar(
        const SnackBar(content: Text('Failed to
pick image')),
      );
    }
  }

  Future<void> _uploadPost() async {
    if (!_formKey.currentState!.validate() ||
_image == null) {
      return;
    }

    if (!mounted) return;
    setState(() {
      _uploading = true;
    });

    try {
      // Upload image to Cloudinary
      final imageUrl = await
CloudinaryService.uploadImage(_image!);
      if (imageUrl == null) {
        throw Exception('Failed to upload
image');
      }

      // Get current user
```

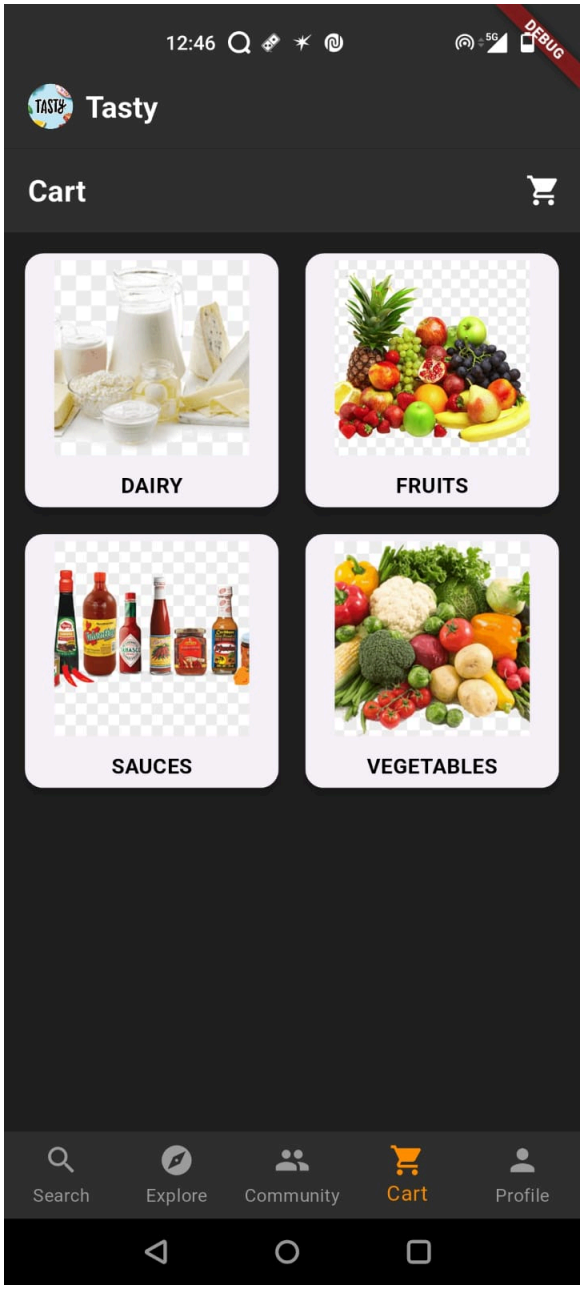
```
      final user =
FirebaseAuth.instance.currentUser;
      if (user == null) {
        throw Exception('User not logged in');
      }

      // Get user data from Firestore
      final userData = await
FirebaseFirestore.instance
          .collection('users')
          .doc(user.uid)
          .get();

      // Create post in Firestore
      await
FirebaseFirestore.instance.collection('comm
unityPosts').add({
        'userId': user.uid,
        'username': userData['username'],
        'imageUrl': imageUrl,
        'caption': _captionController.text.trim(),
        'timestamp':
FieldValue.serverTimestamp(),
      });

      if (!mounted) return;

      // Clear form
      _captionController.clear();
      setState(() {
        _image = null;
      });


ScaffoldMessenger.of(context).showSnackB
ar(
        const SnackBar(content: Text('Post
uploaded successfully!')),
      );
    } catch (e) {
      print('Error uploading post: $e');
      if (!mounted) return;
ScaffoldMessenger.of(context).showSnackB
ar(
        const SnackBar(content: Text('Failed to
upload post')),
      );
    } finally {
      if (!mounted) return;
      setState(() {
        _uploading = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Community'),
      ),
      body: SingleChildScrollView(
        key: const
PageStorageKey('community_scroll'),
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            Form(
              key: _formKey,
              child: Column(
                children: [
                  if (_image != null)
                    ClipRRect(
                      borderRadius:
BorderRadius.circular(8.0),
                      child: Image.file(
                        _image!,
                        height: 200,
                        width: double.infinity,
                        fit: BoxFit.cover,
                      ),
                    )
                  else
                    OutlinedButton(
```

```
                onPressed: _uploading ? null :
_pickImage,
                child: const Text('Select
Image'),
              ),
            const SizedBox(height: 16),

            TextFormField(
              controller: _captionController,
              enabled: !_uploading,
              decoration: const
InputDecoration(
                labelText: 'Caption',
                border: OutlineInputBorder(),
              ),
              maxLines: 3,
              style: const TextStyle(color:
Colors.white),
              validator: (value) {
                if (value == null ||
value.trim().isEmpty) {
                  return 'Please enter a
caption';
                }
                return null;
              },
            ),
            const SizedBox(height: 16),

            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: _uploading ? null :
_uploadPost,
                child: _uploading
                    ? const SizedBox(
                      height: 20,
                      width: 20,
                      child:
CircularProgressIndicator(strokeWidth: 2),
                    )
                    : const Text('Upload Post'),
              ),
            ),
          ],
        ),
      ),
      const SizedBox(height: 32),
      StreamBuilder<QuerySnapshot>(
        stream: FirebaseFirestore.instance
            .collection('communityPosts')
            .orderBy('timestamp',
descending: true)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Center(
              child: Text('Error:
${snapshot.error}'),
            );
          }

          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return const Center(
              child:
CircularProgressIndicator(),
            );
          }

          // Get the documents or an empty
list if they are null
          final posts = snapshot.data?.docs
?? [];

          if (posts.isEmpty) {
            return const Center(
              child: Text('No posts yet!'),
            );
          }

          return ListView.builder(
            key: const
PageStorageKey('community_posts'),
            shrinkWrap: true,
            physics: const
NeverScrollableScrollPhysics(),
            itemCount: posts.length,
```

```
            itemBuilder: (context, index) {
              final post = posts[index];
              final data = post.data() as
Map<String, dynamic>;

              return Card(
                key: ValueKey(post.id),
                margin: const
EdgeInsets.only(bottom: 16.0),
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    ClipRRect(
                      borderRadius:
BorderRadius.circular(8.0),
                      child: Image.network(
                        data['imageUrl'],
                        width: double.infinity,
                        fit: BoxFit.contain, //
Changed to BoxFit.contain
                        loadingBuilder: (context,
child, loadingProgress) {
                          if (loadingProgress ==
null) return child;
                          return SizedBox(
                            height: 300,
                            child: Center(
                              child:
CircularProgressIndicator(
                                value:
loadingProgress.expectedTotalBytes != null
                                    ?
loadingProgress.cumulativeBytesLoaded /

loadingProgress.expectedTotalBytes!
                                    : null,
                              ),
                            ),
                          );
                        },
                        errorBuilder: (context,
error, stackTrace) {
                          return const SizedBox(
                            height: 300,
                            child: Center(
                              child:
Icon(Icons.error_outline, size: 40),
                            ),
                          );
                        },
                      ),
                    ),
                    Padding(
                      padding: const
EdgeInsets.all(16.0),
                      child: Column(
                        crossAxisAlignment:
CrossAxisAlignment.start,
                        children: [
                          Text(
                            data['caption'] ?? '',
                            style: const
TextStyle(color: Colors.black),
                          ),
                          const SizedBox(height:
8),
                          Text(
                            data['username'] ??
'Unknown User',
                            style: const TextStyle(
                              fontWeight:
FontWeight.bold,
                              fontSize: 16,
                              color: Colors.black,
                            ),
                          ),
                        ],
                      ),
                    ),
                  ],
                ),
              );
            },
          ),
        ],
```

Output:


12:46
DEBUG
Tasty
Community
Menu
COOKING
LESSON
WITH CHEF SMITH
NOVEMBER 25TH
SUNDAY 10 AM
REGISTRATION NOW
www.website.com
The most awaited Cooking lesson with chef smith is here !!!!Register Now at www.website.com
Ajay
Search
Explore
Community
Cart
Profile


12:46
DEBUG
Tasty
Cart
DAIRY
FRUITS
SAUCES
VEGETABLES
Search
Explore
Community
Cart
Profile

12:47

Your Orders

Order Placed On: 2025-02-17 – 22:55

Deliver To:
flat
406,Navjeevan, Chembur
Mumbai, Maharashtra - 400007

Items:

banana
Quantity: 1, Price: ₹30.0

butter
Quantity: 1, Price: ₹60.0

Total: ₹90.0
Payment Method: COD

Order Placed On: 2025-02-17 – 21:55

Deliver To:
flat
406,Navjeevan, Chembur
Mumbai, Maharashtra - 400007

Items:

paneer
Quantity: 1, Price: ₹90.0

banana
Quantity: 1, Price: ₹30.0

carrot

12:50

Saved Recipes

Blini Pancakes

Thai Green Curry