Name: Yukta Bhatia

RollNo: 07

Class: D15A

Experiment-5

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

Theory:

Navigation:

Navigation is a fundamental aspect of mobile app development that involves transitioning between different screens or pages. In Flutter, the Navigator class plays a central role in managing the navigation stack, allowing developers to push and pop routes as users move through the app.

Navigator Class:

The Navigator class handles the navigation stack, maintaining a history of routes. The push method adds a new route to the stack, typically triggered by user actions. The pop method removes the current route from the stack, enabling backward navigation.

Routing:

Routing in Flutter involves defining and organizing the paths or routes within the application. Routes represent different screens or pages, and their effective use is crucial for structuring the app's architecture.

MaterialPageRoute and CupertinoPageRoute:

MaterialPageRoute is employed in apps following Material Design principles, providing a standard Android-style transition between screens.

CupertinoPageRoute is used for iOS-style designs, ensuring a consistent and native user experience.

Gestures in Flutter:

Gestures enhance user interaction by allowing the app to respond to touch or mouse inputs. In Flutter, the GestureDetector widget is instrumental in recognizing and handling various gestures.

Code:

home_screen.dart

```
import 'dart:developer';
```

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import '../api/apis.dart';
import '../helper/dialogs.dart';
import '../main.dart';
import '../models/chat_user.dart';
import '../widgets/chat_user_card.dart';
import 'profile_screen.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});
  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<ChatUser> _list = [];
  final List<ChatUser> _searchList = [];
  bool _isSearching = false;
  @override
  void initState() {
    super.initState();
    APIs.getSelfInfo();
    SystemChannels.lifecycle.setMessageHandler((message) {
      log('Message: $message');
      if (APIs.auth.currentUser != null) {
        if (message.toString().contains('resume')) {
          APIs.updateActiveStatus(true);
        }
        if (message.toString().contains('pause')) {
          APIs.updateActiveStatus(false);
        }
      }
```

```
    return Future.value(message);
  });
}
@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusScope.of(context).unfocus(),
    child: WillPopScope(
      onWillPop: () {
        if (_isSearching) {
          setState(() {
            _isSearching = !_isSearching;
          });
          return Future.value(false);
        } else {
          return Future.value(true);
        }
      },
      child: Scaffold{
        appBar: AppBar(
          leading: const Icon(CupertinoIcons.home),
          title: _isSearching
              ? TextField(
                  decoration: const InputDecoration(
                      border: InputBorder.none, hintText: 'Name, Email, ...'),
                  autofocus: true,
                  style: const TextStyle(fontSize: 17, letterSpacing: 0.5),
                  onChanged: (val) {
                    _searchList.clear();

                    for (var i in _list) {
                      if (i.name.toLowerCase().contains(val.toLowerCase()) ||
                          i.email.toLowerCase().contains(val.toLowerCase())) {
                        _searchList.add(i);
```

```
                setState(() {
                  _searchList;
                });
              }
            }
          },
        )
      : const Text('Quick Chats'),
  actions: [
    IconButton(
      onPressed: () {
        setState(() {
          _isSearching = !_isSearching;
        });
      },
      icon: Icon(_isSearching
          ? CupertinoIcons.clear_circled_solid
          : Icons.search)),
    IconButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (_) => ProfileScreen(user: APIs.me)));
      },
      icon: const Icon(Icons.more_vert))
  ],
),
floatingActionButton: Padding(
  padding: const EdgeInsets.only(bottom: 10),
  child: FloatingActionButton(
    onPressed: () {
      _addChatUserDialog();
    },
```

```
      child: const Icon(Icons.add_comment_rounded)),
),
//body
body: StreamBuilder(
  stream: APIs.getMyUsersId(),
  builder: (context, snapshot) {
    switch (snapshot.connectionState) {
      case ConnectionState.waiting:
      case ConnectionState.none:
        return const Center(child: CircularProgressIndicator());

      case ConnectionState.active:
      case ConnectionState.done:
        return StreamBuilder(
          stream: APIs.getAllUsers(
              snapshot.data?.docs.map((e) => e.id).toList() ?? []),

          builder: (context, snapshot) {
            switch (snapshot.connectionState) {

              case ConnectionState.waiting:
              case ConnectionState.none:
              case ConnectionState.active:
              case ConnectionState.done:
                final data = snapshot.data?.docs;
                _list = data
                        ?.map((e) => ChatUser.fromJson(e.data()))
                        .toList() ??
                    [];

                if (_list.isNotEmpty) {
                  return ListView.builder(
                      itemCount: _isSearching
                          ? _searchList.length
```

```
                    : _list.length,
                padding: EdgeInsets.only(top: mq.height * .01),
                physics: const BouncingScrollPhysics(),
                itemBuilder: (context, index) {
                  return ChatUserCard(
                      user: _isSearching
                          ? _searchList[index]
                          : _list[index]);
                });
            } else {
              return const Center(
                child: Text('No Connections Found!',
                    style: TextStyle(fontSize: 20)),
              );
            }
          },
        );
      }
    ),
  ),
 ),
);
}
void _addChatUserDialog() {
 String email = '';
 showDialog(
    context: context,
    builder: (_) => AlertDialog(
        contentPadding: const EdgeInsets.only(
            left: 24, right: 24, top: 20, bottom: 10),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(20)),
```

```
title: Row(
  children: const [
    Icon(
      Icons.person_add,
      color: Colors.blue,
      size: 28,
    ),
    Text('  Add User')
  ],
),
content: TextFormField(
  maxLines: null,
  onChanged: (value) => email = value,
  decoration: InputDecoration(
      hintText: 'Email Id',
      prefixIcon: const Icon(Icons.email, color: Colors.blue),
      border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(15))),
),
actions: [
  MaterialButton(
      onPressed: () {
        Navigator.pop(context);
      },
      child: const Text('Cancel',
          style: TextStyle(color: Colors.blue, fontSize: 16))),
  MaterialButton(
      onPressed: () async {
        //hide alert dialog
        Navigator.pop(context);
        if (email.isNotEmpty) {
          await APIs.addChatUser(email).then((value) {
            if (!value) {
              Dialogs.showSnackbar(
```

```
                    context, 'User does not Exists!');
                }
              });
            }
          },
          child: const Text(
            'Add',
            style: TextStyle(color: Colors.blue, fontSize: 16),
          ))
      ],
    ));
  }
}
```

chat_user_card.dart

```
import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import '../api/apis.dart';
import '../helper/my_date_util.dart';
import '../main.dart';
import '../models/chat_user.dart';
import '../models/message.dart';
import '../screens/chat_screen.dart';
import 'dialogs/profile_dialog.dart';

//card to represent a single user in home screen
class ChatUserCard extends StatefulWidget {
  final ChatUser user;
  const ChatUserCard({super.key, required this.user});
  @override
  State<ChatUserCard> createState() => _ChatUserCardState();
}
class _ChatUserCardState extends State<ChatUserCard> {
  Message? _message;
```

```
@override
Widget build(BuildContext context) {
  return Card(
    margin: EdgeInsets.symmetric(horizontal: mq.width * .04, vertical: 4),
    elevation: 0.5,
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
    child: InkWell(
      onTap: () {
        //for navigating to chat screen
        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (_) => ChatScreen(user: widget.user)));
      },
      child: StreamBuilder(
        stream: APIs.getLastMessage(widget.user),
        builder: (context, snapshot) {
          final data = snapshot.data?.docs;
          final list =
              data?.map((e) => Message.fromJson(e.data())).toList() ?? [];
          if (list.isNotEmpty) _message = list[0];
          return ListTile{
            leading: InkWell(
              onTap: () {
                showDialog(
                    context: context,
                    builder: (_) => ProfileDialog(user: widget.user));
              },
              child: ClipRRect(
                borderRadius: BorderRadius.circular(mq.height * .03),
                child: CachedNetworkImage(
                  width: mq.height * .055,
                  height: mq.height * .055,
                  imageUrl: widget.user.image,
```

```
        errorWidget: (context, url, error) => const CircleAvatar(
            child: Icon(CupertinoIcons.person)),
      ),
    ),
  ),
  title: Text(widget.user.name),
  subtitle: Text(
    _message != null
      ? _message!.type == Type.image
        ? 'image'
        : _message!.msg
      : widget.user.about,
    maxLines: 1),

  trailing: _message == null
    ? null
    : _message!.read.isEmpty &&
          _message!.fromId != APIs.user.uid
      ?
      Container(
        width: 15,
        height: 15,
        decoration: BoxDecoration(
          color: Colors.greenAccent.shade400,
          borderRadius: BorderRadius.circular(10)),
      )
      :
      Text(
        MyDateUtil.getLastMessageTime(
          context: context, time: _message!.sent),
        style: const TextStyle(color: Colors.black54),
      ),
  );
},
```

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

chat_screen.dart

```
import 'dart:developer';
import 'dart:io';
import 'package:cached_network_image/cached_network_image.dart';
import 'package:emoji_picker_flutter/emoji_picker_flutter.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import '../api/apis.dart';
import '../helper/my_date_util.dart';
import '../main.dart';
import '../models/chat_user.dart';
import '../models/message.dart';
import '../widgets/message_card.dart';
import 'view_profile_screen.dart';

class ChatScreen extends StatefulWidget {
  final ChatUser user;
  const ChatScreen({super.key, required this.user});
  @override
  State<ChatScreen> createState() => _ChatScreenState();
}
class _ChatScreenState extends State<ChatScreen> {
  List<Message> _list = [];
  final _textController = TextEditingController();
  bool _showEmoji = false, _isUploading = false;
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusScope.of(context).unfocus(),
```

```
child: SafeArea(
  child: WillPopScope(
    onWillPop: () {
      if (_showEmoji) {
        setState(() => _showEmoji = !_showEmoji);
        return Future.value(false);
      } else {
        return Future.value(true);
      }
    },
    child: Scaffold(
      //app bar
      appBar: AppBar(
        automaticallyImplyLeading: false,
        flexibleSpace: _appBar(),
      ),
      backgroundColor: const Color.fromARGB(255, 234, 248, 255),
      body: Column(
        children: [
          Expanded(
            child: StreamBuilder(
              stream: APIs.getAllMessages(widget.user),
              builder: (context, snapshot) {
                switch (snapshot.connectionState) {
                  //if data is loading
                  case ConnectionState.waiting:
                  case ConnectionState.none:
                    return const SizedBox();
                  case ConnectionState.active:
                  case ConnectionState.done:
                    final data = snapshot.data?.docs;
                    _list = data
                            ?.map((e) => Message.fromJson(e.data()))
                            .toList() ??
```

```
            [];

        if (_list.isNotEmpty) {
          return ListView.builder(
            reverse: true,
            itemCount: _list.length,
            padding: EdgeInsets.only(top: mq.height * .01),
            physics: const BouncingScrollPhysics(),
            itemBuilder: (context, index) {
              return MessageCard(message: _list[index]);
            });
        } else {
          return const Center(
            child: Text('Say Hii! 👋',
                style: TextStyle(fontSize: 20)),
          );
        }
      }
    },
  ),
),
if (_isUploading)
  const Align(
      alignment: Alignment.centerRight,
      child: Padding(
          padding:
              EdgeInsets.symmetric(vertical: 8, horizontal: 20),
          child: CircularProgressIndicator(strokeWidth: 2))),
_chatInput(),
if (_showEmoji)
  SizedBox(
    height: mq.height * .35,
    child: EmojiPicker(
      textEditingController: _textController,
```

```
                config: Config(
                  bgColor: const Color.fromARGB(255, 234, 248, 255),
                  columns: 8,
                  emojiSizeMax: 32 * (Platform.isIOS ? 1.30 : 1.0),
                ),
              ),
          )
          ],
        ),
      ),
    ),
  );
}
Widget _appBar() {
  return InkWell(
    onTap: () {
      Navigator.push(
          context,
          MaterialPageRoute(
              builder: (_) => ViewProfileScreen(user: widget.user)));
    },
    child: StreamBuilder(
        stream: APIs.getUserInfo(widget.user),
        builder: (context, snapshot) {
          final data = snapshot.data?.docs;
          final list =
              data?.map((e) => ChatUser.fromJson(e.data())).toList() ?? [];

          return Row(
            children: [
              //back button
              IconButton(
                  onPressed: () => Navigator.pop(context),
```

```
    icon:
        const Icon(Icons.arrow_back, color: Colors.black54)),

//user profile picture
ClipRRect(
  borderRadius: BorderRadius.circular(mq.height * .03),
  child: CachedNetworkImage(
    width: mq.height * .05,
    height: mq.height * .05,
    imageUrl:
        list.isNotEmpty ? list[0].image : widget.user.image,
    errorWidget: (context, url, error) => const CircleAvatar(
        child: Icon(CupertinoIcons.person)),
  ),
),

//for adding some space
const SizedBox(width: 10),

//user name & last seen time
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    //user name
    Text(list.isNotEmpty ? list[0].name : widget.user.name,
        style: const TextStyle(
            fontSize: 16,
            color: Colors.black87,
            fontWeight: FontWeight.w500)),

    const SizedBox(height: 2),

    Text(
```

```
                    list.isNotEmpty
                        ? list[0].isOnline
                            ? 'Online'
                            : MyDateUtil.getLastActiveTime(
                                context: context,
                                lastActive: list[0].lastActive)
                        : MyDateUtil.getLastActiveTime(
                            context: context,
                            lastActive: widget.user.lastActive),
                    style: const TextStyle(
                        fontSize: 13, color: Colors.black54)),
                ],
              )
            ],
          );
        }));
}
Widget _chatInput() {
  return Padding(
    padding: EdgeInsets.symmetric(
        vertical: mq.height * .01, horizontal: mq.width * .025),
    child: Row(
      children: [
        //input field & buttons
        Expanded(
          child: Card(
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(15)),
            child: Row(
              children: [
                IconButton(
                    onPressed: () {
                      FocusScope.of(context).unfocus();
                      setState(() => _showEmoji = !_showEmoji);
```

```
    },
    icon: const Icon(Icons.emoji_emotions,
        color: Colors.blueAccent, size: 25)),

Expanded(
    child: TextField(
  controller: _textController,
  keyboardType: TextInputType.multiline,
  maxLines: null,
  onTap: () {
    if (_showEmoji) setState(() => _showEmoji = !_showEmoji);
  },
  decoration: const InputDecoration(
      hintText: 'Type Something...',
      hintStyle: TextStyle(color: Colors.blueAccent),
      border: InputBorder.none),
)),
IconButton(
    onPressed: () async {
      final ImagePicker picker = ImagePicker();
      final List<XFile> images =
          await picker.pickMultiImage(imageQuality: 70);
      for (var i in images) {
        log('Image Path: ${i.path}');
        setState(() => _isUploading = true);
        await APIs.sendChatImage(widget.user, File(i.path));
        setState(() => _isUploading = false);
      }
    },
    icon: const Icon(Icons.image,
        color: Colors.blueAccent, size: 26)),
IconButton(
    onPressed: () async {
      final ImagePicker picker = ImagePicker();
```

```
              final XFile? image = await picker.pickImage(
                  source: ImageSource.camera, imageQuality: 70);
              if (image != null) {
                log('Image Path: ${image.path}');
                setState(() => _isUploading = true);

                await APIs.sendChatImage(
                    widget.user, File(image.path));
                setState(() => _isUploading = false);
              }
            },
            icon: const Icon(Icons.camera_alt_rounded,
                color: Colors.blueAccent, size: 26)),
        SizedBox(width: mq.width * .02),
      ],
    ),
  ),
),
MaterialButton(
  onPressed: () {
    if (_textController.text.isNotEmpty) {
      if (_list.isEmpty) {
        //on first message (add user to my_user collection of chat user)
        APIs.sendFirstMessage(
            widget.user, _textController.text, Type.text);
      } else {
        APIs.sendMessage(
            widget.user, _textController.text, Type.text);
      }
      _textController.text = '';
    }
  },
  minWidth: 0,
  padding:
```

```
            const EdgeInsets.only(top: 10, bottom: 10, right: 5, left: 10),
          shape: const CircleBorder(),
          color: Colors.green,
          child: const Icon(Icons.send, color: Colors.white, size: 28),
        )
      ],
    ),
  );
 }
}
```

Output: