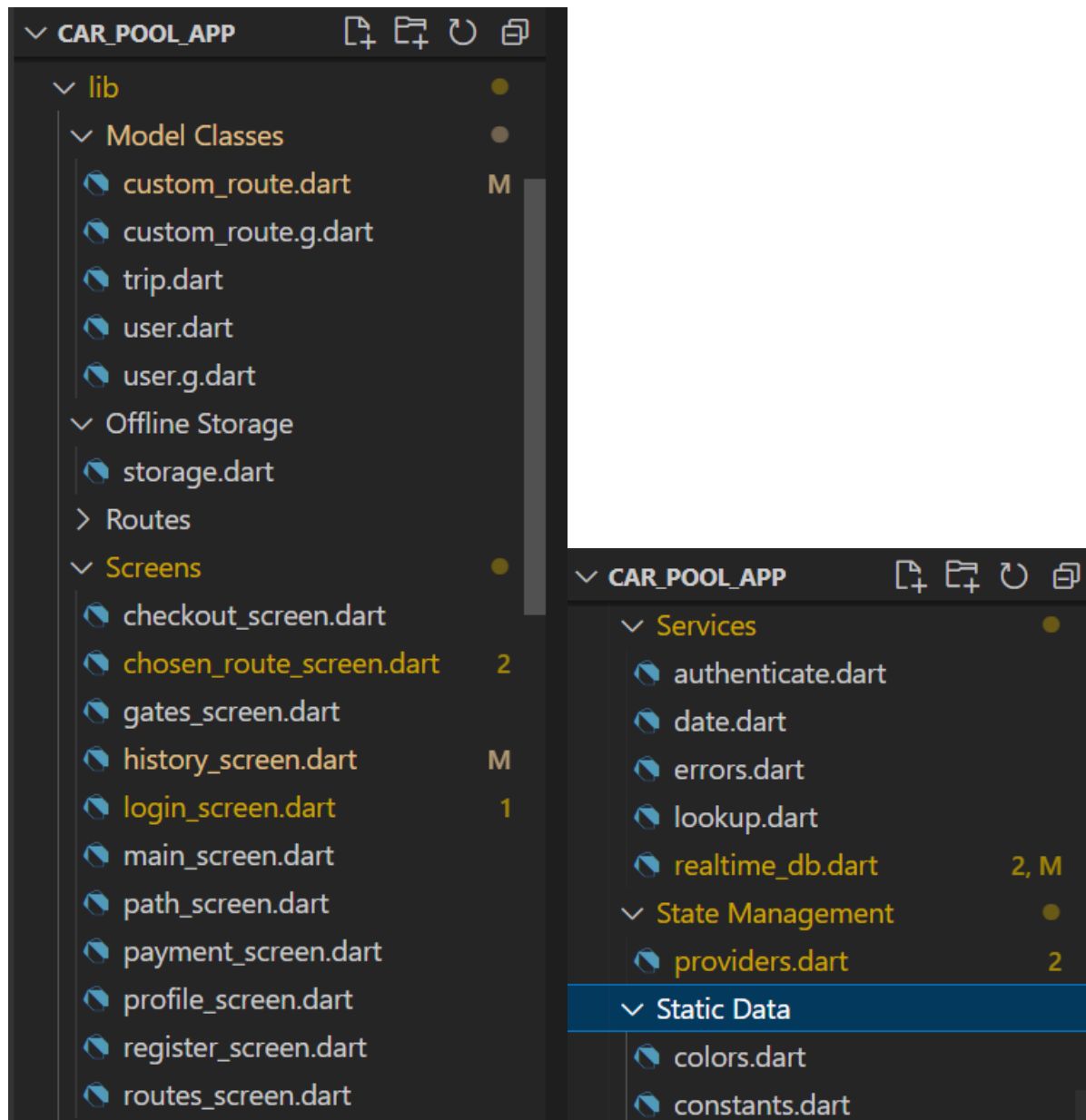


## CSE 431 – Milestone #2

Name: Ahmed Sameh Mohamed Mourad

ID: 19P5861

### Folder Structure:



✓ Widgets

- custom\_button.dart
- custom\_container.dart
- custom\_text.dart
- email\_field.dart
- gate\_widget.dart
- name\_field.dart
- password\_field.dart
- profile\_column.dart
- promo\_code\_field.dart
- routes\_list\_view.dart
- search.dart
- shimmer\_template.dart
- sized\_box.dart
- wrapper.dart
- firebase\_options.dart
- interface.dart
- main.dart

---

“authenticate.dart” Code:

```
import 'package:flutter/material.dart';

import 'package:car_pool_app/Services/lookup.dart';
import 'package:car_pool_app/Services/errors.dart';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:dartz/dartz.dart';

final auth = FirebaseAuth.instance;

Stream<User?> get userStream
{
  debugPrint('User Stream Triggered !!!!!');
  return auth.userChanges();
}
```

```

}

Future<void> signout() async
{
  try{
    return await auth.signOut();
  }
  catch(e){
    debugPrint(e.toString());
    debugPrint('error signing out!');
    return;
  }
}

Future< Either<ErrorTypes, User?> > registerWithEmail({required String email,
required String password}) async
{
  final connection = await LookUp.checkInternetConnection();

  return connection.fold(
    (error) {
      return Left(error);
    },
    (success) async {
      try{
        final userCredential = await auth.createUserWithEmailAndPassword(
          email: email,
          password: password,
        );
        return Right(userCredential.user);
      }
      on FirebaseAuthException catch(e) {

        if(e.code == 'email-already-in-use'){
          debugPrint("email already in use!!");
        }
        else if(e.code == 'weak-password'){
          debugPrint("weak password....");
        }
        else if(e.code == "invalid-email"){
          debugPrint("invalid- Email... pls enter a valid one");
        }
        else if(e.code == 'operation-not-allowed'){
          debugPrint("operation not allowed, contact support !");
        }
      }
    }
  );
}

```

```

        return Left(
            UnexpectedError(
                customError: e.code,
                errorId: 100,
            ),
        );
    }
    catch(e) {
        debugPrint("creating a new email failed!!");
        return Left(
            UnexpectedError(
                customError: e.toString(),
                errorId: 101,
            ),
        );
    }
},
);
}

Future< Either<ErrorTypes, bool> > loginWithEmail({required String email,
required String password}) async
{

    final connection = await LookUp.checkInternetConnection();

    return connection.fold(
        (error) {
            return Left(error);
        },
        (right) async {
            try{
                /*UserCredential user = */await auth.signInWithEmailAndPassword(
                    email: email,
                    password: password,
                );
                //return user.user;
                return const Right(true);
            }
            on FirebaseAuthException catch (e) {
                if(e.code == 'user-not-found')
                {
                    debugPrint('No user found for that email.');
```

```

        customError: e.code,
        errorId: 400,
    ),
);
}
else if(e.code == 'wrong-password')
{
    return const Left(
        WrongPasswordError(),
    );
}
else if(e.code == 'invalid-email'){
    debugPrint('invalid email');
    return Left(
        UnexpectedError(
            customError: e.code,
            errorId: 401,
        ),
    );
}
else if(e.code == 'user-disabled'){
    return const Left(
        UserDisabledError(),
    );
}
return Left(
    UnexpectedError(
        customError: 'Unexpected FirebaseAuthException',
        errorId: 402,
    ),
);
}
catch (e) {
    return Left(
        FirebaseError(
            errorMessage: 'Server Error: $e',
            errorId: 301,
        ),
    );
}
},
);
}

```

---

### “realtime db.dart” Code:

```
import 'dart:io';
import 'package:flutter/material.dart'; // used for debugPrint()

import 'package:car_pool_app/Services/lookup.dart';
import 'package:car_pool_app/Services/errors.dart';
import 'package:car_pool_app/Model%20Classes/custom_route.dart';
import 'package:car_pool_app/Model%20Classes/user.dart';

import 'package:dartz/dartz.dart';
import 'package:firebase_database/firebase_database.dart';

final routesReference = FirebaseDatabase.instance.ref("routes");

class Realtime {

  final String uid;

  Realtime({
    required this.uid,
  });

  final usersReference = FirebaseDatabase.instance.ref().child("/users");

  // UserData Methods

  Future< Either<ErrorTypes, bool> > addUserData(User user) async {
    final connection = await LookUp.checkInternetConnection();
    return connection.fold(
      (error) {
        return Left(error);
      },
      (right) async {
        try {
          await usersReference.child(uid).set(user.toJson());
          return const Right(true);
        }
        catch(e) {
          return Left(
            FirebaseError(
              errorMessage: 'Server Error: $e',
              errorId: 103,
```

```

        ),
    );
}
},
);
}

// CustomRoute Methods

Future< Either<ErrorTypes, bool> > addRoutes(List<CustomRoute> routes) async {
    final connection = await LookUp.checkInternetConnection();
    return connection.fold(
        (error) {
            return Left(error);
        },
        (right) async {
            try {
                for(int i = 0; i < routes.length; i++) {
                    await routesReference.child('$i').set(routes[i].toJson());
                }
                return const Right(true);
            }
            catch(e) {
                return Left(
                    FirebaseError(
                        errorMessage: 'Server Error: $e',
                        errorId: 103,
                    ),
                );
            }
        },
    );
}

Future< Either<ErrorTypes, List<CustomRoute>> > getRoutes() async {
    final connection = await LookUp.checkInternetConnection();
    return connection.fold(
        (error) {
            return Left(error);
        },
        (right) async {
            try {
                List<CustomRoute> routes = [];

                final response = await routesReference.once().then((event) {

```

```

        final List jsonList = event.snapshot.value as List;

        routes = List<CustomRoute>.from(jsonList.map((route) =>
CustomRoute.fromJson(route.cast<String, dynamic>())));
    });

    return Right(routes);
}
catch(e) {
    return Left(
        FirebaseError(
            errorMessage: 'Server Error: $e',
            errorId: 103,
        ),
    );
}
},
);
}
}

```

---

### “register screen.dart” Code:

```

import 'package:car_pool_app/Offline%20Storage/storage.dart';
import 'package:flutter/material.dart';

import 'package:car_pool_app/Widgets/email_field.dart';
import 'package:car_pool_app/Widgets/password_field.dart';
import 'package:car_pool_app/Widgets/sized_box.dart';
import 'package:car_pool_app/Widgets/custom_button.dart';
import 'package:car_pool_app/Widgets/name_field.dart';
import 'package:car_pool_app/Services/authenticate.dart';
import 'package:car_pool_app/Screens/login_screen.dart';
import 'package:car_pool_app/Services/realtime_db.dart';
import 'package:car_pool_app/Model%20Classes/user.dart';

import 'package:provider/provider.dart';
import 'package:firebase_auth/firebase_auth.dart' as auth;

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

```



```

static const routeName = '/register';

@override
State<RegisterScreen> createState() => _RegisterScreenState();
}

class _RegisterScreenState extends State<RegisterScreen> {

  final _registerFormKey = GlobalKey<FormState>();

  final _nameController = TextEditingController();

  final _emailController = TextEditingController();

  final _passController = TextEditingController();

  late FocusNode _nameNode;

  late FocusNode _emailNode;

  late FocusNode _passNode;

  @override
  void initState() {

    _nameNode = FocusNode();
    _emailNode = FocusNode();
    _passNode = FocusNode();

    super.initState();
  }

  @override
  void dispose() {
    _nameNode.dispose();
    _emailNode.dispose();
    _passNode.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(

```

```

        title: const Text('Register Screen'),
        centerTitle: true,
    ),

    body: Form(
      key: _registerFormKey,
      child: Padding(
        padding: const EdgeInsets.all(10.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            NameField(
              autoFocus: true,
              controller: _nameController,
              focusNode: _nameNode,
            ),

            const SizedBox(
              height: 20,
            ),

            EmailField(
              controller: _emailController,
              focusNode: _emailNode,
            ),

            const SizedBox(
              height: 20,
            ),

            PasswordField(
              controller: _passController,
              focusNode: _passNode,
            ),

            const SizedBox(
              height: 20,
            ),

            CustomButton(
              onTap: () async {
                if(_registerFormKey.currentState!.validate()) {
                  final registrationResult = await registerWithEmail(
                    email: _emailController.text,
                    password: _passController.text,

```

```

        );

        registrationResult.fold(
            (error) {},
            (success) async {
                final uid = context.read<auth.User>().uid;

                final user = User(
                    uid: uid,
                    email: _emailController.text,
                    name: _nameController.text,
                    points: 0,
                    tripsCount: 0,
                );

                final addResult = await Realtime(uid:
user.uid).addUserData(user);

                addResult.fold(
                    (error) {},
                    (success) async {
                        await UserStorage.addUser(user);
                    },
                );

                if(context.mounted) {
                    Navigator.pop(context);
                }
            },
        );
    }

    width: 100,
    height: 50,
    child: const Text('Register'),
),

const HSizeBox(
    height: 20,
),

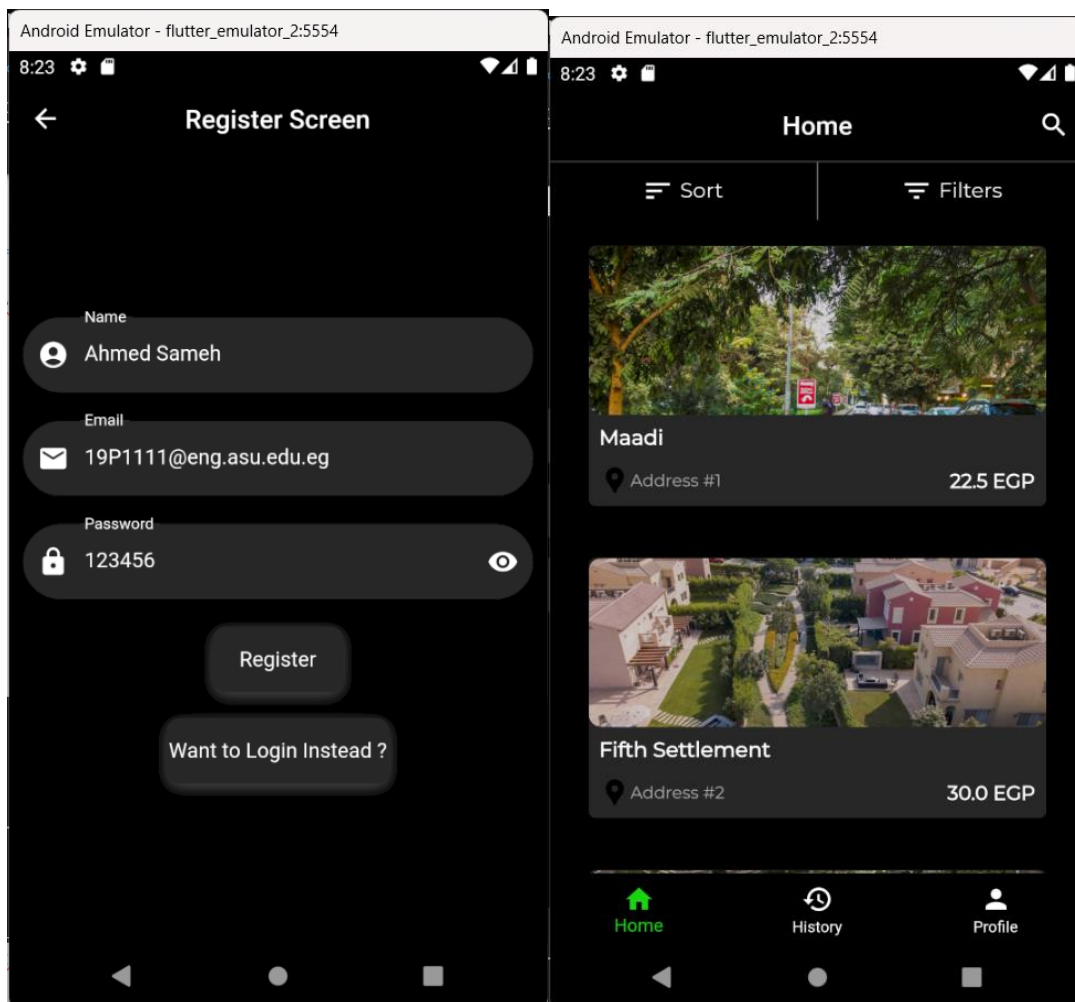
CustomButton(
    onTap: () {
        Navigator.pushReplacementNamed(context, LoginScreen.routeName);
    },

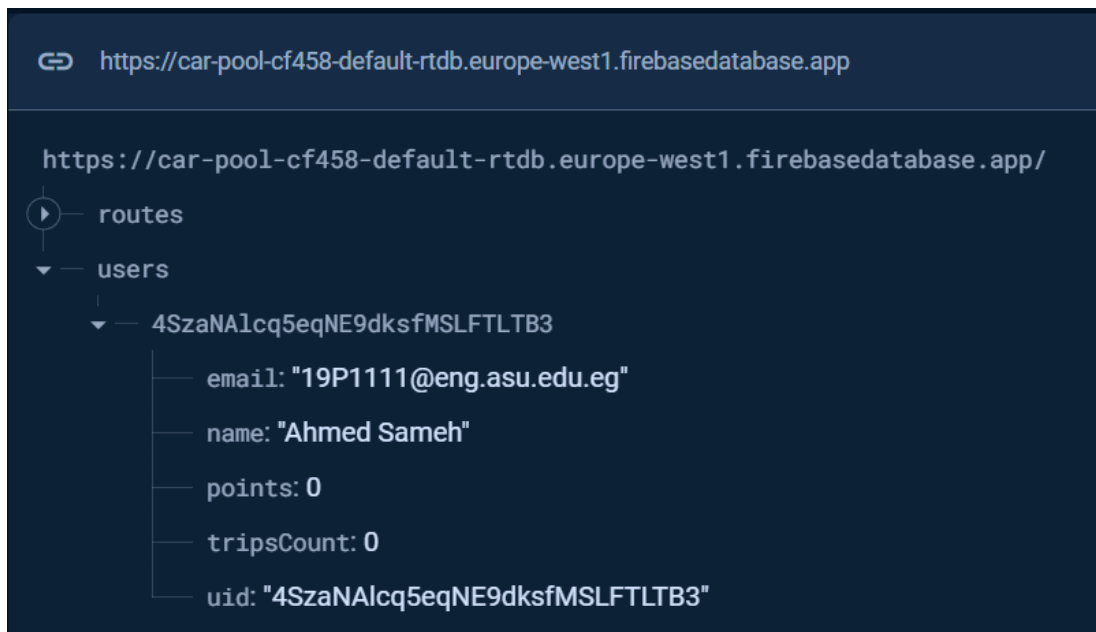
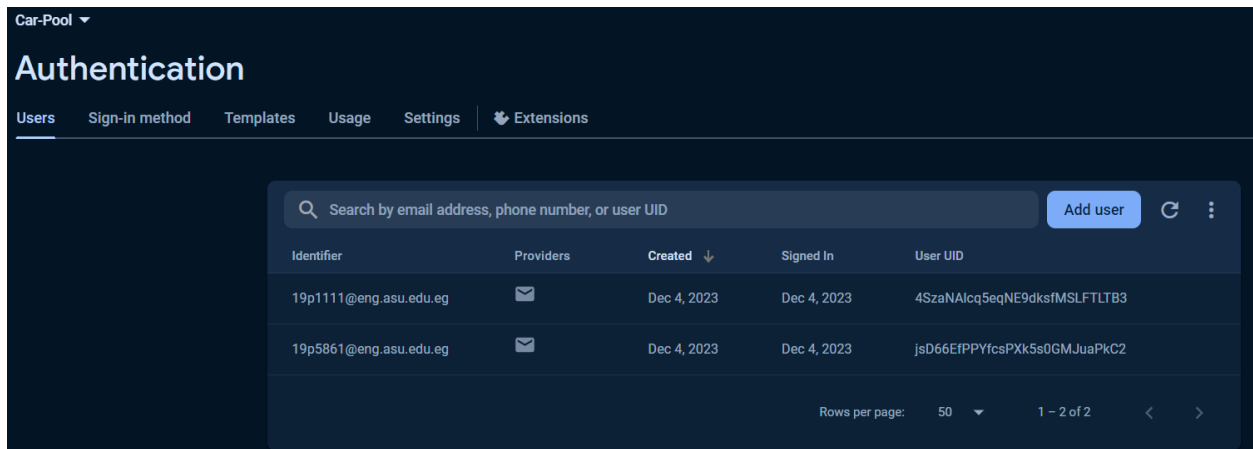
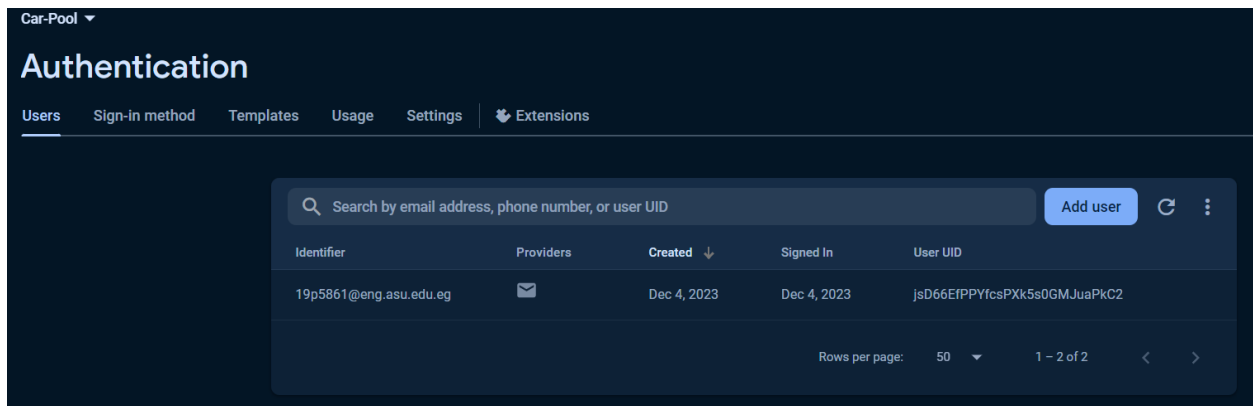
```

```

        width: 170,
        height: 50,
        child: const Text('Want to Login Instead ?'),
      ),
    ],
  ),
),
);
}
}

```





"login\_screen.dart" Code:

```
import 'package:flutter/material.dart';

import 'package:car_pool_app/Widgets/email_field.dart';
import 'package:car_pool_app/Widgets/password_field.dart';
import 'package:car_pool_app/Widgets/sized_box.dart';
import 'package:car_pool_app/Widgets/custom_button.dart';
import 'package:car_pool_app/Screens/path_screen.dart';

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

  static const routeName = '/login';

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {

  final _loginFormKey = GlobalKey<FormState>();

  final _emailController = TextEditingController();

  final _passController = TextEditingController();

  late FocusNode _emailNode;

  late FocusNode _passNode;

  @override
  void initState() {

    _emailNode = FocusNode();
    _passNode = FocusNode();

    _emailNode.requestFocus();

    super.initState();
  }

  @override
  void dispose() {
    _emailNode.dispose();
    _passNode.dispose();
  }
}
```

```

    super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Login Screen'),
      centerTitle: true,
    ),

    body: Form(
      key: _loginFormKey,
      child: Padding(
        padding: const EdgeInsets.all(10.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            EmailField(
              controller: _emailController,
              focusNode: _emailNode,
            ),

            const SizedBox(
              height: 20,
            ),

            PasswordField(
              controller: _passController,
              focusNode: _passNode,
            ),

            const SizedBox(
              height: 20,
            ),

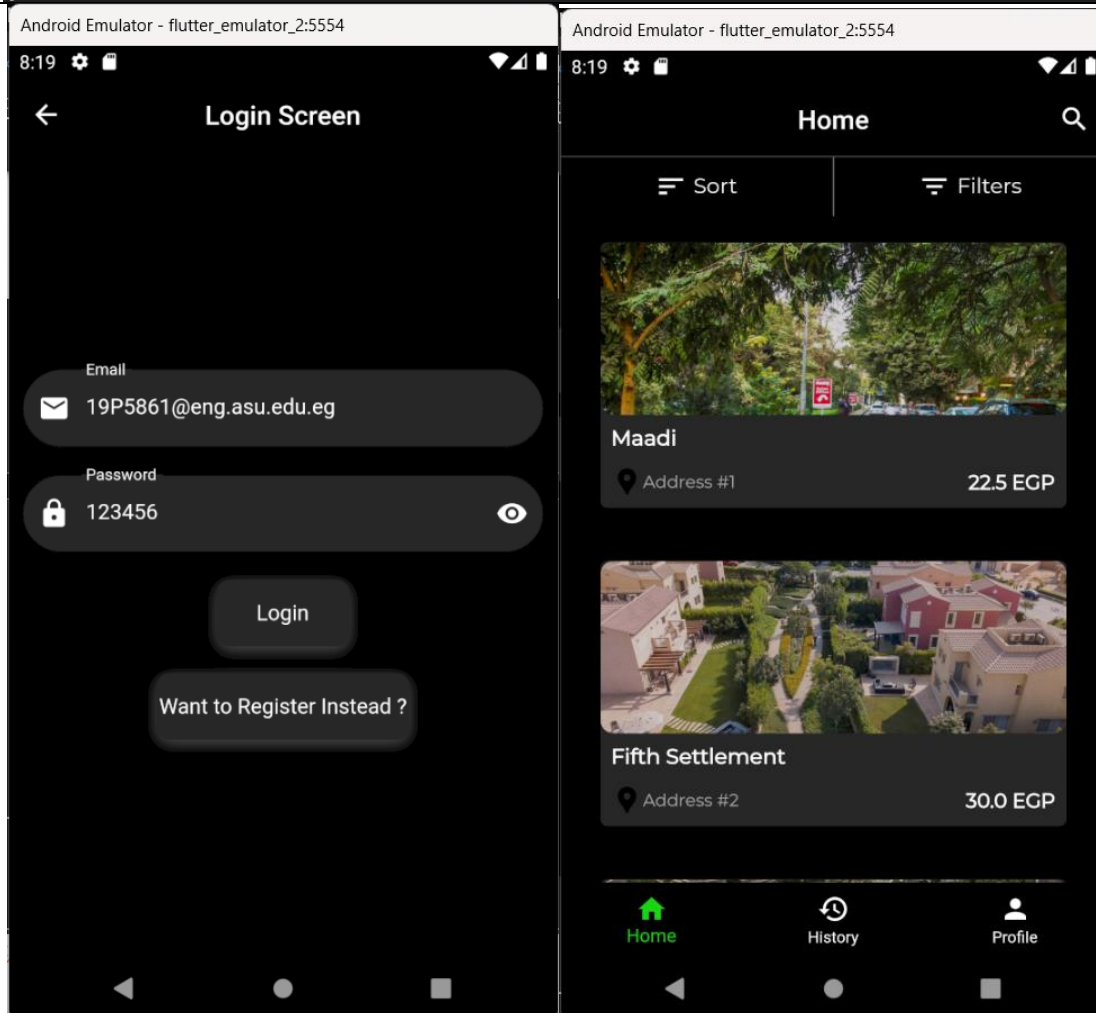
            CustomButton(
              onTap: () {
                // if(_loginFormKey.currentState!.validate()) {}
                Navigator.pushNamed(context, PathScreen.routeName);
              },
              width: 100,
              height: 50,
              child: const Text('Login'),
            ),
          ],
        ),
      ),
    ),
  );
}

```

```

    ],
  ),
),
),
);
}
}

```



“routes\_list\_view.dart” Code:

```

import 'package:flutter/material.dart';

import 'package:car_pool_app/Screens/chosen_route_screen.dart';
import 'package:car_pool_app/Widgets/custom_button.dart';
import 'package:car_pool_app/Widgets/custom_text.dart';
import 'package:car_pool_app/Widgets/sized_box.dart';

```



```

import 'package:car_pool_app/Model%20Classes/custom_route.dart';

class RoutesListView extends StatelessWidget {
  const RoutesListView({
    super.key,
    required this.routes,
  });

  final List<CustomRoute> routes;

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: routes.length,
      itemBuilder: (context, index) {
        return Padding(
          padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 20,),
          child: CustomButton(
            shadow: false,
            borderRadius: 5,
            onTap: () {
              final locationData = routes[index];
              Navigator.pushNamed(context, ChosenRouteScreen.routeName, arguments:
locationData);
            },
            child: Column(
              children: [
                Container(
                  width: MediaQuery.of(context).size.width,
                  height: 130,
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(10),
                    image: DecorationImage(
                      image: AssetImage(
                        'assets/images/${routes[index].name}.jpg'
                      ),
                      opacity: .8,
                      fit: BoxFit.fitWidth,
                    ),
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,

```

```

        children: [
          CustomText(
            text: routes[index].name,
            size: 16,
            fontWeight: FontWeight.bold,
          ),
          const HSizeBox(
            height: 10,
          ),

          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Row(
                children: [
                  const Icon(Icons.location_pin),
                  CustomText(
                    text: routes[index].address,
                    size: 13,
                    textColor: Colors.white60,
                  ),
                ],
              ),
              CustomText(
                text: '${routes[index].price} EGP',
                fontWeight: FontWeight.bold,
              ),
            ],
          ),
        ],
      ),
    ],
  ),
),
);
},
);
}
}
}

```

“routes screen.dart” Code:

```
import 'package:flutter/material.dart';
```

```

import 'package:car_pool_app/Widgets/custom_text.dart';
import 'package:car_pool_app/Widgets/sized_box.dart';
import 'package:car_pool_app/Model%20Classes/custom_route.dart';
import 'package:car_pool_app/Services/realtime_db.dart';
import 'package:car_pool_app/Widgets/routes_list_view.dart';
import 'package:car_pool_app/Widgets/search.dart';
import 'package:car_pool_app/Screens/chosen_route_screen.dart';

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

  static const routeName = '/routes';

  @override
  State<RoutesScreen> createState() => _RoutesScreenState();
}

class _RoutesScreenState extends State<RoutesScreen> {

  Future<List>? schoolsFuture;

  final List<String> sort = ['Alphabetically [A-Z]', 'Alphabetically [Z-A]',
'Lowest to Highest Price', 'Highest to Lowest Price'];

  late String currentSort;

  late Future< List<CustomRoute> > routesFuture;

  late List<CustomRoute> routes;

  void showSortList() {
    showModalBottomSheet(
      context: context,
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.vertical(
          top: Radius.circular(20),
        ),
      ),
      builder: (context) {
        return StatefulBuilder(
          builder: (context, StateSetter setSheetState) {
            return SizedBox(
              height: 230,
              child: ListView.builder(

```

```

        itemCount: sort.length,
        itemBuilder: (context, index) {
            return ListTile(
                title: Text(sort[index]),
                onTap: () {
                    setSheetState(() {
                        setState(() {
                            currentSort = sort[index];
                        });
                    });
                },
                leading: Radio(
                    value: sort[index],
                    groupValue: currentSort,
                    onChanged: (value) {
                        setSheetState(() {
                            setState(() {
                                currentSort = value!;
                            });
                        });
                    },
                ),
            );
        },
    );
},
);
},
),
);
},
);
},
);
},
);
}

void sorting() {
    if(currentSort == 'Lowest to Highest Price') {
        routes.sort(
            (CustomRoute l1, CustomRoute l2) {
                return l1.price.compareTo(l2.price);
            }
        );
    }

    else if(currentSort == 'Highest to Lowest Price') {
        routes.sort(
            (CustomRoute l1, CustomRoute l2) {
                return l2.price.compareTo(l1.price);
            }
        );
    }
}

```

```

    }
  );
}

else if(currentSort == 'Alphabetically [A-Z]') {
  routes.sort(
    (CustomRoute l1, CustomRoute l2) {
      return l1.name.toLowerCase().compareTo(l2.name.toLowerCase());
    }
  );
}

else if(currentSort == 'Alphabetically [Z-A]') {
  routes.sort(
    (CustomRoute l1, CustomRoute l2) {
      return l2.name.toLowerCase().compareTo(l1.name.toLowerCase());
    }
  );
}
}

Future< List<CustomRoute> > initRoutes() async {
  final temp = await Realtime(uid: 'uid').getRoutes();

  return temp.fold(
    (error) {
      return Future< List<CustomRoute> >.error(error);
    },
    (success) {
      return Future< List<CustomRoute> >.value(success);
    },
  );
}

@override
void initState() {
  currentSort = sort[0];

  routesFuture = initRoutes();

  super.initState();
}

@override
Widget build(BuildContext context) {

```

```

return Scaffold(
  appBar: AppBar(
    title: const Text('Home'),
    centerTitle: true,
    actions: [
      IconButton(
        icon: const Icon(Icons.search),
        onPressed: () async {

          final searchResult = await showSearch(
            context: context,
            delegate: Search(routes: routes,),
          );

          if(searchResult != null && context.mounted) {
            Navigator.pushNamed(context, ChosenRouteScreen.routeName,
arguments: searchResult);
          }
        },
      ),
    ],
  bottom: PreferredSize(
    preferredSize: const Size.fromHeight(45),
    child: Row(
      children: [
        Container(
          height: 45,
          width: MediaQuery.of(context).size.width / 2,
          decoration: const BoxDecoration(
            border: Border(
              top: BorderSide(
                width: 0.5,
                color: Color(0xFF707070),
              ),
              right: BorderSide(
                width: 0.5,
                color: Color(0xFF707070),
              ),
            ),
          ),
        ),
        InkWell(
          onTap: showSortList,
          child: const Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [

```

```

        Icon(
          Icons.sort,
          color: Colors.white,
        ),
        WSizeBox(
          width: 5,
        ),
        CustomText(
          text: 'Sort',
          size: 16,
        ),
      ],
    ),
  ),
),

Container(
  height: 45,
  width: MediaQuery.of(context).size.width / 2,
  decoration: const BoxDecoration(
    border: Border(
      top: BorderSide(
        width: 0.5,
        color: Color(0xFF707070),
      ),
      left: BorderSide(
        width: 0.5,
        color: Color(0xFF707070),
      ),
    ),
  ),
  child: const Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Icon(
        Icons.filter_list,
        color: Colors.white,
      ),
      WSizeBox(
        width: 5,
      ),
      CustomText(
        text: 'Filters',
        size: 16,
      ),
    ],
  ),
),

```

```

        ],
      ),
    ),
  ],
),
),
),
body: Center(
  child: FutureBuilder(
    future: routesFuture,
    builder: (context, snapshot) {
      switch(snapshot.connectionState) {
        case ConnectionState.none:
          return const CustomText(
            text: 'None',
            size: 30,
          );

        case ConnectionState.waiting:
        case ConnectionState.active:
          return const CircularProgressIndicator(
            strokeWidth: 2,
          );

        case ConnectionState.done:
          if(snapshot.hasError) {
            return CustomText(
              text: '${snapshot.error}',
              size: 30,
            );
          }

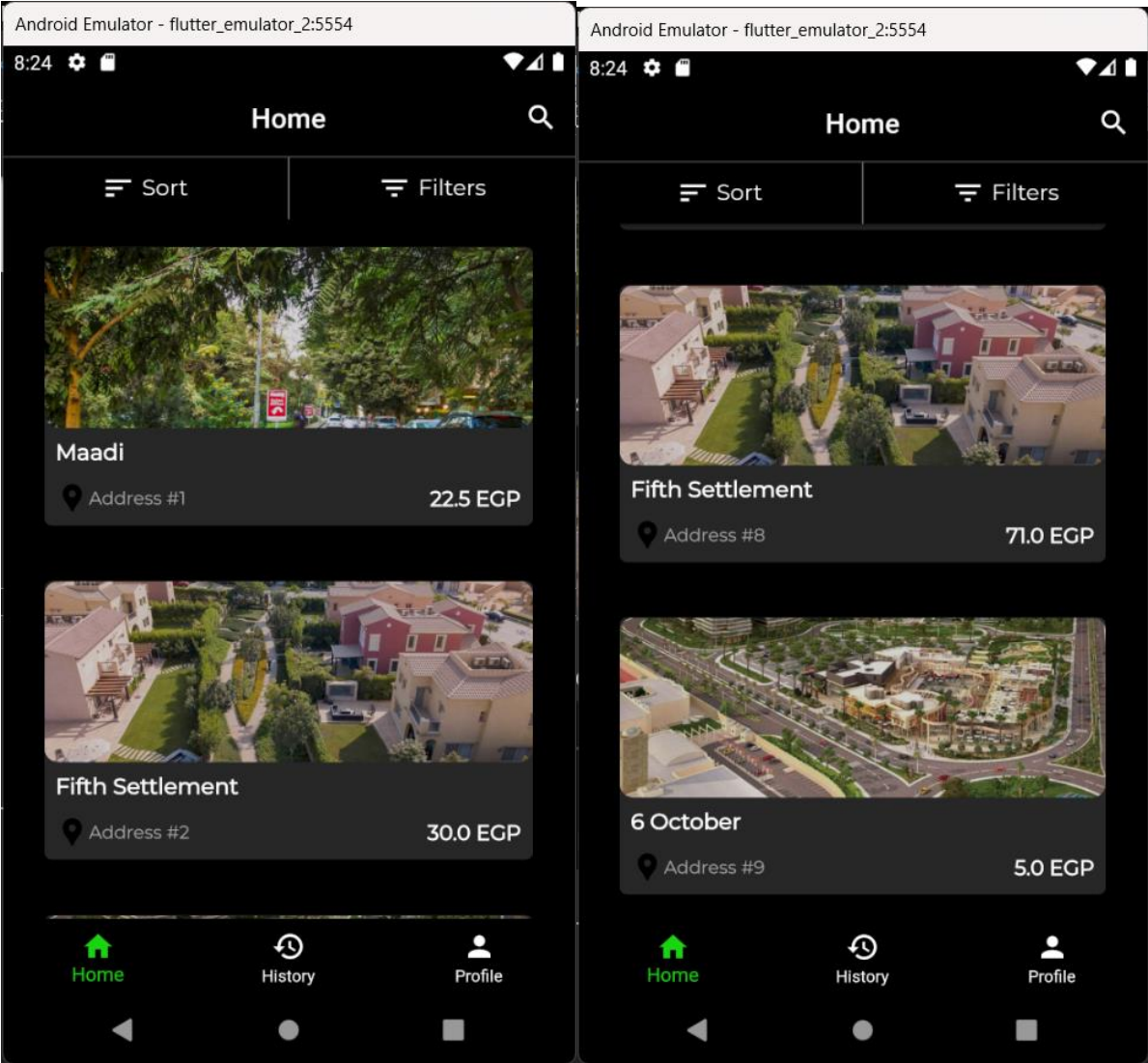
          else if(snapshot.hasData) {
            routes = snapshot.data as List<CustomRoute>;
            return RoutesListView(
              routes: routes,
            );
          }

          return const SizedBox.shrink();
        }
      },
    ),
  ),
),

```



```
}  
}  
);
```




Car-Pool ▾

# Realtime Database

Data Rules Backups Usage Extensions

<https://car-pool-cf458-default-rtdb.europe-west1.firebaseio.com>

<https://car-pool-cf458-default-rtdb.europe-west1.firebaseio.com/>


- routes
  - 0 + 
    - address: "Address #1"
    - location: ""
    - name: "Maadi"
    - price: 22.5
  - 1
    - address: "Address #2"
    - location: ""
    - name: "Fifth Settlement"
    - price: 30

Car-Pool ▾

# Realtime Database

Data Rules Backups Usage Extensions

<https://car-pool-cf458-default-rtdb.europe-west1.firebaseio.com>

- 6
- 7
  - address: "Address #8"
  - location: ""
  - name: "Fifth Settlement"
  - price: 71
- 8 + 
  - address: "Address #9"
  - location: ""
  - name: "6 October"
  - price: 5