

Object Oriented Programming

Project Lab Report



FALL 2024

Submitted by: **Maryam Fareed**

Registration No: **23PWCSE2228**

Section: **A**

Submitted by: **Younna Saifullah**

Registration No: **23PWCSE2246**

Section: **A**

Submitted To:

Mam Sumayyea Salahuddin

Department Of Computer Systems Engineering
University of Engineering And Technology, Peshawar

Rental Car App in Flutter using OOPS

Explanation of code:

Step by step:

On boarding page:

```
1  import 'package:flutter/material.dart';
2  import 'package:myproject/presentation/pages/login_page.dart';
3
4  class OnboardingPage extends StatelessWidget {
5    const OnboardingPage({super.key});
6
7    @override
8    Widget build(BuildContext context) {
9      return Scaffold(
10        backgroundColor: const Color(value: 0xff2C2B34),
11        body: Column(
12          children: <Widget>[]
13        ),
14      );
15    }
16
17    Expanded(
18      flex: 2,
19      child: Container(
20        width: double.infinity,
21        inline-size: double.infinity, ⇌ width: double.infinity,
22        clipBehavior: Clip.antiAlias,
23        decoration: const BoxDecoration(
24          image: DecorationImage(
25            image: AssetImage(assetName: 'assets/whitecar2.jpg'),
26            fit: BoxFit.cover,
27          ), // DecorationImage
28        ), // BoxDecoration
29      ), // Container
30    ), // Expanded
31
32    Expanded(
33      child: Padding(
34        padding: const EdgeInsets.symmetric(horizontal: 24.0), // ✓ Added padding
35        child: Column(
36          mainAxisAlignment: MainAxisAlignment.center,
37          crossAxisAlignment: CrossAxisAlignment.start, // ✓ Keeps text aligned
38          children: <Widget>[
39            const Text(
```

```

34     data: 'Premium Cars\nEnjoy the luxury',
35     style: TextStyle{
36       color: Colors.white,
37       fontSize: 32,
38       fontWeight: FontWeight.bold,
39     }, // TextStyle
40   ), // Text
41   const SizedBox(height: 10), // Added spacing block-size: 10 ⇔ height: 10
42   const Text(
43     data: 'Premium and Prestige car daily rental\nExperiencing the thrill at a lower price',
44     style: TextStyle(color: Colors.grey, fontSize: 16),
45   ), // Text
46   const SizedBox(height: 30), // Increased spacing block-size: 30 ⇔ height: 30
47
48   /// Responsive button with double.infinity
49   SizedBox(
50     width: double.infinity, // inline-size: double.infinity, ⇔ width: double.infinity,
51     height: 54, // block-size: 54, ⇔ height: 54,
52     child: ElevatedButton(
53       onPressed: () {
54         Navigator.of(context).pushAndRemoveUntil<dynamic>(
55           MaterialPageRoute<dynamic>(builder: (BuildContext context) => login()),
56           predicate: (Route<dynamic> route) => false,
57         );
58       },
59       style: ElevatedButton.styleFrom(
60         foregroundColor: Colors.black,
61         backgroundColor: Colors.white,
62       ),
63       child: const Text(

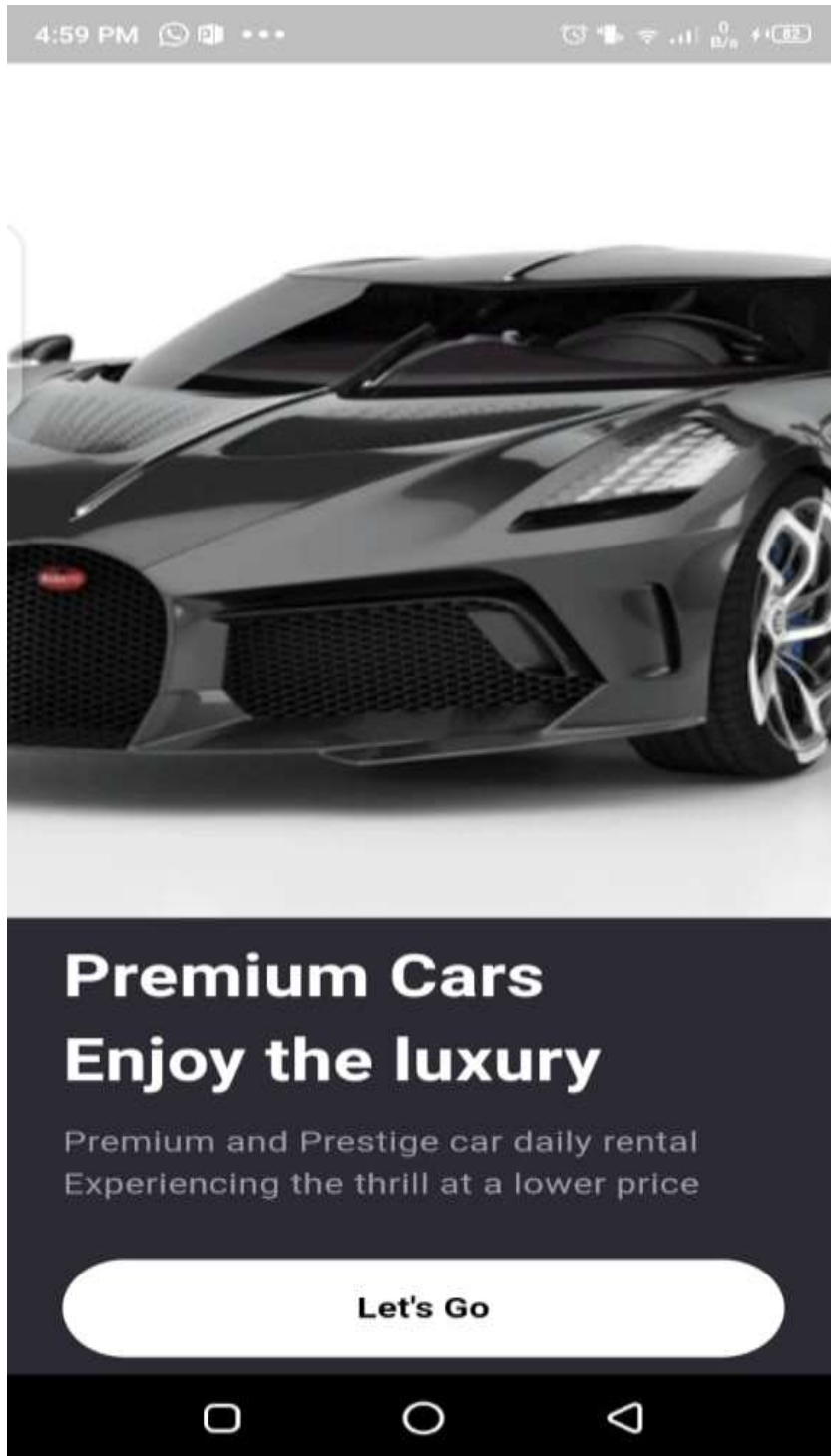
```

```

63       child: const Text(
64         data: "Let's Go",
65         style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
66       ), // Text
67     ), // ElevatedButton
68   ), // SizedBox
69 ],
70 ), // Column
71 ), // Padding
72 ), // Expanded
73 ],
74 ), // Column
75 ); // Scaffold
76 }
77 }

```

Output:



Explanation of code:

Importing Required Packages

- The flutter/material.dart package is imported to use Flutter UI components.

- The `login_page.dart` file is imported to enable navigation to the login screen.

Defining the OnboardingPage Class

- OnboardingPage is a **stateless widget**, meaning it does not maintain any state.
- The constructor includes `{super.key}` to improve widget rebuilding efficiency.

Building the UI with Scaffold

- The Scaffold widget provides the basic structure of the screen.
- The background color is set to a dark shade using a hexadecimal color code.

Using a Column for Layout

- The Column widget is used to arrange UI elements vertically.
- Two Expanded widgets divide the screen into two sections: one for the image and one for text and a button.

Displaying the Image

- An Expanded widget with `flex: 2` ensures the image takes two-thirds of the screen height.
- The image is wrapped inside a Container and displayed using `AssetImage()`.
- `BoxFit.cover` ensures the image covers the entire container.

Adding Text Content

- The second Expanded widget holds the text and button.
- Padding is applied to maintain spacing from screen edges.
- The **title text** is styled with bold white color and a font size of 32.
- A **subtitle** is added in grey color with a smaller font size for additional details.

Adding a Button for Navigation

- A `SizedBox` is used to make the button **full width** and **54 pixels high**.
- An `ElevatedButton` is created with a white background and black text.
- The `onPressed` function uses `Navigator.of(context).pushAndRemoveUntil()` to navigate to the Login page, removing the onboarding screen from memory.

Key OOP Concepts Used

- **Encapsulation:** The OnboardingPage class encapsulates all UI elements inside a single widget.
- **Abstraction:** Implementation details are hidden within widgets like Container, Text, and ElevatedButton.
- **Reusability:** Using a StatelessWidget makes this page easily reusable.
- **Navigation:** Uses the Navigator class to handle screen transitions efficiently.

Conclusion

- This onboarding screen provides a visually appealing introduction to the app.
- It effectively combines **text, images, and navigation** to create a smooth user experience.

Sign-up Page:

```
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:myproject/firebase_auth_services.dart';
import 'package:myproject/presentation/pages/login_page.dart';

class SignupPage extends StatelessWidget {
  SignupPage({super.key});

  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      resizeToAvoidBottomInset: true,
      bottomNavigationBar: _signin(context),
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0,
        toolbarHeight: 50, // block-size: 50, ⇒ Height: 50, 🍌
      ), // AppBar
      body: SafeArea(
        child: SingleChildScrollView(
          padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 16),
          child: Column(
            children: [
              Center(
                child: Text(
                  'Register Account',
                  style: GoogleFonts.raleway(
                    textStyle: const TextStyle(
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                      fontSize: 32
                    ) // TextStyle
                  ), // Text
                ), // Center
                const SizedBox(height: 80), // block-size: 80, ⇒ height: 80, 🍌
                _emailAddress(),
                const SizedBox(height: 20), // block-size: 20, ⇒ height: 20, 🍌
                _password(),
                const SizedBox(height: 50), // block-size: 50, ⇒ height: 50, 🍌
                _signup(context),
              ],
            ), // Column
          ), // SingleChildScrollView
        ), // SafeArea
      ); // Scaffold
    );
  }

  Widget _emailAddress() {
    return Column(
      mainAxisAlignment: MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Email Address',
          style: GoogleFonts.raleway(
```

```

62       style: GoogleFonts.raleway(
63         textStyle: const TextStyle(
64           color: Colors.white,
65           fontWeight: FontWeight.normal,
66           fontSize: 16
67         ) // TextStyle
68       ),
69     ), // Text
70     const SizedBox(height: 16,), // block-size: 16, ⇒ height: 16, 🍌
71     TextField(
72       controller: _emailController,
73       decoration: InputDecoration(
74         filled: true,
75         hintText: 'xyz@gmail.com',
76         hintStyle: const TextStyle(
77           color: Color(0xff6A6A6A),
78           fontWeight: FontWeight.normal,
79           fontSize: 14
80         ), // TextStyle
81         fillColor: const Color(0xffF7F7F9) ,
82         border: OutlineInputBorder(
83           borderSide: BorderSide.none,
84           borderRadius: BorderRadius.circular(14)
85         ) // OutlineInputBorder
86       ), // InputDecoration
87     ) // TextField
88   ],
89 ); // Column
90 }
91
92 Widget _password() {
93   return Column(
94     mainAxisAlignment: MainAxisAlignment.start,
95     crossAxisAlignment: CrossAxisAlignment.start,
96     children: [
97       Text(
98         'Password',
99         style: GoogleFonts.raleway(
100           textStyle: const TextStyle(
101             color: Colors.white,
102             fontWeight: FontWeight.normal,
103             fontSize: 16
104           ) // TextStyle
105         ),
106       ), // Text
107       const SizedBox(height: 16,), // block-size: 16, ⇒ height: 16, 🍌
108       TextField(
109         controller: _passwordController,
110         obscureText: true,
111         decoration: InputDecoration(
112           filled: true,
113           fillColor: const Color(0xffF7F7F9) ,
114           border: OutlineInputBorder(
115             borderSide: BorderSide.none,
116             borderRadius: BorderRadius.circular(14)
117           ) // OutlineInputBorder
118         ), // InputDecoration
119       ) // TextField
120     ],
121   ); // Column

```

```

122     }
123
124     Widget _signup(BuildContext context) {
125       return ElevatedButton(
126         style: ElevatedButton.styleFrom(
127           backgroundColor: const Color(0xff0D6EFD),
128           shape: RoundedRectangleBorder(
129             borderRadius: BorderRadius.circular(14),
130           ), // RoundedRectangleBorder
131           minimumSize: const Size(double.infinity, 60),
132           elevation: 0,
133         ),
134         onPressed: () async {
135           await AuthService().signup(
136             email: _emailController.text,
137             password: _passwordController.text,
138             context: context
139           );
140         },
141         child: const Text("Sign Up", style: TextStyle(color: Colors.white)),
142       ); // ElevatedButton
143     }
144
145     Widget _signin(BuildContext context) {
146       return Padding(
147         padding: const EdgeInsets.only(bottom: 16), // inset-block-end: 16 ⇔ bottom: 16 🍷
148         child: RichText(
149           textAlign: TextAlign.center,
150           text: TextSpan(
151             children: [
152               const TextSpan(
153                 text: "Already Have Account? ",
154                 style: TextStyle(
155                   color: Colors.white,
156                   fontWeight: FontWeight.normal,
157                   fontSize: 16
158                 ), // TextStyle
159             ), // TextSpan
160               TextSpan(
161                 text: "Log In",
162                 style: const TextStyle(
163                   color: Colors.red,
164                   fontWeight: FontWeight.normal,
165                   fontSize: 16
166                 ), // TextStyle
167                 recognizer: TapGestureRecognizer().onTap = () {
168                   Navigator.push(
169                     context,
170                     MaterialPageRoute(
171                       builder: (context) => Login()
172                     ), // MaterialPageRoute
173                   );
174                 },
175               ), // TextSpan
176             ],
177           ), // TextSpan
178         ), // RichText
179       ); // Padding
180     }
181   }

```


Output:

<

WELCOME

Email Address

xyz@gmail.com

Password

Sign In

New User? [Create Account](#)

Explanation of code:

1. Importing Required Packages

- flutter/material.dart is used for UI components.
- flutter/gestures.dart is imported to handle gesture recognition for clickable text.
- google_fonts.dart is included to use custom fonts.
- firebase_auth_services.dart is used for Firebase authentication.
- login_page.dart is imported for navigation to the login screen.

2. Defining the SignupPage Class

- SignupPage is a **stateless widget**, meaning it does not maintain any state.
- Two TextEditingController objects are created to handle user input for email and password.

3. Building the UI with Scaffold

- The Scaffold provides the overall page structure.
- The background color is set to black.
- resizeToAvoidBottomInset: true ensures the UI adjusts when the keyboard appears.
- _signin(context) is used as the bottomNavigationBar to display a login option at the bottom.
- The AppBar is transparent with no elevation and a height of 50.

4. Using SafeArea and SingleChildScrollView

- SafeArea ensures content does not overlap with the system UI.
- SingleChildScrollView allows vertical scrolling to prevent UI overflow.
- padding: EdgeInsets.symmetric(horizontal: 16, vertical: 16) provides uniform spacing.

5. Adding a Title

- The title "Register Account" is centered and styled using GoogleFonts.raleway.
- The text is bold, white, and has a font size of 32.

6. Creating the Email Input Field

- _emailAddress() method returns a Column containing:
 - A label "**Email Address**" with white color and normal weight.
 - A TextField where users enter their email.
 - The field has a hint text "xyz@gmail.com" with a grey color.
 - The text field has rounded corners (borderRadius: 14) and a light fill color (#F7F7F9).

7. Creating the Password Input Field

- _password() method returns a Column containing:
 - A label "**Password**" styled similarly to the email label.
 - A TextField where users enter their password.
 - obscureText: true hides the entered characters for security.
 - The input field styling is the same as the email field.

8. Creating the Signup Button

- _signup(context) method returns an ElevatedButton:
 - The button has a blue background (#0D6EFD) and rounded corners (borderRadius: 14).
 - It has a height of **60 pixels** and expands to full width.

- When clicked, it calls the signup method from AuthService to register the user with Firebase.

9. Creating the Sign-in Option at the Bottom

- `_signin(context)` returns a RichText widget that displays:
 - "Already Have an Account?" in white.
 - "Log In" in red, which is clickable.
- TapGestureRecognizer is used to detect taps and navigate to the Login page.

10. Key OOP Concepts Used

- **Encapsulation:** The UI components are wrapped inside separate functions for better modularity.
- **Abstraction:** `_emailAddress()`, `_password()`, `_signup()`, and `_signin()` hide the implementation details.
- **Reusability:** The UI is structured into reusable methods to avoid redundancy.
- **Navigation:** Uses `Navigator.push()` for smooth screen transitions.

11. Conclusion

- This signup page provides a **modern UI** with Firebase authentication.
- The **scrollable layout** ensures usability on small screens.
- **It follows best practices for structuring UI code in Flutter.**

Login Page:

```
1
2 import 'package:flutter/gestures.dart';
3 import 'package:flutter/material.dart';
4 import 'package:google_fonts/google_fonts.dart';
5 import 'package:myproject/firebase_auth_services.dart';
6 import 'package:myproject/presentation/pages/sign_up_page.dart';
7
8
9 class Login extends StatelessWidget {
10   Login({super.key});
11
12   final TextEditingController _emailController = TextEditingController();
13   final TextEditingController _passwordController = TextEditingController();
14
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       backgroundColor: Colors.black,
19       resizeToAvoidBottomInset: true,
20       bottomNavigationBar: _signup(context),
21       appBar: AppBar(
22         backgroundColor: Colors.transparent,
23         elevation: 0,
24         toolbarHeight: 100,
25         leading: GestureDetector(
26           onTap: () {
27             Navigator.pop(context);
28           },
29           child: Container(
30             margin: EdgeInsets.only(left: 10),
31             decoration: BoxDecoration(
32               color: Color(0xffF7F7F9),
33               shape: BoxShape.circle
34             ), // BoxDecoration
35             child: const Center(
36               child: Icon(
37                 Icons.arrow_back_ios_new_rounded,
38                 color: Colors.black,
39               ), // Icon
40             ), // Center
41           ), // Container
42         ), // GestureDetector
43       ), // AppBar
44       body: SafeArea(
45         child: SingleChildScrollView(
46           padding: EdgeInsets.symmetric(horizontal: 16, vertical: 16),
47           child: Column(
48             mainAxisAlignment: MainAxisAlignment.start,
49             children: [
50               Center(
51                 child: Text(
52                   'WELCOME',
53                   style: GoogleFonts.raleway(
54                     textStyle: const TextStyle(
55                       color: Colors.white,
56                       fontWeight: FontWeight.bold,
57                       fontSize: 32
58                     ) // TextStyle
59                 ), // Text
60               ), // Center
61             ],
62           ),
63         ),
64       ),
65     );
66   }
67 }
```

```

62         _emailAddress(),
63         const SizedBox(height: 20), // block-size: 20, ⇒ height: 20, 🟡
64         _password(),
65         const SizedBox(height: 50), // block-size: 50, ⇒ height: 50, 🟡
66         _signin(context),
67     ],
68   ), // Column
69 ), // SingleChildScrollView
70 ), // SafeArea
71 ); // Scaffold
72 }
73
74
75 widget _emailAddress() {
76   return Column(
77     mainAxisAlignment: MainAxisAlignment.start,
78     crossAxisAlignment: CrossAxisAlignment.start,
79     children: [
80       Text(
81         'Email Address',
82         style: GoogleFonts.raleway(
83           textStyle: const TextStyle(
84             color: Colors.white,
85             fontWeight: FontWeight.normal,
86             fontSize: 16
87           ) // TextStyle
88         ), // Text
89       ),
90       const SizedBox(height: 16), // block-size: 16, ⇒ height: 16, 🟡
91       TextField(
92         controller: _emailController,
93         decoration: InputDecoration(
94           filled: true,
95           hintText: 'xyz@gmail.com',
96           hintStyle: const TextStyle(
97             color: Color(0xff6A6A6A),
98             fontWeight: FontWeight.normal,
99             fontSize: 14
100           ), // TextStyle
101           fillColor: const Color(0xffF7F7F9),
102           border: OutlineInputBorder(
103             borderSide: BorderSide.none,
104             borderRadius: BorderRadius.circular(14)
105           ) // OutlineInputBorder
106         ), // InputDecoration
107       ), // TextField
108     ],
109   ); // Column
110 }
111
112 widget _password() {
113   return Column(
114     mainAxisAlignment: MainAxisAlignment.start,
115     crossAxisAlignment: CrossAxisAlignment.start,
116     children: [
117       Text(
118         'Password',
119         style: GoogleFonts.raleway(
120           textStyle: const TextStyle(
121             color: Colors.white,
122             // ...

```

```

9      class Login extends StatelessWidget {
112      widget _password() {
122          color: Colors.white,
123          fontWeight: FontWeight.normal,
124          fontSize: 16
125        ) // TextStyle
126      ), // Text
127      const SizedBox(height: 16), // block-size: 16, ⇔ height: 16, 🍌
128      TextField(
129        obscureText: true,
130        controller: _passwordController,
131        decoration: InputDecoration(
132          filled: true,
133          fillColor: const Color(0xFF7F7F9),
134          border: OutlineInputBorder(
135            borderSide: BorderSide.none,
136            borderRadius: BorderRadius.circular(14)
137          ) // OutlineInputBorder
138        ), // InputDecoration
139      ) // TextField
140    ],
141  ); // Column
142  }
143
144  Widget _signin(BuildContext context) {
145    return ElevatedButton(
146      style: ElevatedButton.styleFrom(
147        backgroundColor: const Color(0xFF0D6EFD),
148        shape: RoundedRectangleBorder(
149          borderRadius: BorderRadius.circular(14),
150        ), // RoundedRectangleBorder
151        minimumSize: const Size(double.infinity, 60),
152        elevation: 0,
153      ),
154      onPressed: () async {
155        await AuthService().signin(
156          email: _emailController.text,
157          password: _passwordController.text,
158          context: context
159        );
160      },
161      child: const Text("Sign In", style: TextStyle(color: Colors.white)),
162    ); // ElevatedButton
163  }
164
165  Widget _signup(BuildContext context) {
166    return Padding(
167      padding: const EdgeInsets.only(bottom: 16), // inset-block-end: 16 ⇔ bottom: 16 🍌
168      child: RichText(
169        textAlign: TextAlign.center,
170        text: TextSpan(
171          children: [
172            const TextSpan(
173              text: "New User? ",
174              style: TextStyle(
175                color: Colors.white,
176                fontWeight: FontWeight.normal,
177                fontSize: 16
178              ), // TextStyle
179            ), // TextSpan
180            TextSpan(
181              text: "Create Account"

```

```

180       TextSpan(
181         text: "Create Account",
182         style: const TextStyle(
183           color: Colors.red,
184           fontWeight: FontWeight.normal,
185           fontSize: 16
186         ), // TextStyle
187         recognizer: TapGestureRecognizer()..onTap = () {
188           Navigator.push(
189             context,
190             MaterialPageRoute(
191               builder: (context) => SignupPage()
192             ), // MaterialPageRoute
193           );
194         }
195       ), // TextSpan
196     ]
197   ) // TextSpan
198 ), // RichText
199 ); // Padding
200 }
201 }

```

Output:

←

Register Account

Email Address

Password

Sign Up

Already Have Account? [Log In](#)

Explanation of code:

1. Importing Required Packages

- flutter/material.dart for UI components.
- flutter/gestures.dart for handling clickable text.
- google_fonts/google_fonts.dart for using custom fonts.
- firebase_auth_services.dart for authentication.
- sign_up_page.dart for navigation to the sign-up page.

2. Defining the Login Class

- Login is a **stateless widget**, meaning it does not maintain a state.
- Two TextEditingController objects are created for email and password input.

3. Building the UI with Scaffold

- The Scaffold is used to structure the page.
- backgroundColor: Colors.black sets a dark theme.
- resizeToAvoidBottomInset: true ensures the UI adjusts when the keyboard appears.
- _signup(context) is used as bottomNavigationBar to display a sign-up option.

4. AppBar with a Custom Back Button

- The AppBar is transparent with no elevation and a height of 100.
- A **custom back button** (a circular button with an arrow icon) is placed in the leading section.
- Navigator.pop(context) is called to go back to the previous screen when tapped.

5. Using SafeArea and SingleChildScrollView

- SafeArea ensures content does not overlap with the system UI.
- SingleChildScrollView prevents overflow issues when the keyboard appears.
- padding: EdgeInsets.symmetric(horizontal: 16, vertical: 16) provides spacing.

6. Adding a Welcome Title

- A centered **"WELCOME"** title with bold, white text and a font size of 32.

7. Creating the Email Input Field

- _emailAddress() method returns a Column:
 - Label **"Email Address"** styled in white.
 - TextField for user input with a **light fill color (#F7F7F9) and rounded corners**.
 - Placeholder text "xyz@gmail.com" in grey.

8. Creating the Password Input Field

- _password() method returns a Column:
 - Label **"Password"** styled in white.
 - TextField for password input with obscureText: true for security.
 - The styling matches the email input field.

9. Creating the Sign-in Button

- _signin(context) method returns an ElevatedButton:
 - Background color is **blue (#0D6EFD)**.
 - Button has rounded corners (borderRadius: 14) and expands full width.
 - Calls AuthService().signin() to authenticate the user using Firebase.

10. Creating the Sign-up Option at the Bottom

- _signup(context) returns a RichText widget that displays:

- "New User?" in white.
 - "Create Account" in red, which is clickable.
- TapGestureRecognizer is used to detect taps and navigate to the SignupPage().

11. Key OOP Concepts Used

- **Encapsulation:** UI components are modularized into separate methods.
- **Abstraction:** _emailAddress(), _password(), _signin(), and _signup() hide implementation details.
- **Reusability:** The UI components follow a modular structure.
- **Navigation:** Uses Navigator.push() for smooth screen transitions.

12. Conclusion

- This login page provides a **modern UI** with Firebase authentication.
- The **scrollable layout** ensures usability on small screens.
- It follows **best practices** for structuring Flutter UI code.

Homepage:

```

1  import 'package:flutter/material.dart';
2  import 'package:flutter_bloc/flutter_bloc.dart';
3  import 'package:myproject/presentation/bloc/car_bloc.dart';
4  import 'package:myproject/presentation/bloc/car_state.dart';
5  import 'package:myproject/presentation/widgets/car_card.dart';
6
7  class Homepage extends StatelessWidget {
8    const Homepage({super.key});
9
10   @override
11   Widget build(BuildContext context) {
12     return Scaffold(
13       appBar: AppBar(
14         title: const Center(
15           child: Text(
16             "CHOOSE YOUR CAR",
17             style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color: Colors.white),
18           ), // Text
19         ), // Center
20         backgroundColor: Colors.black,
21         foregroundColor: Colors.white,
22       ), // AppBar
23       body: SafeArea(
24         child: Padding(
25           padding: const EdgeInsets.symmetric(vertical: 8.0), // Adds spacing
26           child: BlocBuilder<CarBloc, CarState>(
27             builder: (context, state) {
28               if (state is CarLoading) {
29                 return const Center(
30                   child: CircularProgressIndicator(),
31                 ); // Center
32               } else if (state is CarLoaded) {
33                 // List of three different images

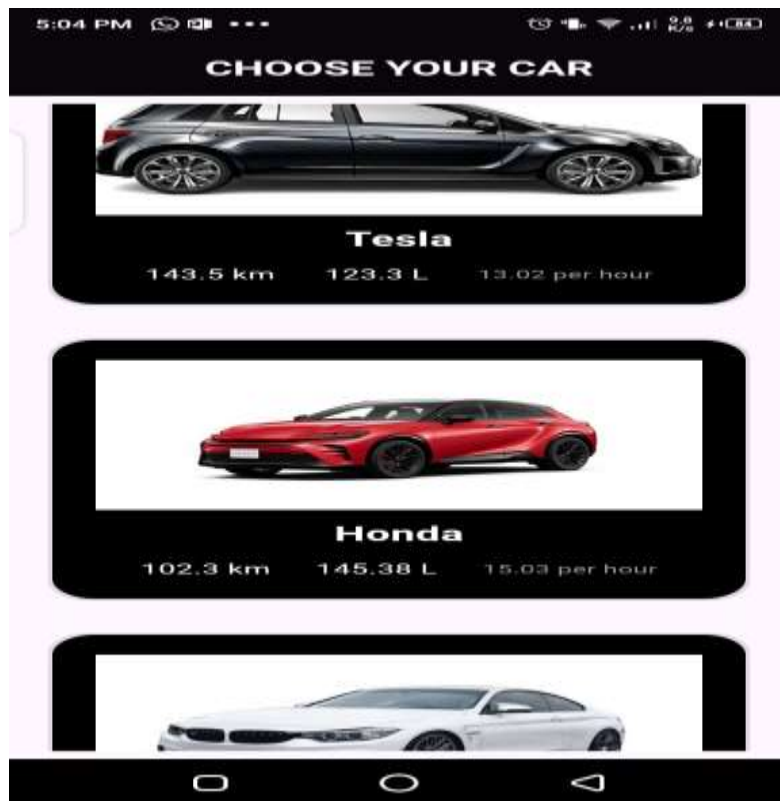
```

```

33 // List of three different images
34 List<String> carImages = [
35   "assets/homepage_car.jpg",
36   "assets/redcar.jpg",
37   "assets/whitecar2.jpg",
38 ];
39
40 return ListView.builder(
41   itemCount: state.cars.length,
42   itemBuilder: (context, index) {
43     return Padding(
44       padding: const EdgeInsets.symmetric(vertical: 8.0), // Adds spacing between cards
45       child: CarCard(
46         car: state.cars[index],
47         imageUrl: carImages[index % carImages.length],
48       ), // CarCard
49     ); // Padding
50   },
51 ); // ListView.builder
52 } else if (state is CarsError) {
53   return Center(
54     child: Text('ERROR: ${state.message}'),
55   ); // Center
56 }
57 return Container();
58 },
59 ); // BlocBuilder
60 ); // Padding
61 ); // SafeArea
62 ); // Scaffold

```

Output:



Explanation of code:

Homepage Widget Explanation

This Homepage widget in Flutter displays a list of cars using the **BLoC (Business Logic Component) pattern**, ensuring a clean separation between UI and business logic. It listens to state changes and updates the UI accordingly.

1. AppBar Configuration

- The AppBar has a **title ("CHOOSE YOUR CAR")**, centered in **bold white text**.
- The **background color is black**, while the foreground (icons) is white.

2. Using BLoC for State Management

- `BlocBuilder<CarBloc, CarState>` listens for state changes and updates the UI.
- The three possible states are:
 1. **CarsLoading** → Displays a loading indicator.
 2. **CarsLoaded** → Displays the list of cars.
 3. **CarsError** → Shows an error message.

3. Handling the Loading State

- If the state is `CarsLoading`, a `CircularProgressIndicator()` is displayed in the center.

4. Displaying the List of Cars

- When `CarsLoaded` state is received, a **list of cars** is displayed using `ListView.builder()`.
- A **list of three car images** is defined and cycled through to assign an image to each car.
- `CarCard` is used as a separate widget to display individual car details.

5. Handling Errors

- If the state is `CarsError`, an error message is displayed in the center of the screen.

6. Object-Oriented Programming Concepts in Use

- **Encapsulation** → The logic for fetching car data is handled in CarBloc, separate from the UI.
- **Abstraction** → The CarCard widget abstracts how each car is displayed.
- **Modularity** → The UI is cleanly separated from business logic, making the code more reusable.

Car Detail page:

```
lib > presentation > pages > car_detailpage.dart > _CarDetailsPageState >
1  import 'package:flutter/material.dart';
2  import 'package:myproject/presentation/pages/mapsfeaturespages.dart';
3  import 'package:myproject/presentation/widgets/more_car.dart';
4  import 'package:myproject/data/data_module/car.dart';
5
6  class CarDetailsPage extends StatefulWidget {
7    final Car car;
8
9    const CarDetailsPage({super.key, required this.car});
10
11    @override
12    State<CarDetailsPage> createState() => _CarDetailsPageState();
13  }
14
15  class _CarDetailsPageState extends State<CarDetailsPage> with SingleTickerProviderStateMixin {
16    late AnimationController _controller;
17    late Animation<double> _animation;
18
19    @override
20    void initState() {
21      super.initState();
22
23      _controller = AnimationController(
24        duration: const Duration(seconds: 3),
25        vsync: this,
26      ); // AnimationController
27
28      _animation = Tween<double>(begin: 1.0, end: 1.5).animate(_controller)
29        ..addListener(() {
30          setState(() {});
31        });
32
33      _controller.forward();
34    }
35
36    @override
37    void dispose() {
38      _controller.dispose();
39      super.dispose();
40    }
41
42    @override
43    Widget build(BuildContext context) {
44      return Scaffold(
45        appBar: AppBar(
46          backgroundColor: Colors.black,
47          title: const Row(
48            mainAxisAlignment: MainAxisAlignment.center,
49            children: [
50              Icon(Icons.info_outline, color: Colors.white),
51              SizedBox(width: 10),
52              Text(
53                'Information',
54                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold, color: Colors.white),
55              ), // Text
56            ],
57          ), // Row
58        ), // AppBar
59        body: SingleChildScrollView(
60          child: Padding(
61            padding: const EdgeInsets.all(20),
62            child: Column(
```

```

15 class _CardDetailsPageState extends State<CardDetailsPage> with SingleTickerProviderStateMixin {
16   widget build(BuildContext context) {
17     padding: const EdgeInsets.all(20),
18     child: Column(
19       children: [
20         const SizedBox(height: 30), // block-size: 20 == height: 20 🟡
21
22         /// Avatar & Map Section
23         SizedBox(
24           width: double.infinity, // Ensures the row takes full width // inline-size: double.infinity, == width: double.infinity, 🟡
25           child: Row(
26             mainAxisAlignment: MainAxisAlignment.spaceBetween,
27             children: [
28               /// Avatar Box
29               Flexible(
30                 child: Container(
31                   height: 200, // block-size: 200, == height: 200, 🟡
32                   padding: const EdgeInsets.all(20),
33                   decoration: BoxDecoration(
34                     color: const Color(0xfffff3f3),
35                     borderRadius: BorderRadius.circular(20),
36                     boxShadow: const [
37                       BoxShadow(
38                         color: Colors.black12,
39                         blurRadius: 10,
40                         spreadRadius: 5,
41                       ) // BoxShadow
42                     ],
43                   ), // BoxDecoration
44                   child: const Column(
45                     mainAxisAlignment: MainAxisAlignment.center,
46                     children: [
47                       CircleAvatar(
48                         radius: 40,
49                         backgroundImage: AssetImage('assets/avatar.jpg'),
50                       ), // CircleAvatar
51                       SizedBox(height: 10), // block-size: 10 == height: 10 🟡
52                       Text(
53                         'Alexander',
54                         style: TextStyle(fontWeight: FontWeight.bold),
55                       ), // Text
56                       Text(
57                         '$ 4.255',
58                         style: TextStyle(color: Colors.grey),
59                       ), // Text
60                     ],
61                   ), // Column
62                 ), // Container
63               ), // Flexible
64
65               const SizedBox(width: 10), // Add some spacing // inline-size: 10 == width: 10 🟡
66
67               /// Map Image with Animation
68               Flexible(
69                 child: GestureDetector(
70                   onTap: () {
71                     Navigator.push(
72                       context,
73                       MaterialPageRoute(
74                         builder: (context) => MapDetailsPage(car: widget.car),
75                       ), // MaterialPageRoute
76                     );
77                   },
78                 ),
79               ),
80             ],
81           ),
82         ),
83       ],
84     ),
85   }
86 }

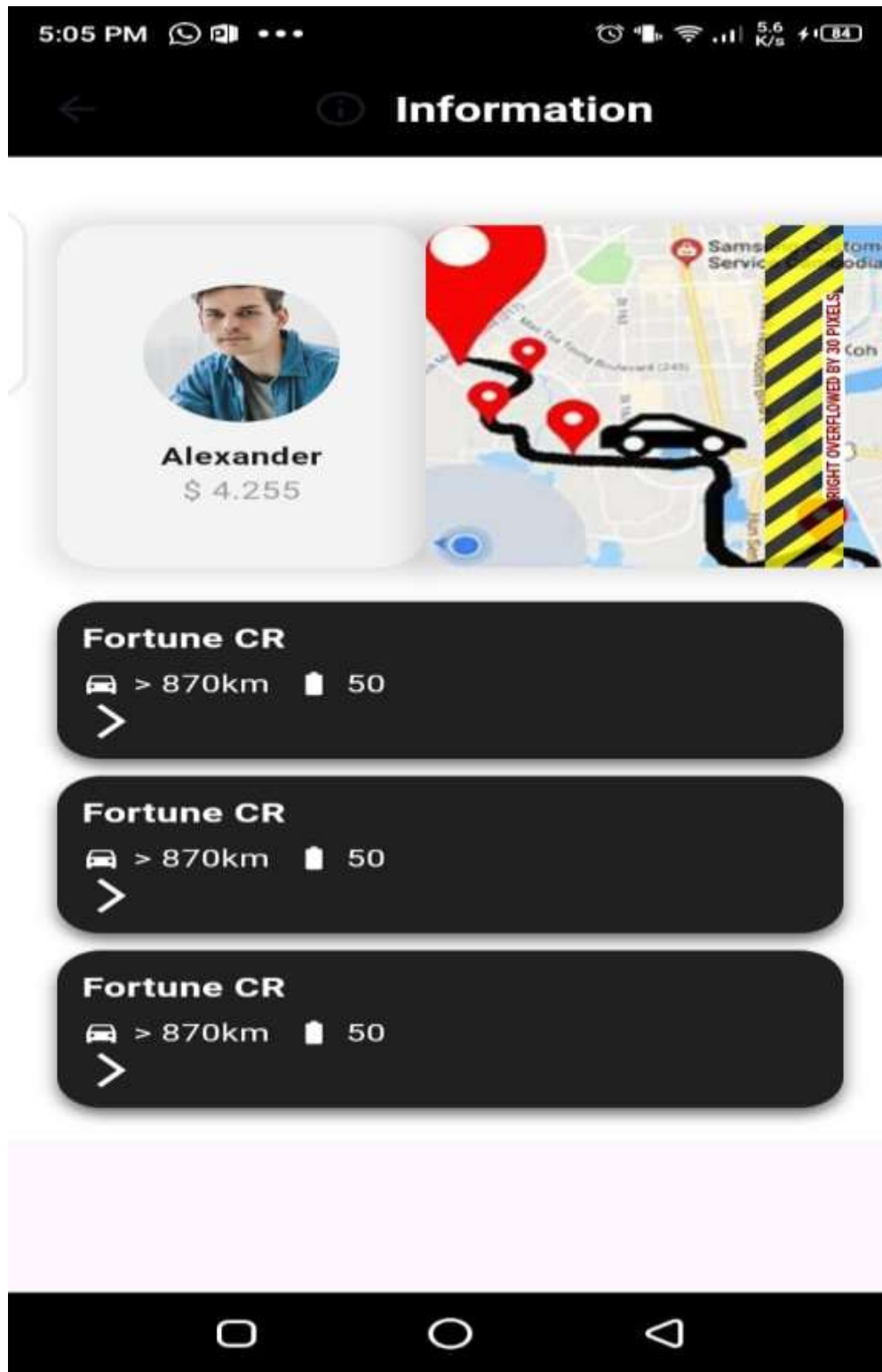
```

```

119         ), // MaterialPageRoute
120     );
121 },
122 child: Container(
123     height: 200, // block-size: 200, ⇒ height: 200, 📏
124     decoration: BoxDecoration(
125         borderRadius: BorderRadius.circular(15),
126         boxShadow: const [
127             BoxShadow(
128                 color: Colors.black12,
129                 blurRadius: 10,
130                 spreadRadius: 5,
131             ) // BoxShadow
132         ],
133     ), // BoxDecoration
134     child: ClipRect(
135         clipBehavior: Clip.antiAlias,
136         borderRadius: BorderRadius.circular(15),
137         child: Transform.scale(
138             scale: _animation.value,
139             alignment: Alignment.center,
140             child: Image.asset(
141                 'assets/map.jpg',
142                 fit: BoxFit.cover,
143                 width: double.infinity, // inline-size: double.infinity, ⇒ width: double.infinity, 📏
144             ), // Image.asset
145         ), // Transform.scale
146     ), // ClipRect
147 ), // Container
148 ), // GestureDetector
149 ), // Flexible
150 ],
151 ), // Row
152 ), // SizedBox
153
154 const SizedBox(height: 10), // block-size: 10 ⇒ height: 10 📏
155
156 /// List of More Cars
157 column(
158     children: [
159         MoreCard(
160             car: Car(model: "Fortune CR", distance: "870", fuel_capacity: "50", price_per_hour: "45"),
161         ), // MoreCard
162         const SizedBox(height: 10), // block-size: 10 ⇒ height: 10 📏
163         MoreCard(
164             car: Car(model: "Fortune CR", distance: "870", fuel_capacity: "50", price_per_hour: "45"),
165         ), // MoreCard
166         const SizedBox(height: 10), // block-size: 10 ⇒ height: 10 📏
167         MoreCard(
168             car: Car(model: "Fortune CR", distance: "870", fuel_capacity: "50", price_per_hour: "45"),
169         ), // MoreCard
170     ],
171 ), // Column
172 ],
173 ), // Column
174 ), // Padding
175 ), // SingleChildScrollView
176 ); // Scaffold
177 }
178 }

```

Output:



Explanation:

1. Overview

The CarDetailsPage is a **stateful widget** that displays detailed information about a specific car, including its profile (avatar), map view (animated), and a list of similar cars.

2. Constructor & State Management

- **Constructor:** The page accepts a Car object as an argument (required this.car), representing the selected car for which the details will be shown.
- **Stateful Widget:** It uses StatefulWidget because it manages an animation, making it necessary to rebuild the widget whenever the animation value changes.

3. Animation Controller

- **Animation Controller:** An AnimationController named `_controller` is set to run for **3 seconds**.
- **Animation:** A Tween<double> is used to scale the map image from **1.0 to 1.5**, which is applied to the Transform.scale widget for the map image.
- The animation is **triggered in initState()** and updates the UI every time the value of `_animation` changes, causing the map to "zoom" in.

4. AppBar

- The AppBar has a centered title "Information" with an **info icon** next to it.
- It has a black background, with white text and icons for contrast.

5. Body Layout

The body is wrapped in a SingleChildScrollView to allow for scrolling when content overflows. The content is divided into three main sections:

A. Avatar & Map Section

- **Avatar Box:**
 - Contains a CircleAvatar with a profile picture (assets/avator.jpg), a name ("Alexander"), and a price (\\$ 4.255).
 - It's displayed inside a Container with **rounded corners**, padding, and a subtle **box shadow** for a neat appearance.
- **Map Image (with Animation):**
 - A **map image** (assets/map.jpg) is displayed and animated using the `_animation` value (scale from 1.0 to 1.5).
 - The map image is clickable using GestureDetector, and when clicked, it navigates to the MapDetailsPage to show more details related to the map.

B. List of More Cars

- This section presents a list of more cars that are similar to the selected one. It uses the MoreCard widget to display each car's information (like model, distance, fuel capacity, and price per hour).

6. Object-Oriented Programming (OOP) Concepts

- **Encapsulation:** The Car object encapsulates all relevant car details (model, price, etc.), making it easier to pass the data around the app without exposing unnecessary implementation details.
- **State Management:** The State class manages the widget's state, including the animation controller. This ensures the UI can be updated whenever the animation progresses.
- **Inheritance:** CarDetailsPage extends StatefulWidget, which is a part of Flutter's widget framework. This inheritance allows it to handle state and UI updates.

Map Features page:

```
1  import 'package:flutter/material.dart';
2  import 'package:flutter_map/flutter_map.dart';
3  import 'package:latlong2/latlong.dart';
4  import 'package:myproject/data/data_module/car.dart';
5  import 'package:myproject/presentation/widgets/carddetail.dart';
6
7  class MapDetailsPage extends StatelessWidget {
8
9    final Car car;
10
11    const MapDetailsPage({super.key, required this.car});
12    @override
13    Widget build(BuildContext context) {
14      return Scaffold(
15        //
16        appBar: AppBar(
17          backgroundColor: Colors.transparent,
18          leading: IconButton(onPressed: () {
19            // Navigator.push(context);
20            // icon: const Icon(Icons.arrow_back)),
21          },),
22        body: Stack(
23          children: [
24            FlutterMap(
25              options: const MapOptions(
26                initialCenter: LatLng(51.0, -0.09),
27                initialZoom: 13,
28                minZoom: 2,
29                maxZoom: 19,
30              ), // MapOptions
31              children: [
32                TileLayer(
33                  urlTemplate: "https://(s).tile.openstreetmap.org/{z}/{x}/{y}.png",
34                  subdomains: const ['a', 'b', 'c'],
35                ), // TileLayer
36              ], // FlutterMap
37            Positioned(
38              bottom: 7.0, // inset-block-end: 7.0, == bottom: 7.0,
39              left: 0, // inset-inline-start: 0, == left: 0,
40              right: 0, // inset-inline-end: 0, == right: 0,
41              child: CardDetailCard(car: car), // Positioned
42            ), // Stack
43          ], // Scaffold
44        ),
45      );
46    }
47  }
```

Output:



Explanation of code:

The MapDetailsPage widget is designed to show a map using FlutterMap with OpenStreetMap tiles, and also displays the details of a specific car at the bottom of the map. Here's a breakdown of its components:

1. Constructor

- The MapDetailsPage accepts a Car object (required this.car) which contains information about the car that will be displayed at the bottom of the map.

2. Body Layout

- The body of the page is wrapped inside a **Stack** widget. This allows the map to be displayed in the background and the car details card (cardetailiscard) to be positioned on top of the map.

A. FlutterMap

- **FlutterMap:** This is used to render the map. It uses the latlong2 package to handle geographical coordinates and MapOptions for map settings.
 - **initialCenter:** Sets the initial center of the map (latitude 51.0, longitude -0.09), which is somewhere near London.
 - **initialZoom:** The initial zoom level is set to 13, which offers a detailed view of the area.
 - **minZoom and maxZoom:** These properties restrict the zoom range of the map.
- **TileLayer:** It defines the map tiles used for rendering the map. The URL template <https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png> retrieves OpenStreetMap tiles, and subdomains specifies different subdomains for better load distribution.

B. Positioned Widget

- The Positioned widget is used to position the **car details card** at the bottom of the screen.
 - **bottom: 7.0:** The car details card is positioned 7 pixels from the bottom.
 - **left: 0 and right: 0:** It stretches the card across the entire width of the screen.

C. Car Details Card

- The cardetailiscard widget is used to display the car's details, like the car model, price, or other properties. This widget is passed the car object so it can show relevant information about the selected car.

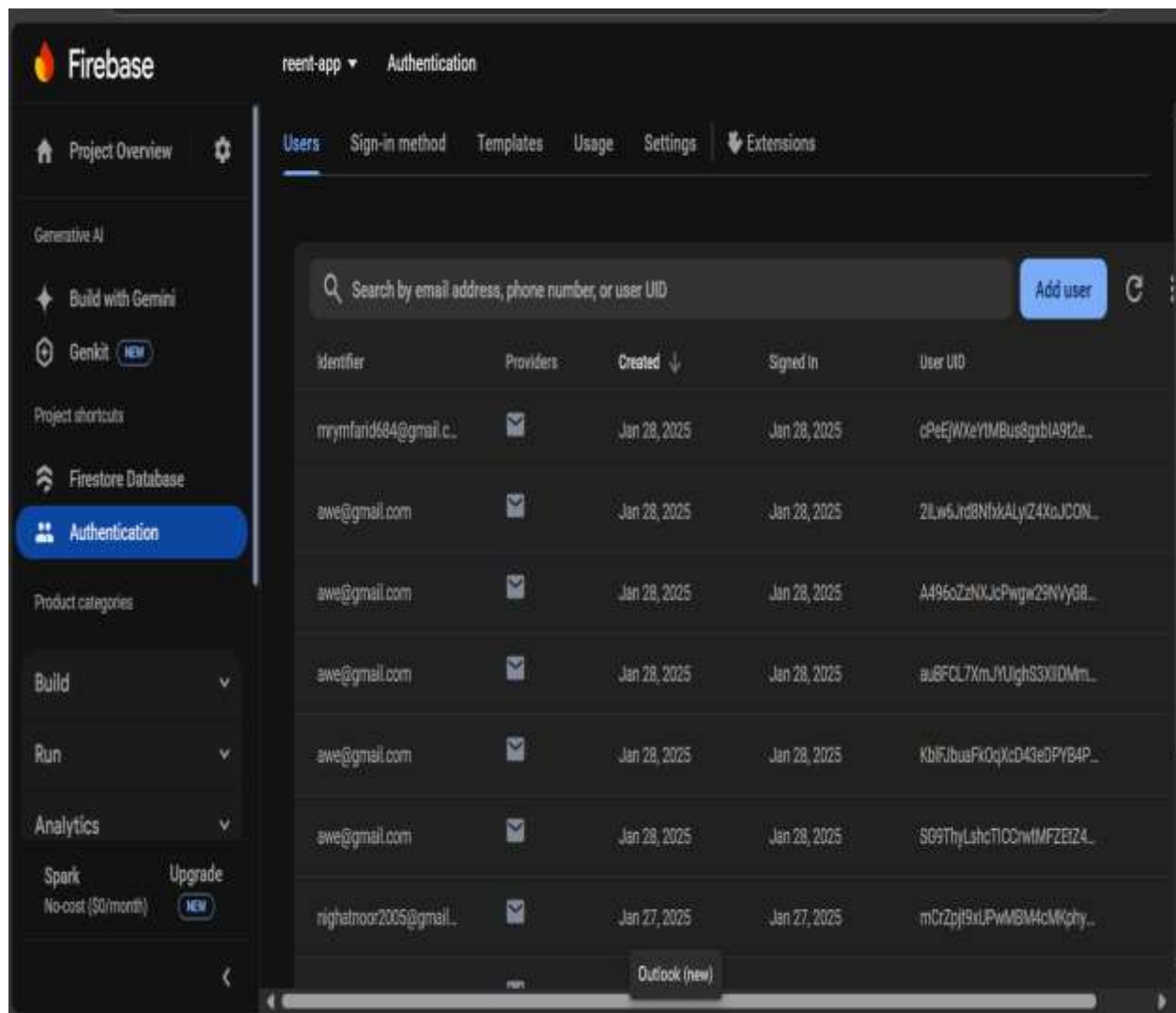
3. Future Improvements

- You may want to pass in the actual location of the car by updating the initialCenter to use the car's latitude and longitude if available. You can get this from the Car object and update the map's initial position accordingly.
- You can also add **markers** on the map to represent car locations dynamically.

Summary

This page displays an interactive map and the details of a car in a stacked layout. It uses FlutterMap to load OpenStreetMap tiles and places the car details card on top of the map. The page is simple and uses basic positioning with a stack to manage overlapping widgets.

Firestore Authentication:



Firebase Authentication:

Firebase Authentication is handling user sign-in and registration in app. Based on the image:

- We have multiple users registered, with email-based authentication (@gmail.com email addresses).
- Each user has a **unique user identifier (UID)** assigned by Firebase.
- The table shows:
 - **Identifier:** The email address used for login.
 - **Providers:** Email authentication is being used.
 - **Created:** The date when the user registered.
 - **Signed In:** Last sign-in date.
 - **User UID:** A unique identifier for each user.

How it Works in App:

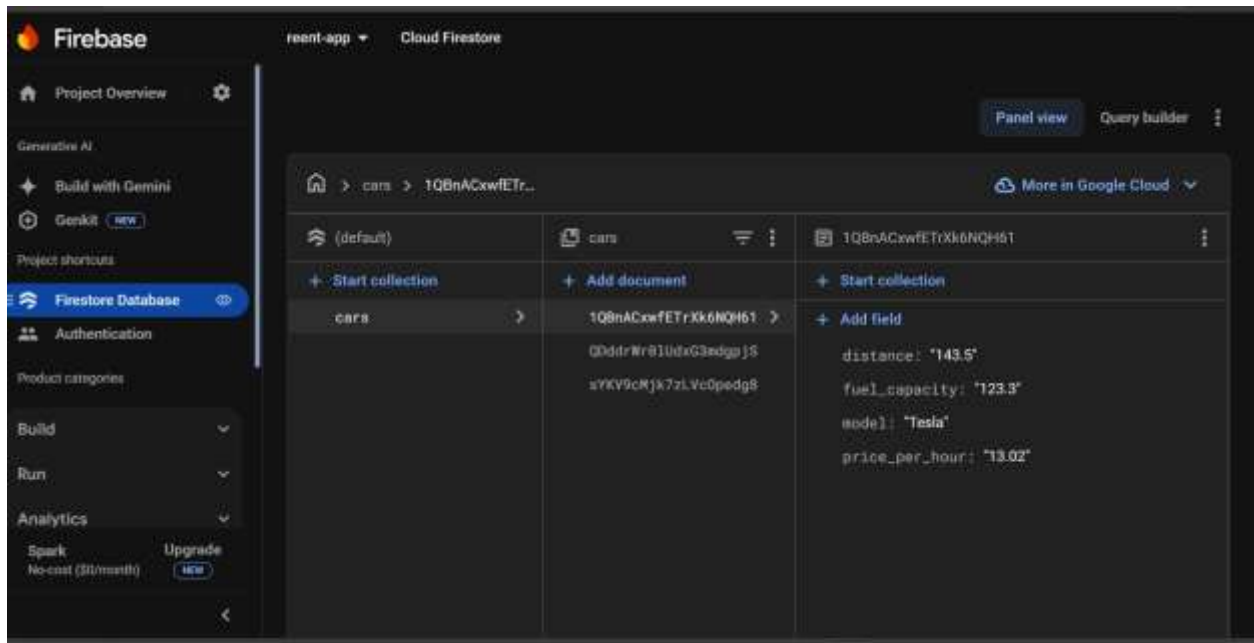
- When a user signs up, Firebase creates an account and assigns a **UID**.
- This UID can be used to associate user data in **Firestore**.
- We can retrieve user details using `FirebaseAuth.instance.currentUser`.

Firebase data base:

. Cloud Firestore Database :

Our app stores car-related data in Firestore. The structure in the image shows:

- A **collection** named "cars".
- Inside this collection, there are **documents** with unique IDs (1QBnACxwFETrXk6NQH61).
- Each document represents a **car** and contains fields:
 - **model:** "Tesla" (Car model)
 - **distance:** "143.5" (Distance covered)
 - **fuel_capacity:** "123.3" (Fuel capacity)
 - **price_per_hour:** "13.02" (Rental price per hour)



How These Connect to Your Code:

- CarBloc is handling Firebase data retrieval.
- CarDetailsPage displays car info, including an animation and map feature (MapDetailsPage).
- Homepage lists cars dynamically, using Firestore images.
- The authentication system ensures that only signed-in users access features.

Additional Changes:

Car Images Update:

- **Previous Implementation:** The homepage previously displayed the same car image repeated multiple times for each car listed.
- **Change Implemented:** To enhance the visual appeal and provide variety, three distinct car images were added. These images are displayed dynamically, with each car in the list associated with a different image.
- **Reason for the Change:** This change was made to improve the user experience by making the car options more visually diverse and engaging. By using different images, users can better distinguish between the cars available for selection.
- **Implementation:** The `ListView.builder` was updated to assign a different image to each car in the list using a cyclic assignment. A list of three images was created, and each car was assigned one of the images using the modulus operator (%) to cycle through them.

Integration of Sign-In and Sign-Up Pages with Firebase Authentication:

- **Previous Implementation:** The app previously featured an onboarding page followed by the homepage, with no user authentication process.
- **Change Implemented:** A new sign-in and sign-up page was introduced, integrating Firebase Authentication for user login and registration.
- **Reason for the Change:** This change was made to allow users to securely log in or register for the app using Firebase Authentication. Implementing this feature enhances the security and personalization of the app by allowing only authenticated users to access the main content.
- **Implementation:** The onboarding page was removed and replaced with a sign-in screen, where users can either sign in with their credentials or sign up for a new account. Firebase Authentication was utilized to handle user authentication, ensuring seamless login and registration experiences. The flow is now as follows: the user first encounters the sign-in page; upon successful authentication, they are redirected to the homepage.

Image Size Management Across the App:

- **Previous Implementation:** Images in the app were not consistently managed in terms of size, which sometimes led to overflow or improper display of content.
- **Change Implemented:** Image sizes were controlled and adjusted across the entire app to ensure that they fit appropriately within the layout and prevent overflow.
- **Reason for the Change:** This change was made to maintain a consistent and responsive design, ensuring that images are displayed correctly on different screen sizes without causing layout issues or overflow.
- **Implementation:** The size of images was dynamically managed using constraints, such as `BoxFit`, and `SizedBox`, ensuring that each image adapts well to the screen size. This ensures images scale appropriately without disrupting the overall layout of the app, providing a better user experience across different devices.

The globally defined widgets in our app:

Firebase_auth –services.dart:

lib >  firebase_auth_services.dart >  AuthService >  signup

```
1 import 'package:firebase_auth/firebase_auth.dart';
2 import 'package:flutter/material.dart';
3 import 'package:fluttertoast/fluttertoast.dart';
4 import 'package:myproject/presentation/pages/homepages.dart';
5 import 'package:myproject/presentation/pages/login_page.dart';
6
7
8
9 class AuthService {
10
11   Future<void> signup([
12     required String email,
13     required String password,
14     required BuildContext context
15   ]) async {
16
17     try {
18
19       await FirebaseAuth.instance.createUserWithEmailAndPassword(
20         email: email,
21         password: password
22       );
23
24       await Future.delayed(const Duration(seconds: 1));
25       Navigator.pushReplacement(
26         context,
27         MaterialPageRoute(
28           builder: (BuildContext context) => const Homepage(),
29         ) // MaterialPageRoute
30       );
31
32     } on FirebaseAuthException catch(e) {
33       String message = '';
34       if (e.code == 'weak-password') {
35         message = 'The password provided is too weak.';
36       } else if (e.code == 'email-already-in-use') {
37         message = 'An account already exists with that email.';
38       }
39       Fluttertoast.showToast(
40         msg: message,
41         toastLength: Toast.LENGTH_LONG,
42         gravity: ToastGravity.SNACKBAR,
43         backgroundColor: Colors.black54,
44         textColor: Colors.white,
45         fontSize: 14.0,
46       );
47     }
48     catch(e){
49
50     }
51
52   }
53
54   Future<void> signin([
55     required String email,
56     required String password,
57     required BuildContext context
58   ]) async {
59
60     try {
61
62       await FirebaseAuth.instance.signInWithEmailAndPassword(
```



```

123     email: email,
124     password: password
125   );
126
127   await Future.delayed(const Duration(seconds: 1));
128   Navigator.pushReplacement(
129     context,
130     MaterialPageRoute(
131       builder: (BuildContext context) => const HomeScreen()
132     ) // MaterialPageRoute
133   );
134
135   // FirebaseAuthException catch(e) {
136   String message = '';
137   if (e.code == 'invalid-email') {
138     message = 'No user found for that email.';
139   } else if (e.code == 'invalid-credential') {
140     message = 'Wrong password provided for that user.';
141   }
142   Fluttertoast.showToast(
143     msg: message,
144     toastLength: Toast.LENGTH_LONG,
145     gravity: ToastGravity.BOTTOM,
146     backgroundColor: Colors.black54,
147     textColor: Colors.white,
148     fontSize: 16.0,
149   );
150   }
151   catch(e){
152
153   }
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Explanation of AuthService (User-Defined Widget for Authentication):


The AuthService class is a custom authentication service that manages user authentication using Firebase in a Flutter application. It includes three main functions:

1. **Sign-Up Function (signup)**
 - This function registers a new user with an email and password using Firebase Authentication.
 - If the registration is successful, the user is redirected to the homepage.
 - Error handling is implemented to display messages if the password is too weak or if the email is already registered.
 - The function provides feedback using Fluttertoast to notify users of any authentication issues.
2. **Sign-In Function (signin)**
 - This function allows an existing user to log in using their email and password.
 - If authentication is successful, the user is navigated to the homepage.
 - Error handling ensures that appropriate messages are displayed if the email is not found or if the password is incorrect.
 - Similar to the sign-up function, Fluttertoast is used for instant feedback.
3. **Sign-Out Function (signout)**
 - This function logs out the currently signed-in user.
 - After signing out, the user is redirected to the login page.
 - A delay is added before navigation to ensure a smooth transition.

Features of the AuthService Class:Key

- **Encapsulates authentication logic**, making it reusable across the app.
- **Uses Firebase Authentication** for secure login, sign-up, and logout functionality.
- **Implements error handling** to guide users in case of authentication failures.
- **Provides feedback through Fluttertoast**, enhancing user experience.
- **Utilizes navigation** to manage user flow after authentication actions.

firebase_options.dart:

```
lib >  firebase_options.dart > ...
1  // File generated by FlutterFire CLI.
2  // ignore_for_file: type=lint
3  import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
4  import 'package:flutter/foundation.dart'
5  |   show defaultTargetPlatform, kIsWeb, TargetPlatform;
6
7  /// Default [FirebaseOptions] for use with your Firebase apps.
8  ///
9  /// Example:
10 /// ```dart
11 /// import 'firebase_options.dart';
12 /// // ...
13 /// await Firebase.initializeApp(
14 ///   options: DefaultFirebaseOptions.currentPlatform,
15 /// );
16 /// ```
17 class DefaultFirebaseOptions {
18   static FirebaseOptions get currentPlatform {
19     if (kIsWeb) {
20       return web;
21     }
22     switch (defaultTargetPlatform) {
23       case TargetPlatform.android:
24         return android;
25       case TargetPlatform.iOS:
26         return ios;
27       case TargetPlatform.macOS:
28         return macos;
29       case TargetPlatform.windows:
30         return windows;
31       case TargetPlatform.linux:
32         throw UnsupportedError(
33           'DefaultFirebaseOptions have not been configured for linux - '
34           'you can reconfigure this by running the FlutterFire CLI again.',
35         );
36       default:
37         throw UnsupportedError(
38           'DefaultFirebaseOptions are not supported for this platform.',
39         );
40     }
41   }
42
43   static const FirebaseOptions web = FirebaseOptions(
44     apiKey: 'AIzaSyDy6eySePk7s6BA5lXt5yeUGwwB8whKs8c',
45     appId: '1:325480975565:web:5c9e34a7c1918de2f6ed84',
46     messagingSenderId: '325480975565',
47     projectId: 'reent-app',
48     authDomain: 'reent-app.firebaseio.com',
49     storageBucket: 'reent-app.firebaseio.com',
50     measurementId: 'G-EMGQT755Q4',
51   );
52 }
```

```

lib > firebase_options.dart > ...
17 class DefaultFirebaseOptions {
41 }
42
43 static const FirebaseOptions web = FirebaseOptions(
44   apiKey: 'AIzaSyDy6ey5ePk7s6BA5lXt5yeUGwwB8whKs8c',
45   appId: '1:325480975565:web:5c9e34a7c1918de2f6ed84',
46   messagingSenderId: '325480975565',
47   projectId: 'reent-app',
48   authDomain: 'reent-app.firebaseio.com',
49   storageBucket: 'reent-app.firebaseio.com',
50   measurementId: 'G-EMGQT755Q4',
51 );
52
53 static const FirebaseOptions android = FirebaseOptions(
54   apiKey: 'AIzaSyB9ZlSetwFrX2MwwNM9sfPk7K3vzHQandc',
55   appId: '1:325480975565:android:69fab055c052d6a5f6ed84',
56   messagingSenderId: '325480975565',
57   projectId: 'reent-app',
58   storageBucket: 'reent-app.firebaseio.com',
59 );
60
61 static const FirebaseOptions ios = FirebaseOptions(
62   apiKey: 'AIzaSyCpsfJivmxYaT2yxoIz4FWRY2RTuPx1oQA',
63   appId: '1:325480975565:ios:08823e33d0c2f967f6ed84',
64   messagingSenderId: '325480975565',
65   projectId: 'reent-app',
66   storageBucket: 'reent-app.firebaseio.com',
67   iosBundleId: 'com.example.myproject',
68 );
69
70 static const FirebaseOptions macos = FirebaseOptions(
71   apiKey: 'AIzaSyCpsfJivmxYaT2yxoIz4FWRY2RTuPx1oQA',
72   appId: '1:325480975565:ios:08823e33d0c2f967f6ed84',
73   messagingSenderId: '325480975565',
74   projectId: 'reent-app',
75   storageBucket: 'reent-app.firebaseio.com',
76   iosBundleId: 'com.example.myproject',
77 );
78
79 static const FirebaseOptions windows = FirebaseOptions(
80   apiKey: 'AIzaSyDy6ey5ePk7s6BA5lXt5yeUGwwB8whKs8c',
81   appId: '1:325480975565:web:598976118191d6aef6ed84',
82   messagingSenderId: '325480975565',
83   projectId: 'reent-app',
84   authDomain: 'reent-app.firebaseio.com',
85   storageBucket: 'reent-app.firebaseio.com',
86   measurementId: 'G-E0W8ECV6DS',
87 );
88 }
89

```

Explanation of firebase_options.dart (User-Defined Widget in Flutter):

The `firebase_options.dart` file is a **user-defined class** that provides Firebase configuration settings for different platforms in a Flutter app. It is generated automatically using the **FlutterFire CLI** and is essential for initializing Firebase services like authentication, Firestore, and storage.

This file contains a class called `DefaultFirebaseOptions`, which selects the correct Firebase settings based on the platform (Web, Android, iOS, macOS, Windows). It ensures that the app connects to Firebase using the appropriate API keys and project details without requiring manual setup for each platform.

The class determines the **target platform** using `defaultTargetPlatform` and `kIsWeb`. If an unsupported platform (like Linux) is detected, it throws an error. The Firebase configurations for each platform are stored as `FirebaseOptions` objects containing parameters such as `apiKey`, `appId`, `projectId`, and `storageBucket`.

To use this configuration in the Flutter app, Firebase is initialized using `DefaultFirebaseOptions.currentPlatform`. This ensures that Firebase works correctly on different platforms while keeping the setup **clean, modular, and scalable**.

Injection_container.dart:

```
lib > injection_container.dart > ...
1  import 'package:cloud_firestore/cloud_firestore.dart';
2  import 'package:get_it/get_it.dart';
3  import 'package:myproject/data/datasources/firebase_car_data_source.dart';
4  import 'package:myproject/data/repositories/car_repository_impl.dart';
5  import 'package:myproject/domain/repositories/car_repository.dart';
6  import 'package:myproject/domain/usecases/get_cars.dart';
7  import 'package:myproject/presentation/bloc/car_bloc.dart';
8  GetIt getIt=GetIt.instance;
9  void initInjection(){
10   try{
11     getIt.registerLazySingleton<FirebaseFirestore>(()=>FirebaseFirestore.instance);
12     getIt.registerLazySingleton<FirebaseCarDataSource>(()=>FirebaseCarDataSource(firebase: getIt<FirebaseFirestore>()));
13     getIt.registerLazySingleton<CarRepository>(()=>CarRepositoryImpl(getIt<FirebaseCarDataSource>()));
14     getIt.registerLazySingleton<GetCars>(()=>GetCars(getIt<CarRepository>()));
15     getIt.registerFactory(()=>CarBloc(getCars: getIt<GetCars>()));
16   }
17   catch(e){
18     throw e;
19   }
20 }
21
22
23
24
25
26
27 }
```

Explanation of Dependency Injection Using GetIt in Flutter:

The given code implements dependency injection in a Flutter project using the `GetIt` package. Dependency injection helps manage dependencies centrally, reducing tight coupling and improving modularity, scalability, and testability.

1. Purpose of initInjection()

The `initInjection()` function registers dependencies using `GetIt`, ensuring that required instances are created and managed efficiently throughout the app.

2. Key Components

- **GetIt `getIt = GetIt.instance;`**
 - `GetIt.instance` provides a global service locator to manage dependencies.
- **Dependency Registration:**
 - `registerLazySingleton`: Creates a **single instance** of a class that is reused whenever required.
 - `registerFactory`: Creates a **new instance** whenever needed.

Main.dart:

```
lib > main.dart > ...
1  //import 'package:cloud_firestore/cloud_firestore.dart';
2  import 'package:firebase_core/firebase_core.dart';
3  import 'package:flutter/material.dart';
4  import 'package:flutter_bloc/flutter_bloc.dart';
5  import 'package:myproject/firebase_options.dart';
6  import 'package:myproject/injection_container.dart';
7  import 'package:myproject/presentation/bloc/car_bloc.dart';
8  import 'package:myproject/presentation/bloc/car_event.dart';
9  import 'package:myproject/presentation/pages/onboarding_page.dart';
10
11  Run | Debug | Profile
12  Future<void> main() async {
13    WidgetsFlutterBinding.ensureInitialized();
14    await Firebase.initializeApp(
15      options: DefaultFirebaseOptions.currentPlatform
16    );
17    /*
18    WidgetsFlutterBinding.ensureInitialized();
19    await Firebase.initializeApp(
20      options: DefaultFirebaseOptions.currentPlatform
21    );
22    */
23
24    initInjection();
25    runApp(const MyApp());
26  }
27
28  class MyApp extends StatelessWidget {
29    const MyApp({super.key});
30
31    // This widget is the root of your application.
32    @override
33    Widget build(BuildContext context) {
34      return BlocProvider(
35        create: (_) => getIt<CarBloc>().add(LoadCars()),
36        child: MaterialApp(
37          debugShowCheckedModeBanner: false,
38          title: 'Flutter Demo',
39          theme: ThemeData(
40
41            colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
42            useMaterial3: true,
43          ), // ThemeData
44          home: const OnboardingPage(),
45        ), // MaterialApp
46      ); // BlocProvider
47    }
48  }
49
```

Explanation of the Main Flutter App Initialization and BLoC Setup:

This code demonstrates the initialization of a Flutter app with Firebase, dependency injection using GetIt, and the BLoC (Business Logic Component) pattern to manage state. Below is a detailed breakdown of the code:

1. Firebase Initialization

In the `main()` function, Firebase is initialized before the app starts:

- **`Firebase.initializeApp()`** is called to initialize Firebase services, using platform-specific configuration from `DefaultFirebaseOptions.currentPlatform`.
- **`WidgetsFlutterBinding.ensureInitialized()`** ensures that the Flutter framework is fully initialized before performing any asynchronous operations, such as Firebase initialization.

This setup ensures that Firebase is ready to be used across different platforms (Android, iOS, Web, etc.).

2. Dependency Injection Setup

The **`initInjection()`** function is invoked to register all required dependencies using **`GetIt`**, a service locator. This includes Firebase, data sources, repositories, use cases, and BLoCs. By registering these dependencies, they can be easily accessed throughout the app.

3. Root Widget: MyApp

`MyApp` is the root widget of the app. It is wrapped in a **`BlocProvider`** that provides the `CarBloc` to the widget tree:

- **`BlocProvider`**: Manages the lifecycle of the `CarBloc` instance and ensures it's available to all descendants.
- **`getIt<CarBloc>()`** retrieves the `CarBloc` instance from the dependency injection container.
- **`..add(LoadCars())`**: Immediately dispatches the `LoadCars` event when the app starts, triggering the loading of car data.

The app uses **`MaterialApp`** for UI design, with a color scheme defined by `ColorScheme.fromSeed(seedColor: Colors.deepPurple)` and enables **`Material 3`** components.

4. BLoC (Business Logic Component) Integration:

The **BLoC pattern** is used to manage state and business logic. The CarBloc listens for events like LoadCars() and manages the state related to car data.

- **CarBloc:** Handles the events and updates the state of the app, such as loading or displaying cars.
- **LoadCars():** This event is triggered to fetch car data, likely from Firebase or another data source.

5. Theme and UI Configuration:

The MaterialApp widget also configures the app's theme and appearance:

- **ColorScheme.fromSeed(seedColor: Colors.deepPurple):** Defines the app's color scheme using a seed color (deep purple).
- **useMaterial3: true:** Activates Material 3 design components for the app's UI.

6. Onboarding Page:

The home screen is set to the **OnboardingPage**:

- **OnboardingPage** likely serves as the initial screen, providing information or guiding new users through the app.

Summary:

- Firebase is initialized to ensure that Firebase services are available.
- **GetIt** is used to register and inject dependencies like Firebase, data sources, repositories, and BLoCs.
- The **BLoC pattern** is implemented to manage the state of the app by handling business logic and user actions.
- The app uses **Material Design 3** for its theme, and the **OnboardingPage** is set as the starting screen for users.