Safe Zone Project Source Code

Main.dart:

```dart
import 'dart:convert';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'admin_view.dart';
import 'home.dart';
import 'onboarding.dart';

Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  // Initialize Firebase
  await Firebase.initializeApp();
  // Save the notification to SharedPreferences
  SharedPreferences prefs = await SharedPreferences.getInstance();
  List<String>? messagesJson = prefs.getStringList('notifications') ?? [];
  messagesJson.insert(0, jsonEncode(message.toMap()));
  await prefs.setStringList('notifications', messagesJson);
}

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();

  // Set up background message handler
  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);

  runApp(MaterialApp(
    home: MyApp(),
  ));
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Color(0xFFF86C1D)),
        useMaterial3: true,
      ),
      home: AuthenticationWrapper(),
    );
  }
}
```

```dart
class AuthenticationWrapper extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StreamBuilder<User?>(
      stream: FirebaseAuth.instance.authStateChanges(),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(
            child: Image.asset('assets/loading.png'), // Loading image
          );
        } else if (snapshot.hasError) {
          return Center(child: Text('Error fetching authentication state'));
        } else {
          final User? user = snapshot.data;
          if (user == null) {
            // User is signed out, navigate to the onboarding screen
            return OnboardingScreen();
          } else {
            return FutureBuilder<DocumentSnapshot>(
              future: FirebaseFirestore.instance
                  .collection('users')
                  .doc(user.uid)
                  .get(),
              builder: (context, userSnapshot) {
                if (userSnapshot.connectionState == ConnectionState.waiting) {
                  return Center(
                    child: Image.asset('assets/loading.png'), // Loading image
                  );
                } else if (!userSnapshot.hasData ||
                    !userSnapshot.data!.exists) {
                  return Center(child: Text('User document not found!'));
                } else {
                  final bool isAdmin =
                      userSnapshot.data!.get('isAdmin') ?? false;
                  if (isAdmin) {
                    return AdminView();
                  } else {
                    return HomeScreen();
                  }
                }
              },
            );
          }
        }
      },
    );
  }
}
```

```dart
class NotificationsPage extends StatefulWidget {
  @override
  _NotificationsPageState createState() => _NotificationsPageState();
}

class _NotificationsPageState extends State<NotificationsPage>
    with AutomaticKeepAliveClientMixin {
  late FirebaseMessaging _messaging;
  List<RemoteMessage> _messages = [];

  @override
  bool get wantKeepAlive => true;

  @override
  void initState() {
    super.initState();
    _messaging = FirebaseMessaging.instance;

    FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      setState(() {
        _messages.add(message);
        _saveNotifications();
      });
    });

    _loadNotifications();

    _messaging.getToken().then((token) {
      print("FCM Token: $token");
      // Send this token to your server
    });
  }

  void _loadNotifications() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    List<String>? messagesJson = prefs.getStringList('notifications');
    if (messagesJson != null) {
      setState(() {
        _messages = messagesJson
            .map((json) => RemoteMessage.fromMap(jsonDecode(json)))
            .toList();
      });
    }
  }

  void _saveNotifications() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    List<String> messagesJson =
        _messages.map((message) => jsonEncode(message.toMap())).toList();
    await prefs.setStringList('notifications', messagesJson);}}
```

```
  }
}
```

Add_car.dart:

```dart
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'home.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(AddCar());
}

class AddCar extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        scaffoldBackgroundColor: const Color(0xFFF9FAFB),
      ),
      home: VehicleDetailsScreen(),
    );
  }
}

class VehicleDetailsScreen extends StatefulWidget {
  @override
  _VehicleDetailsScreenState createState() => _VehicleDetailsScreenState();
}

class _VehicleDetailsScreenState extends State<VehicleDetailsScreen> {
  File? _carImage;
  File? _licenseImage;
  String? _selectedVehicleType;
  String? _selectedBrand;
  String? _firstName;
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController _modelController = TextEditingController();
  final TextEditingController _plateController = TextEditingController();
  bool _isUploading = false;
```

```dart
@override
void initState() {
  super.initState();
  _fetchUserName();
}

Future<void> _fetchUserName() async {
  try {
    User? user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      DocumentSnapshot userDoc = await FirebaseFirestore.instance
          .collection('users')
          .doc(user.uid)
          .get();
      setState(() {
        _firstName = userDoc['firstName'];
      });
    }
  } catch (e) {
    print("Error fetching user name: $e");
  }
}

Future<void> _getCarImage() async {
  final pickedFile =
      await ImagePicker().pickImage(source: ImageSource.gallery);
  if (pickedFile != null) {
    setState(() {
      _carImage = File(pickedFile.path);
    });
  }
}

Future<void> _getLicenseImage() async {
  final pickedFile =
      await ImagePicker().pickImage(source: ImageSource.gallery);
  if (pickedFile != null) {
    setState(() {
      _licenseImage = File(pickedFile.path);
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(),
    body: SingleChildScrollView(
      padding: const EdgeInsets.symmetric(horizontal: 16.0),
      child: Form(
```

```
        key: _formKey,
        child: Column(
          children: [
            const SizedBox(height: 20),
            Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Expanded(
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      const Text(
                        'Hello',
                        style: TextStyle(
                          color: Colors.black,
                          fontSize: 24,
                          fontFamily: 'Roboto Serif',
                          fontWeight: FontWeight.w500,
                        ),
                      ),
                      const SizedBox(height: 20),
                      Text(
                        _firstName ?? 'Loading...',
                        style: const TextStyle(
                          color: Color(0xFFF76B1C),
                          fontSize: 24,
                          fontFamily: 'Roboto Serif',
                          fontWeight: FontWeight.w500,
                        ),
                      ),
                      const SizedBox(height: 20),
                      const Text(
                        'Please add your',
                        style: TextStyle(
                          color: Colors.black,
                          fontSize: 24,
                          fontFamily: 'Roboto Serif',
                          fontWeight: FontWeight.w500,
                        ),
                      ),
                      const SizedBox(height: 20),
                      const Text(
                        'vehicle’s details',
                        style: TextStyle(
                          color: Colors.black,
                          fontSize: 24,
                          fontFamily: 'Roboto Serif',
                          fontWeight: FontWeight.w500,
                        ),
                      ),
```

```
                ],
              ),
            ),
            const SizedBox(width: 20),
            Column(
              children: [
                Stack(
                  children: [
                    CircleAvatar(
                      radius: 75,
                      backgroundColor: Colors.grey[300],
                      backgroundImage: _carImage != null
                          ? FileImage(_carImage!)
                          : null,
                      child: _carImage == null
                          ? Icon(
                              Icons.camera_alt,
                              size: 50,
                              color: Colors.grey[800],
                            )
                          : null,
                    ),
                    Positioned(
                      bottom: -13,
                      right: 4,
                      child: IconButton(
                        icon: const Icon(
                          Icons.add_a_photo_rounded,
                          color: Colors.black,
                        ),
                        onPressed: _getCarImage,
                      ),
                    ),
                  ],
                ),
                const SizedBox(height: 8),
                const Text(
                  'Car Picture',
                  style: TextStyle(
                    color: Colors.black,
                    fontSize: 16,
                    fontFamily: 'Roboto',
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
      const SizedBox(height: 40),
      _buildDropdown(
```

```
            context,
            'Type of Vehicle',
            _selectedVehicleType,
            ['Car', 'Van', 'Autobus'],
            (value) {
              setState(() {
                _selectedVehicleType = value;
              });
            },
          ),
          const SizedBox(height: 20),
          _buildDropdown(
            context,
            'Brand',
            _selectedBrand,
            [
              'Chevrolet',
              'Nissan',
              'BMW',
              'Mercedes',
              'Hyundai',
              'Kia',
              'Toyota',
              'Renault',
              'Skoda',
              'Volkswagen',
              'Peugeot',
              'Opel',
              'MG',
              'Chery',
              'Suzuki',
              'Mitsubishi',
              'Fiat',
              'Jeep',
              'Volvo',
              'Seat',
              'Audi',
              'Citroën',
              'Porsche',
              'BYD',
              'Geely',
              'Honda',
              'Tesla',
              'Subaru',
              'Lada',
              'Haval',
              'Jetour',
              'Mazda',
              'Dodge',
              'Cadillac',
```

```
                'Acura'
              ],
              (value) {
                setState(() {
                  _selectedBrand = value;
                });
              },
            ),
            const SizedBox(height: 20),
            _buildTextField('Model', _modelController),
            const SizedBox(height: 20),
            _buildTextField('Plate', _plateController),
            const SizedBox(height: 20),
            _buildImagePicker('License', _licenseImage, _getLicenseImage),
            const SizedBox(height: 40),
            Container(
              height: 48,
              width: 300,
              child: ElevatedButton(
                onPressed: _isUploading ? null : _saveCarDetails,
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color(0xFFF76B1C),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10.0),
                  ),
                ),
                child: _isUploading
                    ? CircularProgressIndicator(
                        valueColor:
                            AlwaysStoppedAnimation<Color>(Colors.white),
                      )
                    : const Text(
                        'Save',
                        textAlign: TextAlign.center,
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 16,
                          fontFamily: 'Roboto',
                          fontWeight: FontWeight.w700,
                        ),
                      ),
              ),
            ),
            const SizedBox(height: 24),
            if (_selectedVehicleType != null || _selectedBrand != null)
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Text(
                  'Selected Vehicle Type: $_selectedVehicleType\nSelected Brand:
$_selectedBrand',
```

```
                  style: const TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}

Widget _buildDropdown(BuildContext context, String hint, String? value,
    List<String> items, ValueChanged<String?> onChanged) {
  return Container(
    height: 48,
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: BorderRadius.circular(8.0),
      boxShadow: const [
        BoxShadow(
          color: Colors.black26,
          blurRadius: 4,
          offset: Offset(0, 4),
        ),
      ],
    ),
    child: DropdownButton<String>(
      dropdownColor: Colors.white,
      isExpanded: true,
      value: value,
      items: items.map((item) {
        return DropdownMenuItem<String>(
          value: item,
          child: Text(item),
        );
      }).toList(),
      onChanged: onChanged,
      hint: Text(hint),
      underline: const SizedBox(),
    ),
  );
}

Widget _buildTextField(String hint, TextEditingController controller) {
  return Container(
    height: 48,
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
```

```
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(8.0),
        boxShadow: const [
          BoxShadow(
            color: Colors.black26,
            blurRadius: 4,
            offset: Offset(0, 4),
          ),
        ],
      ),
      child: TextFormField(
        controller: controller,
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter some text';
          }
          return null;
        },
        decoration: InputDecoration(
          border: InputBorder.none,
          hintText: hint,
        ),
      ),
    );
  }

  Widget _buildImagePicker(
      String label, File? imageFile, VoidCallback onPickImage) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          label,
          style: const TextStyle(
            color: Colors.black,
            fontSize: 16,
            fontFamily: 'Roboto',
          ),
        ),
        const SizedBox(height: 8),
        GestureDetector(
          onTap: onPickImage,
          child: Container(
            height: 200,
            width: double.infinity,
            decoration: BoxDecoration(
              color: Colors.grey[200],
              borderRadius: BorderRadius.circular(10),
              image: imageFile != null
```

```
                ? DecorationImage(
                    image: FileImage(imageFile),
                    fit: BoxFit.cover,
                  )
                : null,
          ),
          child: imageFile == null
              ? Center(
                  child: Icon(
                    Icons.add_a_photo,
                    size: 50,
                    color: Colors.grey[800],
                  ),
                )
              : null,
        ),
      ),
    ],
  );
}

Future<void> _saveCarDetails() async {
  if (_formKey.currentState?.validate() ?? false) {
    setState(() {
      _isUploading = true;
    });

    try {
      String? carImageUrl;
      String? licenseImageUrl;

      if (_carImage != null) {
        carImageUrl = await _uploadFile(_carImage!, 'car_images');
      }

      if (_licenseImage != null) {
        licenseImageUrl = await _uploadFile(_licenseImage!, 'license_images');
      }

      User? user = FirebaseAuth.instance.currentUser;
      if (user != null) {
        await FirebaseFirestore.instance.collection('vehicles').add({
          'userId': user.uid,
          'vehicleType': _selectedVehicleType,
          'brand': _selectedBrand,
          'model': _modelController.text,
          'plate': _plateController.text,
          'carImageUrl': carImageUrl,
          'licenseImageUrl': licenseImageUrl,
        });
```

```
        }

        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Vehicle details saved successfully')),
        );

        // Navigate to the home screen after showing the success message
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => HomeScreen()),
        );
      } catch (e) {
        print('Error saving vehicle details: $e');
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Failed to save vehicle details')),
        );
      } finally {
        setState(() {
          _isUploading = false;
        });
      }
    }
  }

  Future<String> _uploadFile(File file, String folder) async {
    try {
      String fileName =
          '${DateTime.now().millisecondsSinceEpoch}_${file.path.split('/').last}';
      Reference storageRef =
          FirebaseStorage.instance.ref().child('$folder/$fileName');
      UploadTask uploadTask = storageRef.putFile(file);
      TaskSnapshot snapshot = await uploadTask;
      return await snapshot.ref.getDownloadURL();
    } catch (e) {
      print('Error uploading file: $e');
      return '';
    }
  }
}
```

Admin_notifications.dart:

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

class AdminNotificationPage extends StatefulWidget {
  @override
  _AdminNotificationPageState createState() => _AdminNotificationPageState();
}

class _AdminNotificationPageState extends State<AdminNotificationPage> {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final TextEditingController _titleController = TextEditingController();
  final TextEditingController _messageController = TextEditingController();

  String? _selectedUserId;
  String? _selectedUserName;

  void _sendNotification(String userId, String title, String message) async {
    await _firestore.collection('notifications').add({
      'userId': userId,
      'title': title,
      'message': message,
      'timestamp': FieldValue.serverTimestamp(),
    });

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Notification sent to $_selectedUserName')),
    );

    // Clear the text fields and selected user
    setState(() {
      _selectedUserId = null;
      _selectedUserName = null;
      _titleController.clear();
      _messageController.clear();
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Admin Notifications'),
        backgroundColor: Color(0xFFF86C1D),
      ),
      body: Center(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
```

```
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              StreamBuilder<QuerySnapshot>(
                stream: _firestore.collection('users').snapshots(),
                builder: (context, snapshot) {
                  if (!snapshot.hasData) {
                    return Center(child: CircularProgressIndicator());
                  }

                  final users = snapshot.data!.docs;

                  return DropdownButtonFormField<String>(
                    decoration: InputDecoration(
                      labelText: 'Select User',
                      border: OutlineInputBorder(),
                    ),
                    value: _selectedUserId,
                    items: users.map((user) {
                      return DropdownMenuItem<String>(
                        value: user.id,
                        child: Text(
                            '${user['firstName']} ${user['secondName']}'),
                      );
                    }).toList(),
                    onChanged: (value) {
                      setState(() {
                        _selectedUserId = value;
                        _selectedUserName = users.firstWhere(
                                (user) => user.id == value)['firstName'] +
                            ' ' +
                            users.firstWhere(
                                (user) => user.id == value)['secondName'];
                      });
                    },
                  );
                },
              ),
              SizedBox(height: 16),
              TextField(
                controller: _titleController,
                decoration: InputDecoration(
                  labelText: 'Title',
                  border: OutlineInputBorder(),
                ),
              ),
              SizedBox(height: 16),
              TextField(
                controller: _messageController,
```

```
                decoration: InputDecoration(
                  labelText: 'Message',
                  border: OutlineInputBorder(),
                ),
                maxLines: 3,
              ),
              SizedBox(height: 16),
              Center(
                child: ElevatedButton(
                  onPressed: () {
                    if (_selectedUserId != null &&
                        _titleController.text.isNotEmpty &&
                        _messageController.text.isNotEmpty) {
                      _sendNotification(
                        _selectedUserId!,
                        _titleController.text,
                        _messageController.text,
                      );
                    } else {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(content: Text('Please fill all fields')),
                      );
                    }
                  },
                  child: Text('Send Notification'),
                  style: ElevatedButton.styleFrom(
                    foregroundColor: Colors.white,
                    backgroundColor: Color(0xFFF86C1D),
                    padding: EdgeInsets.symmetric(
                        vertical: 16.0, horizontal: 30.0),
                    textStyle: TextStyle(fontSize: 16),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

Admin_received_notifications.dart:

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

class AdminReceivedNotifications extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'Admin Notifications',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: Color(0xFFF76B1C), // Light orange color
      ),
      body: StreamBuilder<QuerySnapshot>(
        stream: FirebaseFirestore.instance
            .collection('adminNotifications')
            .orderBy('timestamp', descending: true)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }

          if (!snapshot.hasData) {
            return Center(child: CircularProgressIndicator());
          }

          final notifications = snapshot.data!.docs;

          if (notifications.isEmpty) {
            return Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Icon(
                    Icons.notifications_none,
                    size: 100,
                    color: Colors.grey,
                  ),
                  SizedBox(height: 20),
                  Text('No Notifications Found.'),
                ],
              ),
            );
          }

          return ListView.builder(
```

```
          itemCount: notifications.length,
          itemBuilder: (context, index) {
            var notification = notifications[index];
            return FutureBuilder<String>(
              future: _getFullName(notification['userId']),
              builder: (context, userSnapshot) {
                if (!userSnapshot.hasData) {
                  return _buildLoadingCard();
                }
                return _buildNotificationCard(
                    context, notification, userSnapshot.data!);
              },
            );
          },
        );
      },
    ),
  );
}

Future<String> _getFullName(String userId) async {
  var userSnapshot =
      await FirebaseFirestore.instance.collection('users').doc(userId).get();
  var userData = userSnapshot.data();
  return '${userData?['firstName']} ${userData?['secondName']}';
}

Widget _buildLoadingCard() {
  return Card(
    elevation: 2,
    margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(15),
    ),
    child: ListTile(
      title: Text(
        'New Notification',
        style: TextStyle(fontWeight: FontWeight.bold),
      ),
      subtitle: Text('Loading...'),
    ),
  );
}

Widget _buildNotificationCard(
    BuildContext context, DocumentSnapshot notification, String fullName) {
  return Card(
    elevation: 2,
    margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
    shape: RoundedRectangleBorder(
```

```
        borderRadius: BorderRadius.circular(15),
      ),
      child: ListTile(
        title: Text(
          'New Notification',
          style: TextStyle(fontWeight: FontWeight.bold),
        ),
        subtitle: Text('User: $fullName'),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) =>
                  NotificationDetail(notification: notification),
            ),
          );
        },
      ),
    );
  }
}

class NotificationDetail extends StatelessWidget {
  final DocumentSnapshot notification;

  const NotificationDetail({required this.notification});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'Notification Details',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: Color(0xFFF76B1C), // Light orange color
      ),
      body: FutureBuilder<Map<String, dynamic>>(
        future: _getDetails(notification),
        builder: (context, detailSnapshot) {
          if (detailSnapshot.hasError) {
            return Center(child: Text('Error: ${detailSnapshot.error}'));
          }

          if (!detailSnapshot.hasData) {
            return Center(child: CircularProgressIndicator());
          }

          var details = detailSnapshot.data!;
          return Padding(
```

```
            padding: EdgeInsets.all(20.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  'User: ${details['fullName']}',
                  style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
                ),
                SizedBox(height: 10),
                Text(
                  '${details['idFieldName']}: ${details['serviceName'] ??
details['vehicleBrand']}',
                  style: TextStyle(fontSize: 16),
                ),
                if (details.containsKey('appointmentDateTime')) ...[
                  SizedBox(height: 10),
                  Text(
                    'Appointment Date & Time: ${details['appointmentDateTime']}',
                    style: TextStyle(fontSize: 16),
                  ),
                ],
                SizedBox(height: 10),
                Text(
                  'Timestamp: ${notification['timestamp'].toDate()}',
                  style: TextStyle(fontSize: 16),
                ),
              ],
            ),
          );
        },
      ),
    );
  }

  Future<Map<String, dynamic>> _getDetails(
      DocumentSnapshot notification) async {
    var userSnapshot = await FirebaseFirestore.instance
        .collection('users')
        .doc(notification['userId'])
        .get();
    var userData = userSnapshot.data();

    String? idFieldName;
    String? idValue;

    // Check if the notification contains 'packageId' or 'serviceId' field
    if ((notification.data() as Map<String, dynamic>)
        .containsKey('packageId')) {
      idFieldName = 'Package Name';
      idValue = notification['packageId'];
```

```
    } else if ((notification.data() as Map<String, dynamic>)
        .containsKey('serviceId')) {
      idFieldName = 'Service Name';
      idValue = notification['serviceId'];
    }

    var documentSnapshot = await FirebaseFirestore.instance
        .collection(idFieldName == 'package_Name' ? 'packages' : 'services')
        .doc(idValue)
        .get();

    var documentData = documentSnapshot.data();

    return {
      'fullName': '${userData?['firstName']} ${userData?['secondName']}',
      'idFieldName': capitalize(idFieldName ?? ''), // Capitalize the field name
      'serviceName': documentData?['service_name'],
      'vehicleBrand': documentData?['brand'],
      if (documentData?.containsKey('appointmentDateTime') ?? false)
        'appointmentDateTime': documentData?['appointmentDateTime'],
    };
  }

  String capitalize(String input) {
    if (input.isEmpty) {
      return input;
    }
    return input[0].toUpperCase() + input.substring(1);
  }
}
```

Admin_view.dart:

```dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:safe_zone/admin_received_notifications.dart';
import 'users.dart';
import 'cars.dart';
import 'profile.dart';
import 'admin_notifications.dart';

class AdminView extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Admin View'),
        actions: [
          IconButton(
            icon: Icon(Icons.logout),
            onPressed: () {
              FirebaseAuth.instance.signOut();
            },
          ),
        ],
      ),
      body: Center(
        child: SingleChildScrollView(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'SAFE ZONE',
                style: TextStyle(
                  color: Color(0xFFF86C1D),
                  fontFamily: 'Poppins',
                  fontSize: 24.0,
                  fontWeight: FontWeight.w600,
                  letterSpacing: -0.408,
                ),
                textAlign: TextAlign.center,
              ),
              SizedBox(height: 17.0),
              Text(
                "Admin View",
                style: TextStyle(
                  color: Color(0xFF040415),
                  fontFamily: 'Poppins',
                  fontSize: 20.0,
                  fontWeight: FontWeight.w500,
                ),
```

```
              textAlign: TextAlign.center,
            ),
            SizedBox(height: 20.0), // Adjusted spacing
            _buildFrame(context, 'Users', Users()),
            SizedBox(height: 30.0), // Adjusted spacing
            _buildFrame(context, 'Cars', Cars()),
            SizedBox(height: 30.0), // Adjusted spacing
            _buildFrame(
                context, 'Send Notifications', AdminNotificationPage()),
            SizedBox(height: 30.0), // Adjusted spacing
            _buildFrame(context, 'Profile', ProfilePage()),
            SizedBox(height: 30.0), // Adjusted spacing
            _buildFrame(context, 'Notifications',
                AdminReceivedNotifications()), // New frame for Notifications
          ],
        ),
      ),
    ),
  );
}

Widget _buildFrame(BuildContext context, String title, Widget page) {
  return GestureDetector(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => page),
      );
    },
    child: Container(
      width: 330.0,
      height: 140.0,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(30.0),
        color: Color(0xFFFF5C00),
        border: Border.all(
          color: Colors.black,
          width: 2.0,
        ),
      ),
      child: Stack(
        children: [
          Positioned(
            left: 13.0,
            top: 10.0,
            child: Container(
              width: 20.0,
              height: 30.0,
              decoration: BoxDecoration(
                color: Colors.transparent,
```

```
              ),
              child: Icon(
                Icons.arrow_forward,
                size: 26.0,
                color: Color(0xFF1E232C),
              ),
            ),
          ),
          Center(
            child: Text(
              title,
              style: TextStyle(
                color: Colors.white,
                fontFamily: 'Poppins',
                fontSize: 32.0,
                fontWeight: FontWeight.w700,
              ),
              textAlign: TextAlign.center,
            ),
          ),
        ],
      ),
    ),
  );
 }
}

void main() {
 runApp(MaterialApp(
   home: AdminView(),
 ));
}
```

Cars.dart:

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

// Define a Vehicle class to represent the structure of a vehicle
class Vehicle {
  final String name;
  final String model;
  final String userId; // Add userId to the Vehicle class

  Vehicle(this.name, this.model, this.userId);
}

void main() {
  runApp(const Cars());
}

class Cars extends StatelessWidget {
  const Cars({Key? key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: 'Users',
      home: UserPage(),
    );
  }
}

class UserPage extends StatelessWidget {
  const UserPage({Key? key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Align(
          alignment: Alignment.topCenter,
          child: Text(
            'Cars',
            textAlign: TextAlign.center,
            style: TextStyle(
              color: Color(0xFFF76B1C),
              fontSize: 17,
              fontFamily: 'Roboto',
              fontWeight: FontWeight.w700,
              height: 0,
              letterSpacing: -0.41,
```

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

        List<Vehicle> vehicles =
            snapshot.data!.docs.map((DocumentSnapshot document) {
          Map<String, dynamic> data = document.data() as Map<String, dynamic>;
          return Vehicle(data['brand'], data['model'], data['userId']);
        }).toList();

        return SingleChildScrollView(
          child: Align(
            alignment: Alignment.topCenter,
            child: Padding(
              padding:
                  const EdgeInsets.only(top: 10.0, left: 23.5, right: 23.5),
              child: Column(
                children: List.generate(
                  vehicles.length,
                  (index) => VehicleBox(
                    vehicleIndex: index + 1,
                    vehicle: vehicles[index],
                  ),
                ),
              ),
            ),
          ),
        );
      },
    ),
  );
 }
}

class VehicleBox extends StatelessWidget {
  final int vehicleIndex;
  final Vehicle vehicle;

  const VehicleBox(
      {Key? key, required this.vehicleIndex, required this.vehicle});
```

```dart
@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.only(top: 20.0),
    child: SizedBox(
      width: 435,
      height: 145,
      child: Container(
        clipBehavior: Clip.antiAlias,
        decoration: BoxDecoration(
          color: Colors.white,
          shape: BoxShape.rectangle,
          borderRadius: BorderRadius.circular(32),
          border: Border.all(width: 2, color: const Color(0xFFF76B1C)),
          boxShadow: const [
            BoxShadow(
              color: Color(0x0F000000),
              blurRadius: 12,
              offset: Offset(0, 2),
            ),
          ],
        ),
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                children: [
                  const Icon(Icons.directions_car_filled_sharp,
                      color: Color(0xFFF76B1C)),
                  const SizedBox(width: 8),
                  Text(
                    '${vehicle.name} ${vehicle.model}',
                    style: const TextStyle(
                      color: Colors.black,
                      fontSize: 15,
                      fontFamily: 'Roboto',
                      fontWeight: FontWeight.w900,
                    ),
                  ),
                ],
              ),
              const SizedBox(height: 6),
              const Divider(
                height: 2,
                thickness: 2,
                color: Color(0xFFF76B1C),
              ),
              const SizedBox(
```

```
                height: 4,
              ),
              const SizedBox(height: 8),
              Padding(
                padding: const EdgeInsets.only(left: 28),
                child: RichText(
                  text: TextSpan(
                    children: [
                      TextSpan(
                        text: 'Car $vehicleIndex:\n',
                        style: const TextStyle(
                          color: Colors.black,
                          fontSize: 20,
                          fontFamily: 'Roboto',
                          fontWeight: FontWeight.w700,
                          decoration: TextDecoration.underline,
                          wordSpacing: 2,
                        ),
                      ),
                      const TextSpan(
                        text: '\n',
                        style: TextStyle(
                          fontSize: 10,
                        ),
                      ),
                      TextSpan(
                        text: 'User Details',
                        style: const TextStyle(
                          color: Color(0xFFF76B1C),
                          fontSize: 20,
                          fontFamily: 'Roboto',
                          fontWeight: FontWeight.w700,
                          decoration: TextDecoration.underline,
                        ),
                        recognizer: TapGestureRecognizer()
                          ..onTap = () {
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                builder: (context) => UserDetailsPage(
                                  userId: vehicle.userId,
                                ),
                              ),
                            );
                          },
                      ),
                      const TextSpan(
                        text: '   ',
                        style: TextStyle(
                          color: Colors.black,
```

```
                          fontSize: 20,
                          fontFamily: 'Roboto',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                      const TextSpan(
                        text: '          ',
                      ),
                    ],
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
  }
}

class UserDetailsPage extends StatelessWidget {
  final String userId;

  const UserDetailsPage({Key? key, required this.userId}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'User Details',
          style: TextStyle(
            color: Color(0xFFF76B1C),
            fontSize: 17,
            fontFamily: 'Roboto',
            fontWeight: FontWeight.w700,
            letterSpacing: -0.41,
          ),
        ),
      ),
      body: FutureBuilder<DocumentSnapshot>(
        future:
            FirebaseFirestore.instance.collection('users').doc(userId).get(),
        builder:
            (BuildContext context, AsyncSnapshot<DocumentSnapshot> snapshot) {
          if (!snapshot.hasData) {
            return const Center(
              child: CircularProgressIndicator(),
            );
```

```
          }

          if (!snapshot.data!.exists) {
            return const Center(
              child: Text('User not found'),
            );
          }

          Map<String, dynamic> data =
              snapshot.data!.data() as Map<String, dynamic>;

          return SingleChildScrollView(
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                children: [
                  data['profileImgUrl'] != null
                      ? Image.network(data['profileImgUrl'])
                      : const Icon(Icons.account_circle, size: 100),
                  const SizedBox(height: 16),
                  Text('First Name: ${data['firstName']}'),
                  Text('Second Name: ${data['secondName']}'),
                  Text('Gender: ${data['gender']}'),
                  Text('Birthdate: ${data['birthdate']}'),
                  Text('Phone Number: ${data['phoneNumber']}'),
                  Text('Email: ${data['email']}'),
                ],
              ),
            ),
          );
        },
      ),
    );
  }
}
```

Firebase_options.dart:

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        return windows;
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyCpLHHWzjsgJkeDUFmCkW2dDX14-BORSLQ',
    appId: '1:824440030133:web:b1692d61423ce527e44ffc',
    messagingSenderId: '824440030133',
    projectId: 'safezone-9f5a9',
    authDomain: 'safezone-9f5a9.firebaseapp.com',
```

```
    storageBucket: 'safezone-9f5a9.appspot.com',
    measurementId: 'G-05X4SHLSEW',
  );

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyAIsj4kDe1GQTFSK3rX4Xx-fswqiBEAEe4',
    appId: '1:824440030133:android:5f81be414f4380bae44ffc',
    messagingSenderId: '824440030133',
    projectId: 'safezone-9f5a9',
    storageBucket: 'safezone-9f5a9.appspot.com',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyAM1x5_0yurqiqrufeVVAWO1aeTKfEnsNw',
    appId: '1:824440030133:ios:64b500871079143ce44ffc',
    messagingSenderId: '824440030133',
    projectId: 'safezone-9f5a9',
    storageBucket: 'safezone-9f5a9.appspot.com',
    iosBundleId: 'com.example.safeZone',
  );

  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyAM1x5_0yurqiqrufeVVAWO1aeTKfEnsNw',
    appId: '1:824440030133:ios:64b500871079143ce44ffc',
    messagingSenderId: '824440030133',
    projectId: 'safezone-9f5a9',
    storageBucket: 'safezone-9f5a9.appspot.com',
    iosBundleId: 'com.example.safeZone',
  );

  static const FirebaseOptions windows = FirebaseOptions(
    apiKey: 'AIzaSyCpLHHWzjsgJkeDUFmCkW2dDX14-BORSLQ',
    appId: '1:824440030133:web:50a3eebf48bf2c10e44ffc',
    messagingSenderId: '824440030133',
    projectId: 'safezone-9f5a9',
    authDomain: 'safezone-9f5a9.firebaseapp.com',
    storageBucket: 'safezone-9f5a9.appspot.com',
    measurementId: 'G-E31BH9741F',
  );
}
```

Home.dart:

```dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'my_cars.dart';
import 'profile.dart';
import 'packages.dart';
import 'add_car.dart';
import 'services.dart';
import 'notifications.dart';
import 'my_reservations.dart'; // Import MyReservations screen
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:url_launcher/url_launcher.dart'; // Import url_launcher package

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      // Define routes
      routes: {
        '/my_cars': (context) => MyCars(),
        // Add other routes as needed
      },
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatelessWidget {
  const HomeScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'SAFE ZONE',
          style: TextStyle(
            color: Color(0xFFF76B1C),
            fontSize: 24,
            fontFamily: 'Roboto Serif',
            fontWeight: FontWeight.w600,
            height: 0.5,
            letterSpacing: -0.41,
          ),
        ),
```

```
        centerTitle: true,
        actions: [
          IconButton(
            icon: const Icon(Icons.notifications),
            onPressed: () async {
              final user = FirebaseAuth.instance.currentUser;
              if (user != null) {
                final userId = user.uid;
                Navigator.push(
                  context,
                  PageRouteBuilder(
                    transitionDuration: Duration(milliseconds: 500),
                    pageBuilder: (context, animation, secondaryAnimation) =>
                        NotificationsPage(
                      userId: userId,
                    ),
                    transitionsBuilder:
                        (context, animation, secondaryAnimation, child) {
                      var begin = Offset(1.0, 0.0);
                      var end = Offset.zero;
                      var curve = Curves.easeInOutQuart;
                      var tween = Tween(begin: begin, end: end)
                          .chain(CurveTween(curve: curve));
                      var offsetAnimation = animation.drive(tween);
                      return SlideTransition(
                        position: offsetAnimation,
                        child: child,
                      );
                    },
                  ),
                );
              }
            },
          ),
        ],
      ),
      drawer: Drawer(
        child: StreamBuilder<DocumentSnapshot>(
          stream: _userInfoStream(),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return Center(child: CircularProgressIndicator());
            } else if (snapshot.hasError) {
              return Center(child: Text('Error: ${snapshot.error}'));
            } else if (snapshot.hasData && snapshot.data != null) {
              final userData = snapshot.data!;
              final firstName = userData['firstName'] ?? '';
              final secondName = userData['secondName'] ?? '';
              final email = userData['email'] ?? '';
              final profileImgUrl = userData['profileImgUrl'] ?? '';
```

```
        return ListView(
          children: [
            UserAccountsDrawerHeader(
              accountName: Text('$firstName $secondName'),
              accountEmail: Text(email),
              currentAccountPicture: CircleAvatar(
                radius: 50,
                backgroundColor: Colors.grey,
                backgroundImage: profileImgUrl.isNotEmpty
                    ? NetworkImage(profileImgUrl)
                    : null,
              ),
              decoration: BoxDecoration(
                color: Color(0xFFF76B1C),
              ),
            ),
            // Add divider
            ListTile(
              title: const Text(
                'Profile',
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 18,
                  fontFamily: 'Roboto Serif',
                  fontWeight: FontWeight.w600,
                ),
              ),
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => Profile()),
                );
              },
            ),
            const Divider(), // Add divider
            ListTile(
              title: const Text(
                'Home',
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 18,
                  fontFamily: 'Roboto Serif',
                  fontWeight: FontWeight.w600,
                ),
              ),
              onTap: () {
                Navigator.popUntil(context, ModalRoute.withName('/'));
              },
            ),
```

```
                const Divider(), // Add divider
                ListTile(
                  title: const Text(
                    'My Cars', // Add My Cars option
                    style: TextStyle(
                      color: Colors.black,
                      fontSize: 18,
                      fontFamily: 'Roboto Serif',
                      fontWeight: FontWeight.w600,
                    ),
                  ),
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => MyCars(),
                      ),
                    );
                  },
                ),
                const Divider(), // Add divider
                ListTile(
                  title: const Text(
                    'Live Camera', // Add Live Camera option
                    style: TextStyle(
                      color: Colors.black,
                      fontSize: 18,
                      fontFamily: 'Roboto Serif',
                      fontWeight: FontWeight.w600,
                    ),
                  ),
                  onTap: () async {
                    const url = "https://www.hik-connect.com/register";
                    // ignore: deprecated_member_use
                    if (await canLaunch(url)) {
                      // ignore: deprecated_member_use
                      await launch(url);
                    } else {
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: Text('Error'),
                            content: Text(
                                'Failed to launch the URL. Please try again later.'),
                            actions: [
                              TextButton(
                                onPressed: () {
                                  Navigator.of(context).pop();
                                },
```

```
                    child: Text('OK'),
                  ),
                ],
              );
            },
          );
        }
      },
    ),
    const Divider(), // Add divider
    ListTile(
      title: const Text(
        'Add Vehicle',
        style: TextStyle(
          color: Colors.black,
          fontSize: 18,
          fontFamily: 'Roboto Serif',
          fontWeight: FontWeight.w600,
        ),
      ),
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => AddCar()),
        );
      },
    ),
    const Divider(), // Add divider
    ListTile(
      title: const Text(
        'Services',
        style: TextStyle(
          color: Colors.black,
          fontSize: 18,
          fontFamily: 'Roboto Serif',
          fontWeight: FontWeight.w600,
        ),
      ),
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => Services()),
        );
      },
    ),
    const Divider(), // Add divider
    ListTile(
      title: const Text(
        'Packages',
        style: TextStyle(
```

```
            color: Colors.black,
            fontSize: 18,
            fontFamily: 'Roboto Serif',
            fontWeight: FontWeight.w600,
          ),
        ),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => Packages()),
          );
        },
      ),
      const Divider(), // Add divider
      ListTile(
        title: const Text(
          'My Reservations', // Add My Reservations option
          style: TextStyle(
            color: Colors.black,
            fontSize: 18,
            fontFamily: 'Roboto Serif',
            fontWeight: FontWeight.w600,
          ),
        ),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) => MyReservations()),
          );
        },
      ),
      const Divider(), // Add divider
      ListTile(
        title: const Text(
          'Contact Us', // Add Contact Us option
          style: TextStyle(
            color: Colors.black,
            fontSize: 18,
            fontFamily: 'Roboto Serif',
            fontWeight: FontWeight.w600,
          ),
        ),
        onTap: () async {
          const phoneNumber = '01018843770';
          final Uri launchUri = Uri(
            scheme: 'tel',
            path: phoneNumber,
          );
          if (await canLaunch(launchUri.toString())) {
```

```
                await launch(launchUri.toString());
              } else {
                throw 'Could not launch $launchUri';
              }
            },
          ),
          const Divider(), // Add divider
          ListTile(
            title: const Text(
              'Logout',
              style: TextStyle(
                color: Colors.black,
                fontSize: 18,
                fontFamily: 'Roboto Serif',
                fontWeight: FontWeight.w600,
              ),
            ),
            onTap: () async {
              bool? confirmLogout = await showDialog<bool>(
                context: context,
                builder: (BuildContext context) {
                  return AlertDialog(
                    title: Text('Confirm Logout'),
                    content: Text('Are you sure you want to log out?'),
                    actions: [
                      TextButton(
                        onPressed: () {
                          Navigator.of(context).pop(false);
                        },
                        child: Text('Cancel'),
                      ),
                      TextButton(
                        onPressed: () {
                          Navigator.of(context).pop(true);
                        },
                        child: Text('Logout'),
                      ),
                    ],
                  );
                },
              );

              if (confirmLogout == true) {
                await FirebaseAuth.instance.signOut();
                Navigator.pop(context);
              }
            },
          ),
          const Divider(),
        ],
```

```
            );
          } else {
            return SizedBox.shrink();
          }
        },
      ),
    ),
    body: Stack(
      children: [
        // Background Image
        Positioned.fill(
          child: Image.asset(
            'assets/background.jpg',
            fit: BoxFit.cover,
            color: Colors.white.withOpacity(0.8),
            colorBlendMode: BlendMode.hardLight,
          ),
        ),
        Center(
          child: SingleChildScrollView(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                GestureDetector(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => Packages()),
                    );
                  },
                  child: Container(
                    margin: EdgeInsets.symmetric(horizontal: 20),
                    decoration: BoxDecoration(
                      color: const Color(0xFFF76B1C),
                      borderRadius: BorderRadius.circular(30),
                    ),
                    width: 300,
                    height: 220,
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: [
                        ClipRRect(
                          borderRadius: BorderRadius.circular(30),
                          child: Image.asset(
                            'assets/pack.jpg',
                            width: 300,
                            height: 189,
                            fit: BoxFit.cover,
                          ),
                        ),
```

```
                const SizedBox(height: 0),
                const Text(
                  'Packages',
                  style: TextStyle(
                    fontFamily: "Roboto Serif",
                    color: Colors.white,
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ],
            ),
          ),
        ),
        const SizedBox(height: 30),
        GestureDetector(
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => Services()),
            );
          },
          child: Container(
            margin: EdgeInsets.symmetric(horizontal: 20),
            decoration: BoxDecoration(
              color: const Color(0xFFF76B1C),
              borderRadius: BorderRadius.circular(30),
            ),
            width: 300,
            height: 220,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                ClipRRect(
                  borderRadius: BorderRadius.circular(30),
                  child: Image.asset(
                    'assets/serv.jpg',
                    width: 300,
                    height: 189,
                    fit: BoxFit.cover,
                  ),
                ),
                const SizedBox(height: 0),
                const Text(
                  'Services',
                  style: TextStyle(
                    fontFamily: "Roboto Serif",
                    color: Colors.white,
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
```

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

 Stream<DocumentSnapshot> _userInfoStream() {
   final user = FirebaseAuth.instance.currentUser;
   if (user != null) {
     return FirebaseFirestore.instance
         .collection('users')
         .doc(user.uid)
         .snapshots();
   }
   throw Exception('User not found');
 }
}
```

Login.dart:

```dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:safe_zone/admin_view.dart';
import 'package:safe_zone/google.dart';
import 'package:safe_zone/home.dart';
import 'package:safe_zone/signup.dart';

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Center(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              SizedBox(height: 92),
              Container(
                width: 76,
                height: 76,
                decoration: BoxDecoration(
                  shape: BoxShape.circle,
                  color: Color(0xFFFF5000),
                ),
              ),
              SizedBox(height: 16),
              Text(
                'Login',
                style: TextStyle(
                  fontFamily: 'Roboto',
                  fontSize: 20,
                  letterSpacing: 0.0,
```

```
            color: Color(0xFFF86C1D),
            fontWeight: FontWeight.w600,
          ),
        ),
        SizedBox(height: 8),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Welcome to ',
              style: TextStyle(
                fontFamily: 'Roboto',
                fontSize: 14,
                color: Color(0xFF040415),
                fontWeight: FontWeight.normal,
              ),
            ),
            Text(
              'Safe Zone',
              style: TextStyle(
                fontFamily: 'Roboto',
                fontSize: 14,
                color: Color(0xFF040415),
                fontWeight: FontWeight.bold,
              ),
            ),
          ],
        ),
        SizedBox(height: 16),
        Center(
          child: Column(
            children: [
              SizedBox(height: 8),
              Container(
                width: 350,
                height: 64,
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(17),
                  color: Color.fromRGBO(255, 255, 255, 0.894),
                  boxShadow: [
                    BoxShadow(
                      color: Colors.black.withOpacity(0.25),
                      offset: Offset(0, 4),
                      blurRadius: 4,
                    ),
                  ],
                ),
                child: Row(
                  children: [
                    SizedBox(width: 8),
```

```
                Icon(Icons.account_circle,
                    size: 32, color: Colors.black),
                SizedBox(width: 8),
                Expanded(
                  child: TextField(
                    controller: _emailController,
                    textAlign: TextAlign.center,
                    style: TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.w500,
                      fontFamily: 'Roboto',
                      color: Colors.black,
                    ),
                    decoration: InputDecoration(
                      border: InputBorder.none,
                      hintText: 'Email',
                      hintStyle: TextStyle(
                        color: Color(0xFFA8AFB9),
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ),
          SizedBox(height: 24),
          Container(
            width: 350,
            height: 64,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(17),
              color: Color.fromRGBO(255, 255, 255, 0.937),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withOpacity(0.25),
                  offset: Offset(0, 4),
                  blurRadius: 4,
                ),
              ],
            ),
            child: Row(
              children: [
                SizedBox(width: 8),
                Icon(Icons.lock, size: 32, color: Colors.black),
                SizedBox(width: 8),
                Expanded(
                  child: TextField(
                    controller: _passwordController,
                    textAlign: TextAlign.center,
                    obscureText: true,
```

```
                    style: TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.w500,
                      fontFamily: 'Roboto',
                      color: Colors.black,
                    ),
                    decoration: InputDecoration(
                      border: InputBorder.none,
                      hintText: 'Password',
                      hintStyle: TextStyle(
                        color: Color(0xFFA8AFB9),
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
    SizedBox(height: 16),
    InkWell(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => ForgotPasswordPage()),
        );
      },
      child: Text(
        'Forgot password?',
        style: TextStyle(
          fontFamily: 'Roboto',
          fontSize: 14,
          color: Color(0xFFFE6610),
          fontWeight: FontWeight.normal,
        ),
      ),
    ),
    SizedBox(height: 24),
    Container(
      width: 360,
      height: 64,
      decoration: BoxDecoration(
        color: Color(0xFFF86C1D),
        boxShadow: [
          BoxShadow(
            color: Colors.black.withOpacity(0.25),
            offset: Offset(0, 4),
```

```
                blurRadius: 4,
              ),
            ],
          ),
          child: InkWell(
            onTap: () async {
              String email = _emailController.text;
              String password = _passwordController.text;
              try {
                final credential = await FirebaseAuth.instance
                    .signInWithEmailAndPassword(
                        email: email, password: password);
                if (credential != null) {
                  final user = FirebaseAuth.instance.currentUser;
                  final userDoc = await FirebaseFirestore.instance
                      .collection('users')
                      .doc(user?.uid)
                      .get();

                  if (userDoc.exists) {
                    final bool isAdmin = userDoc.get('isAdmin') ?? false;
                    if (isAdmin) {
                      Navigator.pushReplacement(
                          context,
                          MaterialPageRoute(
                              builder: (context) => AdminView()));
                    } else {
                      Navigator.pushReplacement(
                          context,
                          MaterialPageRoute(
                              builder: (context) => HomeScreen()));
                    }
                  } else {
                    ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(
                        content: Text('User document not found!'),
                        backgroundColor: Colors.red,
                      ),
                    );
                  }
                }
              } on FirebaseAuthException catch (e) {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(
                    content: Text(e.message ?? 'An error occurred'),
                    backgroundColor: Colors.red,
                  ),
                );
              }
            },
```

```
            child: Center(
              child: Text(
                'Login',
                style: TextStyle(
                  fontFamily: 'Roboto',
                  fontSize: 18,
                  color: Color(0xFFFBFBFB),
                  fontWeight: FontWeight.w600,
                ),
              ),
            ),
          ),
        ),
        SizedBox(height: 20),
        Text(
          '———————— Or ————————',
          style: TextStyle(
            fontFamily: 'Roboto',
            fontSize: 20,
            color: Color(0xFFC4C4C4),
            fontWeight: FontWeight.normal,
          ),
        ),
        Container(
          margin: EdgeInsets.symmetric(vertical: 16.0),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Icon(Icons.facebook, size: 45, color: Color(0xFF4267B2)),
              SizedBox(
                width: 20,
              ),
              GoogleLogo()
            ],
          ),
        ),
        SizedBox(height: 40),
        InkWell(
          onTap: () {
            Navigator.pushReplacement(
              context,
              MaterialPageRoute(builder: (context) => SignUpScreen()),
            );
          },
          child: Text('Don\'t have an account? Sign up!',
              style: TextStyle(
                color: Color(0xFFC4C4C4),
                fontSize: 14,
                fontWeight: FontWeight.bold,
                decoration: TextDecoration.underline,
```

```
                    decorationColor: Colors.grey,
                  )),
            ),
          ],
        ),
      ),
    ),
  );
 }
}

class ForgotPasswordPage extends StatelessWidget {
 final TextEditingController _emailController = TextEditingController();

 Future<void> _resetPassword(BuildContext context) async {
  try {
   await FirebaseAuth.instance.sendPasswordResetEmail(
    email: _emailController.text,
   );
   ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
     content: Text("Password reset email sent"),
     backgroundColor: Colors.green,
    ),
   );
  } catch (e) {
   ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
     content: Text("Failed to send password reset email"),
     backgroundColor: Colors.red,
    ),
   );
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   body: SingleChildScrollView(
    child: Column(
     crossAxisAlignment: CrossAxisAlignment.stretch,
     children: [
      Container(
       padding: EdgeInsets.symmetric(vertical: 10, horizontal: 24),
       color: Color(0xFFF86C1D),
       child: Text(
        '\nForgot Password?',
        style: TextStyle(
         fontSize: 24,
         fontWeight: FontWeight.bold,
```

```
                color: Colors.white,
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(24.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                TextField(
                  controller: _emailController,
                  decoration: InputDecoration(
                    labelText: 'Email',
                    border: OutlineInputBorder(),
                  ),
                ),
                SizedBox(height: 24),
                ElevatedButton(
                  onPressed: () => _resetPassword(context),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Color(0xFFF86C1D),
                    textStyle: TextStyle(fontSize: 20),
                  ),
                  child: Padding(
                    padding: const EdgeInsets.all(16.0),
                    child: Text('Reset Password',
                        style: TextStyle(color: Colors.white)),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

My_cars.dart:

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

class MyCars extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('My Cars'),
        backgroundColor: Color(0xFFF76B1C),
      ),
      body: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Padding(
            padding: EdgeInsets.all(16),
            child: Text(
              'My Cars',
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
          Expanded(
            child: FutureBuilder<QuerySnapshot>(
              future: _fetchUserVehicleInfo(),
              builder: (context, snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return Center(
                    child: CircularProgressIndicator(),
                  );
                } else if (snapshot.hasError) {
                  return Center(
                    child: Text(
                      'Error: ${snapshot.error}',
                      style: TextStyle(color: Colors.red),
                    ),
                  );
                } else if (snapshot.hasData && snapshot.data != null) {
                  final vehicleData = snapshot.data!;
                  final vehicleWidgets = <Widget>[];

                  for (var i = 0; i < vehicleData.docs.length; i++) {
                    final vehicle = vehicleData.docs[i];
                    final brand = vehicle['brand'] ?? '';
                    final model = vehicle['model'] ?? '';
```

```dart
                final plate = vehicle['plate'] ?? '';
                final carImageUrl = vehicle['carImageUrl'] ?? '';

                // Calculate car number
                final carNumber = i + 1;

                vehicleWidgets.add(
                  Container(
                    padding: EdgeInsets.all(16),
                    margin: EdgeInsets.symmetric(vertical: 8),
                    decoration: BoxDecoration(
                      color: Color(0xFFFFE5B4),
                      borderRadius: BorderRadius.circular(10),
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        _buildInfoRow('Car $carNumber Brand:', brand),
                        SizedBox(height: 8),
                        _buildInfoRow('Car $carNumber Model:', model),
                        SizedBox(height: 8),
                        _buildInfoRow('Plate:', plate),
                        Padding(
                          padding: EdgeInsets.only(top: 8),
                          child: ClipRRect(
                            borderRadius: BorderRadius.circular(10),
                            child: Image.network(
                              carImageUrl,
                              width: double.infinity,
                              height: 200,
                              fit: BoxFit.cover,
                            ),
                          ),
                        ),
                        SizedBox(height: 8),
                        Row(
                          mainAxisAlignment: MainAxisAlignment.spaceBetween,
                          children: [
                            Text(
                              'Subscribed to:',
                              style: TextStyle(
                                fontSize: 14,
                                fontWeight: FontWeight.w500,
                                color: Colors.black,
                              ),
                            ),
                            Text(
                              'Gold Package',
                              style: TextStyle(
                                color: Color(0xFFF76B1C),
```

```
                                fontSize: 14,
                                fontWeight: FontWeight.w500,
                              ),
                            ),
                          ],
                        ),
                      ],
                    ),
                  ),
                );
              }

              return ListView(
                shrinkWrap: true,
                children: vehicleWidgets,
              );
            } else {
              return SizedBox.shrink();
            }
          },
        ),
      ),
    ],
  ),
);
}

Widget _buildInfoRow(String label, String value) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Text(
            label,
            style: TextStyle(
              fontSize: 14,
              fontWeight: FontWeight.w500,
              color: Colors.black,
            ),
          ),
          Text(
            value,
            style: TextStyle(
              fontSize: 14,
              fontWeight: FontWeight.w500,
              color: Colors.black,
            ),
          ),
```

```
          ],
        ),
        SizedBox(height: 4), // Added spacing between fields
        Container(
          height: 1,
          color: Colors.grey.withOpacity(0.5), // Separator color
        ),
      ],
    );
  }

  Future<QuerySnapshot> _fetchUserVehicleInfo() async {
    final user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      return await FirebaseFirestore.instance
          .collection('vehicles')
          .where('userId', isEqualTo: user.uid)
          .get();
    }
    throw Exception('User not found');
  }
}
```

My_reservations.dart:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyReservations());
}

class MyReservations extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primaryColor: Color(0xFFF86C1D),
        hintColor: Color(0xFF040415),
        textTheme: TextTheme(
          displayLarge: TextStyle(
            fontFamily: 'Roboto Serif',
            fontSize: 22,
            fontWeight: FontWeight.bold,
```

```
            color: Color(0xFFF86C1D),
          ),
          bodyLarge: TextStyle(
            fontFamily: 'Roboto Serif',
            fontSize: 18,
            color: Colors.black87,
          ),
          bodyMedium: TextStyle(
            fontFamily: 'Roboto Serif',
            fontSize: 16,
            color: Colors.black54,
          ),
        ),
        buttonTheme: ButtonThemeData(
          buttonColor: Color(0xFFF86C1D),
          textTheme: ButtonTextTheme.primary,
        ),
        appBarTheme: AppBarTheme(
          backgroundColor: Color(0xFFF86C1D),
        ),
      ),
      home: ReservationScreen(),
    );
  }
}

class ReservationScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'My Reservations',
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: ReservationList(),
    );
  }
}

class ReservationList extends StatefulWidget {
  @override
  _ReservationListState createState() => _ReservationListState();
}

class _ReservationListState extends State<ReservationList>
    with SingleTickerProviderStateMixin {
  late AnimationController _animationController;
  late Animation<double> _animation;
```

```dart
@override
void initState() {
  super.initState();
  _animationController = AnimationController(
    duration: const Duration(milliseconds: 500),
    vsync: this,
  );

  _animation = CurvedAnimation(
    parent: _animationController,
    curve: Curves.easeIn,
  );

  _animationController.forward();
}

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

Future<List<QueryDocumentSnapshot>> _fetchReservations() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    final querySnapshot = await FirebaseFirestore.instance
        .collection('reservations')
        .where('userId', isEqualTo: user.uid)
        .orderBy('timestamp', descending: true)
        .get();
    return querySnapshot.docs;
  }
  return [];
}

Future<String> _fetchPackageName(String packageId) async {
  final doc = await FirebaseFirestore.instance
      .collection('packages')
      .doc(packageId)
      .get();
  return doc['package_name'];
}

Future<Map<String, String>> _fetchVehicleDetails(String vehicleId) async {
  final doc = await FirebaseFirestore.instance
      .collection('vehicles')
      .doc(vehicleId)
      .get();
  return {
```

```
      'brand': doc['brand'],
      'model': doc['model'],
    };
  }

  void _cancelReservation(String reservationId) async {
    await FirebaseFirestore.instance
        .collection('reservations')
        .doc(reservationId)
        .delete();
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Reservation canceled successfully!')),
    );
    setState(() {});
  }

  Future<void> _showCancellationConfirmationDialog(String reservationId) async {
    return showDialog<void>(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text('Cancel Reservation'),
          content: Text('Are you sure you want to cancel your reservation?'),
          actions: <Widget>[
            TextButton(
              child: Text('No'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
            TextButton(
              child: Text('Yes'),
              onPressed: () {
                Navigator.of(context).pop();
                _cancelReservation(reservationId);
              },
            ),
          ],
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    final user = FirebaseAuth.instance.currentUser;

    if (user == null) {
      return Center(
        child: Text('Please log in to view your reservations.'),
```

```
    );
  }

  return FutureBuilder<List<QueryDocumentSnapshot>>(
    future: _fetchReservations(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(child: CircularProgressIndicator());
      }

      if (snapshot.hasError) {
        print('Error fetching reservations: ${snapshot.error}');
        return Center(
            child: Text('Error fetching reservations: ${snapshot.error}'));
      }

      if (!snapshot.hasData || snapshot.data!.isEmpty) {
        print('No reservations found for user ID: ${user.uid}');
        return Center(child: Text('No reservations found.'));
      }

      var reservations = snapshot.data!;
      print('Found ${reservations.length} reservations');

      return ListView.builder(
        itemCount: reservations.length,
        itemBuilder: (context, index) {
          var reservation = reservations[index];
          var reservationId = reservation.id;
          var packageId = reservation['packageId'];
          var vehicleId = reservation['vehicleId'];
          var timestamp = reservation['timestamp'] as Timestamp?;
          var appointmentDateTime =
              (reservation['appointmentDateTime'] as Timestamp?)?.toDate();

          return FutureBuilder<String>(
            future: _fetchPackageName(packageId),
            builder: (context, packageSnapshot) {
              if (!packageSnapshot.hasData) {
                return Center(child: CircularProgressIndicator());
              }

              return FutureBuilder<Map<String, String>>(
                future: _fetchVehicleDetails(vehicleId),
                builder: (context, vehicleSnapshot) {
                  if (!vehicleSnapshot.hasData) {
                    return Center(child: CircularProgressIndicator());
                  }

                  var vehicleDetails = vehicleSnapshot.data!;
```

```
                    return FadeTransition(
                      opacity: _animation,
                      child: Card(
                        margin: EdgeInsets.symmetric(
                            vertical: 10.0, horizontal: 20.0),
                        child: ListTile(
                          title: Text('Package: ${packageSnapshot.data}',
                              style: Theme.of(context).textTheme.displayLarge),
                          subtitle: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Text(
                                  'Vehicle: ${vehicleDetails['brand']}
${vehicleDetails['model']}',
                                  style: Theme.of(context).textTheme.bodyLarge),
                              Text(
                                  'Reservation Time: ${appointmentDateTime != null ?
appointmentDateTime.toString() : 'Unknown'}',
                                  style:
                                      Theme.of(context).textTheme.bodyMedium),
                            ],
                          ),
                          isThreeLine: true,
                          trailing: TextButton(
                            child: Text(
                              'Cancel',
                              style: TextStyle(color: Colors.red),
                            ),
                            onPressed: () {
                              _showCancellationConfirmationDialog(
                                  reservationId);
                            },
                          ),
                          onTap: () {
                            if (appointmentDateTime != null) {
                              showDialog(
                                context: context,
                                builder: (context) {
                                  return AlertDialog(
                                    title: Text('Reservation Details'),
                                    content: Text(
                                        'Package: ${packageSnapshot.data}\n'
                                        'Vehicle: ${vehicleDetails['brand']}
${vehicleDetails['model']}\n'
                                        'Appointment Date and Time: $appointmentDateTime'),
                                    actions: [
                                      TextButton(
                                        onPressed: () {
                                          Navigator.of(context).pop();
```

```
                                    },
                                    child: Text('OK'),
                                  ),
                                ],
                              );
                            },
                          );
                        }
                      },
                    ),
                  ),
                );
              },
            );
          },
        );
      },
    );
  }
}

class Packagess extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Packages'),
        actions: [
          IconButton(
            icon: Icon(Icons.list),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => ReservationScreen()),
              );
            },
          ),
        ],
      ),
      body: SingleChildScrollView(
        padding: EdgeInsets.all(20.0),
        child: StreamBuilder(
          stream: FirebaseFirestore.instance.collection('packages').snapshots(),
          builder: (context, snapshot) {
            if (!snapshot.hasData) {
              return Center(
                child: CircularProgressIndicator(),
              );
```

```
          }
          var packages = snapshot.data!.docs;
          return Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: packages.map<Widget>((package) {
              var packageName = package['package_name'];
              var packageServices =
                  List<String>.from(package['package_services']);
              var packageId = package.id;
              return _buildPackageBox(
                  context, packageName, packageServices, packageId);
            }).toList(),
          );
        },
      ),
    ),
  );
}

Widget _buildPackageBox(BuildContext context, String title,
    List<String> features, String packageId) {
  return Container(
    width: double.infinity,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(10.0),
      border: Border.all(color: Colors.grey, width: 0.5),
      color: Colors.white,
      boxShadow: [
        BoxShadow(
          color: Colors.black.withOpacity(0.1),
          offset: Offset(4.0, 4.0),
          blurRadius: 15.0,
          spreadRadius: 1.0,
        ),
      ],
    ),
    margin: EdgeInsets.symmetric(horizontal: 5.0, vertical: 10.0),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            title,
            style: Theme.of(context).textTheme.displayLarge?.copyWith(
                  decoration: TextDecoration.underline,
                  fontSize: 20.0,
                ),
          ),
          SizedBox(height: 16.0),
```

```
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: features
                .map((feature) => Padding(
                      padding: const EdgeInsets.only(bottom: 16.0),
                      child: Row(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Icon(
                            Icons.circle,
                            size: 10,
                            color: Colors.grey,
                          ),
                          SizedBox(width: 8.0),
                          Expanded(
                            child: Text(
                              feature,
                              style: Theme.of(context).textTheme.bodyLarge,
                            ),
                          ),
                        ],
                      ),
                    ))
                .toList(),
          ),
          SizedBox(height: 16.0),
          _buildSubscriptionBox(context, packageId),
        ],
      ),
    ),
  );
 }

 Widget _buildSubscriptionBox(BuildContext context, String packageId) {
   return ElevatedButton(
     onPressed: () {
       Navigator.push(
         context,
         MaterialPageRoute(
             builder: (context) => MyReservationScreen(packageId: packageId)),
       );
     },
     child: Text('Select Package'),
     style: ButtonStyle(
       backgroundColor: MaterialStateProperty.all<Color>(Color(0xFFF86C1D)),
     ),
   );
 }
}
```

```dart
class MyReservationScreen extends StatelessWidget {
  final String packageId;

  const MyReservationScreen({required this.packageId});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Reservation Details'),
      ),
      body: Padding(
        padding: EdgeInsets.all(20.0),
        child: VehicleSelection(packageId: packageId),
      ),
    );
  }
}

class VehicleSelection extends StatefulWidget {
  final String packageId;

  const VehicleSelection({required this.packageId});

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

class _VehicleSelectionState extends State<VehicleSelection> {
  String? selectedVehicleId;
  late Future<List<DocumentSnapshot>> vehiclesFuture;

  @override
  void initState() {
    super.initState();
    vehiclesFuture = _fetchVehicles();
  }

  Future<List<DocumentSnapshot>> _fetchVehicles() async {
    final user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      final vehiclesSnapshot = await FirebaseFirestore.instance
          .collection('vehicles')
          .where('userId', isEqualTo: user.uid)
          .get();
      return vehiclesSnapshot.docs;
    }
    return [];
  }
```

```
void _saveReservation() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null && selectedVehicleId != null) {
    await FirebaseFirestore.instance.collection('reservations').add({
      'userId': user.uid,
      'vehicleId': selectedVehicleId,
      'packageId': widget.packageId,
      'timestamp': FieldValue.serverTimestamp(),
      'appointmentDateTime':
          Timestamp.now(), // Add a timestamp for the appointment
    });
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Reservation saved successfully!')),
    );
    Navigator.pop(context);
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Please select a vehicle.')),
    );
  }
}

@override
Widget build(BuildContext context) {
  return FutureBuilder<List<DocumentSnapshot>>(
    future: vehiclesFuture,
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(child: CircularProgressIndicator());
      }
      if (!snapshot.hasData || snapshot.data!.isEmpty) {
        return Center(child: Text('No vehicles available.'));
      }
      var vehicles = snapshot.data!;
      return Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            'You have selected package:',
            style: Theme.of(context).textTheme.displayLarge,
          ),
          Text(
            widget.packageId,
            style: Theme.of(context).textTheme.bodyLarge,
          ),
          SizedBox(height: 20.0),
          Text(
            'Select a vehicle:',
            style: Theme.of(context).textTheme.displayLarge,
          ),
```

```
            SizedBox(height: 10.0),
            DropdownButton<String>(
              isExpanded: true,
              value: selectedVehicleId,
              hint: Text('Select a vehicle'),
              items: vehicles.map((vehicle) {
                var vehicleName = vehicle['brand'] + ' ' + vehicle['model'];
                var vehicleId = vehicle.id;
                return DropdownMenuItem<String>(
                  value: vehicleId,
                  child: Text(vehicleName),
                );
              }).toList(),
              onChanged: (value) {
                setState(() {
                  selectedVehicleId = value;
                });
              },
            ),
            SizedBox(height: 20.0),
            ElevatedButton(
              onPressed: _saveReservation,
              child: Text('Confirm Reservation'),
            ),
          ],
        );
      },
    );
  }
}
```

Notification_provider.dart:

```dart
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:firebase_messaging/firebase_messaging.dart';

class NotificationsProvider with ChangeNotifier {
  List<RemoteMessage> _messages = [];

  NotificationsProvider() {
    loadMessages();
  }

  List<RemoteMessage> get messages => _messages;

  void addMessage(RemoteMessage message) {
    _messages.add(message);
    saveMessages();
    notifyListeners();
  }

  void loadMessages() async {
    final prefs = await SharedPreferences.getInstance();
    final messagesData = prefs.getStringList('notifications') ?? [];
    _messages = messagesData
        .map((e) =>
            RemoteMessage.fromMap(Map<String, dynamic>.from(jsonDecode(e))))
        .toList();
    notifyListeners();
  }

  void saveMessages() async {
    final prefs = await SharedPreferences.getInstance();
    final messagesData = _messages.map((e) => jsonEncode(e.toMap())).toList();
    prefs.setStringList('notifications', messagesData);
  }
}
```

Notifications.dart:

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

class NotificationsPage extends StatelessWidget {
  final String userId;

  NotificationsPage({required this.userId});

  @override
  Widget build(BuildContext context) {
    print('User ID for notifications: $userId'); // Debugging

    return Scaffold(
      appBar: AppBar(
        title: Text(
          'User Notifications',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: Color(0xFFF76B1C), // Light orange color
      ),
      body: StreamBuilder<QuerySnapshot>(
        stream: FirebaseFirestore.instance
            .collection('notifications')
            .where('userId', isEqualTo: userId)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }

          if (!snapshot.hasData) {
            return Center(child: CircularProgressIndicator());
          }

          final notifications = snapshot.data!.docs;

          if (notifications.isEmpty) {
            return Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Icon(
                    Icons.notifications_none,
                    size: 100,
                    color: Colors.grey,
                  ),
                  SizedBox(height: 20),
                  Text('No notifications found.'),
```

```
                ],
              ),
            );
          }

          return RefreshIndicator(
            onRefresh: () async {
              // Implement your refresh logic here, e.g., refetch data
              // You can fetch data again from Firestore or perform any other operations
            },
            child: ListView.builder(
              itemCount: notifications.length,
              itemBuilder: (context, index) {
                var notification = notifications[index];
                return Card(
                  elevation: 2,
                  margin: EdgeInsets.symmetric(vertical: 5, horizontal: 10),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(15),
                  ),
                  child: ListTile(
                    title: Text(
                      notification['title'],
                      style: TextStyle(fontWeight: FontWeight.bold),
                    ),
                    subtitle: Text(notification['message']),
                  ),
                );
              },
            ),
          );
        },
      ),
    );
  }
}
```

Onboarding.dart:

```
import 'package:flutter/material.dart';
import 'login.dart'; // Import the LoginPage widget from login.dart

void main() {
  runApp(MaterialApp(
    home: OnboardingScreen(),
  ));
}

class OnboardingScreen extends StatefulWidget {
  @override
  _OnboardingScreenState createState() => _OnboardingScreenState();
}

class _OnboardingScreenState extends State<OnboardingScreen> {
  final PageController _pageController = PageController();
  final List<Map<String, String>> pages = [
    {
      'imagePath': 'assets/Parking-bro 1.png',
      'text':
          'Welcome to Safe Zone, the ultimate app for all your vehicle service and parking needs!\n\nWe understand that when you\'re traveling, you want the peace of mind knowing that your vehicle is in safe hands. That\'s why we\'ve designed an innovative platform that combines convenience, reliability, and creativity to provide you with top-notch services.',
    },
    {
      'imagePath': 'assets/Car_wash-bro_1.png',
      'text':
          'At Safe Zone, we believe that servicing your vehicle should be a hassle-free experience.\n\nWith just a few taps on our app, you can easily book a wide range of vehicle services, including routine maintenance, repairs, and even customization options. Our team of highly skilled technicians and mechanics are dedicated to delivering exceptional quality workmanship, ensuring that your vehicle receives the care it deserves.',
    },
    {
      'imagePath': 'assets/Parking-amico 1.png',
      'text':
          'But that\'s not all we offer.\nWe understand that finding secure parking spaces for your vehicle can be a daunting task, especially when you\'re away for an extended period.\n\nWith our long-term parking system, keep your vehicle safe and secure while you\'re traveling. Our state-of-the-art facilities are equipped with advanced security measures, including 24/7 surveillance, to give you peace of mind throughout your journey.',
    },
  ];
  int _currentPage = 0;

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      body: Column(
        children: [
          Expanded(
            flex: 3,
            child: PageView.builder(
              controller: _pageController,
              itemCount: pages.length,
              onPageChanged: (int page) {
                setState(() {
                  _currentPage = page;
                });
              },
              itemBuilder: (context, index) {
                return OnboardingPage(
                  imagePath: pages[index]['imagePath']!,
                  text: pages[index]['text']!,
                );
              },
            ),
          ),
          SizedBox(height: 20),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: List.generate(
              pages.length,
              (index) => Container(
                margin: EdgeInsets.symmetric(horizontal: 4),
                width: 10,
                height: 10,
                decoration: BoxDecoration(
                  shape: BoxShape.circle,
                  color: _currentPage == index ? Colors.orange : Colors.grey,
                ),
              ),
            ),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              Navigator.pushReplacement(
                context,
                PageRouteBuilder(
                  pageBuilder: (context, animation, secondaryAnimation) =>
                      LoginPage(),
                  transitionsBuilder:
                      (context, animation, secondaryAnimation, child) {
                    var begin = 0.0;
                    var end = 1.0;
                    var curve = Curves.easeInOut;
```

```
                  var tween = Tween(begin: begin, end: end).chain(
                    CurveTween(curve: curve),
                  );

                  return FadeTransition(
                    opacity: animation.drive(tween),
                    child: child,
                  );
                },
              ),
            );
          },
          style: ElevatedButton.styleFrom(
            foregroundColor: Colors.white,
            backgroundColor: Colors.orange, // Button color
          ),
          child: Text('Login'),
        ),
      ],
    ),
  );
 }
}

class OnboardingPage extends StatelessWidget {
  final String imagePath;
  final String text;

  OnboardingPage({
    required this.imagePath,
    required this.text,
  });

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Flexible(
          flex: 3,
          child: Padding(
            padding: EdgeInsets.symmetric(horizontal: 20),
            child: Image.asset(imagePath),
          ),
        ),
        Flexible(
          flex: 2,
          child: Padding(
            padding: EdgeInsets.all(20),
```

```
            child: Text(
              text,
              style: TextStyle(
                color: Colors.black,
                fontFamily: 'Roboto Serif',
                fontSize: 16, // Adjusted font size
                fontWeight: FontWeight.w700,
                height: 1.5,
              ),
            ),
          ),
        ),
      ],
    );
  }
}
```

Packages.dart:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:safe_zone/home.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(Reservation());
}

class Reservation extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Packages(),
    );
  }
}

class Packages extends StatelessWidget {
  final String? selectedPackageId; // Package ID selected by the user

  Packages({this.selectedPackageId});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
    appBar: AppBar(
      title: Text('Packages'),
      backgroundColor: Color(0xFFF86C1D),
    ),
    body: SingleChildScrollView(
      padding: EdgeInsets.all(20.0),
      child: StreamBuilder(
        stream: FirebaseFirestore.instance.collection('packages').snapshots(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return Center(
              child: CircularProgressIndicator(),
            );
          }
          var packages = snapshot.data!.docs;
          // Filter packages if selectedPackageId is not null
          if (selectedPackageId != null) {
            packages = packages
                .where((package) => package.id == selectedPackageId)
                .toList();
          }
          return Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: packages.map<Widget>((package) {
              var packageName = package['package_name'];
              var packageServices =
                  List<String>.from(package['package_services']);
              var packageId = package.id;
              var packagePrice = package['price'];
              var packageDuration = package['duration'];
              return _buildPackageBox(context, packageName, packageServices,
                  packageId, packagePrice, packageDuration);
            }).toList(),
          );
        },
      ),
    ),
  );
}

Widget _buildPackageBox(BuildContext context, String title,
    List<String> features, String packageId, packagePrice, packageDuration) {
  return Container(
    width: double.infinity,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(10.0),
      border: Border.all(color: Colors.black, width: 0.5),
      color: Color.fromRGBO(255, 255, 255, 1),
      boxShadow: [
        BoxShadow(
```

```
          color: Colors.black.withOpacity(0.3),
          offset: Offset(4.0, 4.0),
          blurRadius: 20.0,
          spreadRadius: 2.0,
        ),
      ],
    ),
    margin: EdgeInsets.symmetric(horizontal: 5.0, vertical: 10.0),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            title,
            style: TextStyle(
              color: Color(0xFF040415),
              fontFamily: 'Roboto Serif',
              fontSize: 22.0,
              fontWeight: FontWeight.w600,
              height: 0.90909,
              decoration: TextDecoration.underline,
            ),
          ),
          SizedBox(height: 16.0),
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: features
                .map((feature) => Padding(
                      padding: const EdgeInsets.only(bottom: 16.0),
                      child: Row(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Icon(
                            Icons.circle,
                            size: 10,
                            color: Color(0xFF040415),
                          ),
                          SizedBox(width: 8.0),
                          Expanded(
                            child: Text(
                              feature,
                              style: TextStyle(
                                color: Color(0xFF040415),
                                fontFamily: 'Roboto Serif',
                                fontSize: 18.0,
                                fontWeight: FontWeight.w400,
                                height: 1.11111,
                              ),
                            ),
```

```
                    ),
                  ],
                ),
              ))
          .toList(),
      ),
      SizedBox(height: 16.0),
      Row(
        children: [
          Text(
            'Price: ',
            style: TextStyle(
              color: Color(0xFF040415),
              fontFamily: 'Roboto Serif',
              fontSize: 18.0,
              fontWeight: FontWeight.w400,
              height: 1.11111,
            ),
          ),
          Text(
            '$packagePrice EGP',
            style: TextStyle(
              color: Color(0xFF040415),
              fontFamily: 'Roboto Serif',
              fontSize: 18.0,
              fontWeight: FontWeight.w400,
              height: 1.11111,
            ),
          ),
        ],
      ),
      SizedBox(height: 8.0),
      Row(
        children: [
          Text(
            '\nDuration: ',
            style: TextStyle(
              color: Color(0xFF040415),
              fontFamily: 'Roboto Serif',
              fontSize: 18.0,
              fontWeight: FontWeight.w400,
              height: 1.11111,
            ),
          ),
          Text(
            '\n$packageDuration Month',
            style: TextStyle(
              color: Color(0xFF040415),
              fontFamily: 'Roboto Serif',
              fontSize: 18.0,
```

```
                    fontWeight: FontWeight.w400,
                    height: 1.11111,
                  ),
                ),
              ],
            ),
            SizedBox(height: 16.0),
            Center(child: _buildSubscriptionBox(context, packageId)),
          ],
        ),
      ),
    );
  }

  Widget _buildSubscriptionBox(BuildContext context, String packageId) {
    return ElevatedButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => ReservationScreen(packageId: packageId)),
        );
      },
      style: ElevatedButton.styleFrom(
        foregroundColor: Colors.white,
        backgroundColor: Color(0xFFF86C1D), // foreground color
        padding: EdgeInsets.symmetric(vertical: 16.0, horizontal: 80.0),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(30.0),
        ),
      ),
      child: Text(
        'Select Package',
        style: TextStyle(
          fontSize: 18.0,
          fontWeight: FontWeight.bold,
        ),
      ),
    );
  }
}

class ReservationScreen extends StatelessWidget {
  final String packageId;

  const ReservationScreen({required this.packageId});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(
        title: Text('Reservation Details'),
        backgroundColor: Color(0xFFF86C1D),
      ),
      body: Padding(
        padding: EdgeInsets.all(20.0),
        child: VehicleSelection(packageId: packageId),
      ),
    );
  }
}

class VehicleSelection extends StatefulWidget {
  final String packageId;

  const VehicleSelection({required this.packageId});

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

class _VehicleSelectionState extends State<VehicleSelection> {
  String? selectedVehicleId;
  DateTime? selectedDate;
  TimeOfDay? selectedTime;
  late Future<List<DocumentSnapshot>> vehiclesFuture;

  @override
  void initState() {
    super.initState();
    vehiclesFuture = _fetchVehicles();
  }

  Future<List<DocumentSnapshot>> _fetchVehicles() async {
    final user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      final vehiclesSnapshot = await FirebaseFirestore.instance
          .collection('vehicles')
          .where('userId', isEqualTo: user.uid)
          .get();
      return vehiclesSnapshot.docs;
    }
    return [];
  }

  Future<void> _selectDate(BuildContext context) async {
    final DateTime? picked = await showDatePicker(
      context: context,
      initialDate: DateTime.now(),
      firstDate: DateTime.now(),
```

```
      lastDate: DateTime(2101),
    );
    if (picked != null && picked != selectedDate) {
      setState(() {
        selectedDate = picked;
      });
    }
  }

  Future<void> _selectTime(BuildContext context) async {
    final TimeOfDay? picked = await showTimePicker(
      context: context,
      initialTime: TimeOfDay.now(),
    );
    if (picked != null && picked != selectedTime)
      setState(() {
        selectedTime = picked;
      });
  }

  void _saveReservation() async {
    final user = FirebaseAuth.instance.currentUser;
    if (user != null &&
        selectedVehicleId != null &&
        selectedDate != null &&
        selectedTime != null) {
      final appointmentDateTime = DateTime(
        selectedDate!.year,
        selectedDate!.month,
        selectedDate!.day,
        selectedTime!.hour,
        selectedTime!.minute,
      );

      // Check the total reservations
      final reservationSnapshot =
          await FirebaseFirestore.instance.collection('reservations').get();
      if (reservationSnapshot.size >= 10) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('No parking slots available.')),
        );
        return;
      }

      // Save the reservation
      final reservationRef =
          await FirebaseFirestore.instance.collection('reservations').add({
        'userId': user.uid,
        'vehicleId': selectedVehicleId,
        'packageId': widget.packageId,
```

```
        'appointmentDateTime': appointmentDateTime,
        'timestamp': FieldValue.serverTimestamp(),
      });

      // Notify the admin
      await FirebaseFirestore.instance.collection('adminNotifications').add({
        'title': 'New Reservation',
        'userId': user.uid,
        'vehicleId': selectedVehicleId,
        'packageId': widget.packageId,
        'appointmentDateTime': appointmentDateTime,
        'reservationId': reservationRef.id,
        'timestamp': FieldValue.serverTimestamp(),
      });

      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Reservation saved successfully!')),
      );
      Navigator.pop(context);
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content:
                Text('Please select a vehicle and an appointment date/time.')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return FutureBuilder<List<DocumentSnapshot>>(
      future: vehiclesFuture,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(child: CircularProgressIndicator());
        }
        if (!snapshot.hasData || snapshot.data!.isEmpty) {
          return Center(child: Text('No vehicles available.'));
        }
        var vehicles = snapshot.data!;
        return Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              "Select a car to subscribe to package\n",
              style: TextStyle(fontSize: 20.0, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 20.0),
            Text(
              'Package ID: ${widget.packageId}\n',
```

```
            style: TextStyle(fontSize: 16.0),
          ),
          SizedBox(height: 20.0),
          Text(
            'Select a vehicle:',
            style: TextStyle(fontSize: 18.0, fontWeight: FontWeight.w500),
          ),
          ...vehicles.map((vehicle) {
            var vehicleId = vehicle.id;
            var vehicleName =
                '${vehicle['brand']} - ${vehicle['model']} - (${vehicle['plate']})';
            return RadioListTile<String>(
              title: Text(vehicleName),
              value: vehicleId,
              groupValue: selectedVehicleId,
              onChanged: (value) {
                setState(() {
                  selectedVehicleId = value;
                });
              },
            );
          }).toList(),
          Spacer(),
          SizedBox(height: 20.0),
          Center(
            // Center the buttons
            child: Column(
              children: [
                ElevatedButton(
                  onPressed: () => _selectDate(context),
                  style: ElevatedButton.styleFrom(
                    foregroundColor: Colors.white,
                    backgroundColor: Color(0xFFF86C1D), // foreground color
                    padding: EdgeInsets.symmetric(
                      vertical: 16.0,
                      horizontal: 118.0,
                    ),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(30.0),
                    ),
                  ),
                  child: Text(
                    selectedDate != null
                        ? 'Selected Date: ${selectedDate!.toLocal()}'
                            .split(' ')[0]
                        : 'Select Date',
                    style: TextStyle(
                      fontSize: 16.0,
                      fontWeight: FontWeight.bold,
                    ),
```

```
              ),
            ),
            SizedBox(height: 20.0),
            ElevatedButton(
              onPressed: () => _selectTime(context),
              style: ElevatedButton.styleFrom(
                foregroundColor: Colors.white,
                backgroundColor: Color(0xFFF86C1D), // foreground color
                padding: EdgeInsets.symmetric(
                    vertical: 16.0, horizontal: 116.0),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(30.0),
                ),
              ),
              child: Text(
                selectedTime == null
                    ? 'Select Time'
                    : 'Selected Time: ${selectedTime!.format(context)}',
                style: TextStyle(
                  fontSize: 16.0,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
            SizedBox(height: 20.0),
            ElevatedButton(
              onPressed: _saveReservation,
              style: ElevatedButton.styleFrom(
                foregroundColor: Colors.white,
                backgroundColor: Color(0xFFF86C1D), // foreground color
                padding: EdgeInsets.symmetric(
                    vertical: 16.0, horizontal: 100.0),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(30.0),
                ),
              ),
              child: Text(
                'Book Package',
                style: TextStyle(
                  fontSize: 18.0,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          ),
        ],
      ),
    ),
    SizedBox(
        height: 20.0), // Add some space between buttons and bottom edge
  ],
```

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

Profile.dart:

```
import 'package:cached_network_image/cached_network_image.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';

void main() => runApp(const Profile());

class Profile extends StatelessWidget {
  const Profile({Key? key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Profile Page',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const ProfilePage(),
    );
  }
}

class ProfilePage extends StatefulWidget {
  const ProfilePage({Key? key});

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

class _ProfilePageState extends State<ProfilePage> {
  File? _profileImage;
  File? _coverImage;
  String? _profileImgUrl;
  String? _coverImgUrl;
```

```dart
bool _isUploading = false;
Future<void>? _loadingFuture;

@override
void initState() {
  super.initState();
  _loadingFuture = _loadProfileImage();
}

Future<void> _loadProfileImage() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    try {
      final userData = await FirebaseFirestore.instance
          .collection('users')
          .doc(user.uid)
          .get();
      if (userData.exists) {
        setState(() {
          _profileImgUrl = userData['profileImgUrl'];
          _coverImgUrl = userData['coverImgUrl'];
        });
      }
    } catch (error) {
      print('Error loading profile data: $error');
    }
  }
}

Future<void> _pickImage(bool isCoverImage) async {
  final picker = ImagePicker();
  final pickedFile = await picker.pickImage(source: ImageSource.gallery);

  if (pickedFile != null) {
    setState(() {
      if (isCoverImage) {
        _coverImage = File(pickedFile.path);
      } else {
        _profileImage = File(pickedFile.path);
      }
    });
  }
}

Future<void> _uploadProfileAndCoverImages() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null && (_profileImage != null || _coverImage != null)) {
    setState(() {
      _isUploading = true;
    });
```

```
    try {
      if (_coverImage != null) {
        final coverFileName = 'cover_${user.uid}.jpg';
        final coverDestination = 'users/$coverFileName';
        final coverRef = FirebaseStorage.instance.ref(coverDestination);
        await coverRef.putFile(_coverImage!);
        final coverImageUrl = await coverRef.getDownloadURL();
        await FirebaseFirestore.instance
            .collection('users')
            .doc(user.uid)
            .update({'coverImgUrl': coverImageUrl});
        setState(() {
          _coverImgUrl = coverImageUrl;
          _coverImage = null;
        });
      }

      if (_profileImage != null) {
        final profileFileName = 'profile_${user.uid}.jpg';
        final profileDestination = 'users/$profileFileName';
        final profileRef = FirebaseStorage.instance.ref(profileDestination);
        await profileRef.putFile(_profileImage!);
        final profileImageUrl = await profileRef.getDownloadURL();
        await FirebaseFirestore.instance
            .collection('users')
            .doc(user.uid)
            .update({'profileImgUrl': profileImageUrl});
        setState(() {
          _profileImgUrl = profileImageUrl;
          _profileImage = null;
        });
      }

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Profile and cover pictures updated successfully'),
        backgroundColor: Colors.green,
      ));
    } catch (error) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Failed to update profile and cover pictures: $error'),
        backgroundColor: Colors.red,
      ));
    } finally {
      setState(() {
        _isUploading = false;
      });
    }
  }
}
```

```
void _showProfilePicture() {
  if (_profileImgUrl != null) {
    showDialog(
      context: context,
      builder: (context) => Dialog(
        child: Container(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              CachedNetworkImage(
                imageUrl: _profileImgUrl!,
                placeholder: (context, url) => CircularProgressIndicator(),
                errorWidget: (context, url, error) => Icon(Icons.error),
              ),
              const SizedBox(height: 16),
              ElevatedButton(
                onPressed: () => Navigator.of(context).pop(),
                child: Text('Close'),
              ),
            ],
          ),
        ),
      ),
    );
  } else {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text('No profile picture to display'),
      backgroundColor: Colors.red,
    ));
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SafeArea(
      child: SingleChildScrollView(
        child: FutureBuilder<void>(
          future: _loadingFuture,
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return Center(child: CircularProgressIndicator());
            } else if (snapshot.hasError) {
              return Center(child: Text('Error: ${snapshot.error}'));
            } else {
              return Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
```

```
                GestureDetector(
                  onTap: () => _pickImage(true),
                  child: Container(
                    width: double.infinity,
                    height: 212,
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(10),
                      boxShadow: const [
                        BoxShadow(
                          color: Color(0x3F000000),
                          blurRadius: 4,
                          offset: Offset(4, 5),
                        ),
                      ],
                      color: Colors.grey,
                    ),
                    child: _coverImage != null
                        ? Image.file(_coverImage!, fit: BoxFit.cover)
                        : _coverImgUrl != null
                            ? CachedNetworkImage(
                                imageUrl: _coverImgUrl!,
                                fit: BoxFit.cover,
                                placeholder: (context, url) => Center(
                                    child: CircularProgressIndicator()),
                                errorWidget: (context, url, error) =>
                                    Icon(Icons.error),
                              )
                            : Icon(Icons.add_a_photo,
                                size: 48, color: Colors.white),
                  ),
                ),
                const SizedBox(height: 16),
                Center(
                  child: Column(
                    children: [
                      GestureDetector(
                        onTap: () => _pickImage(false),
                        child: CircleAvatar(
                          radius: 50,
                          backgroundColor: Colors.grey,
                          backgroundImage: _profileImage != null
                              ? FileImage(_profileImage!) as ImageProvider
                              : _profileImgUrl != null
                                  ? CachedNetworkImageProvider(
                                      _profileImgUrl!)
                                  : null,
                          child: _profileImage == null &&
                                  _profileImgUrl == null
                              ? Icon(Icons.add_a_photo,
                                  size: 48, color: Colors.white)
```

```
                    : null,
              ),
            ),
            const SizedBox(height: 16),
            FutureBuilder<DocumentSnapshot>(
              future: _fetchUserInfo(),
              builder: (context, snapshot) {
                if (snapshot.connectionState ==
                    ConnectionState.waiting) {
                  return CircularProgressIndicator();
                } else if (snapshot.hasError) {
                  return Text('Error: ${snapshot.error}');
                } else if (snapshot.hasData &&
                    snapshot.data != null) {
                  final userData = snapshot.data!;
                  final firstName = userData['firstName'] ?? '';
                  final secondName = userData['secondName'] ?? '';

                  return Text(
                    '$firstName $secondName',
                    style: TextStyle(
                      fontSize: 18,
                      fontWeight: FontWeight.bold,
                    ),
                  );
                } else {
                  return SizedBox.shrink();
                }
              },
            ),
          ],
        ),
      ),
      const SizedBox(height: 16),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
            onPressed: _showProfilePicture,
            style: ElevatedButton.styleFrom(
              backgroundColor: const Color(0xFFFFE5B4),
              padding: const EdgeInsets.symmetric(
                  vertical: 8, horizontal: 12),
            ),
            child: Text(
              'Show Profile Picture',
              style: TextStyle(fontSize: 12, color: Colors.black),
            ),
          ),
          const SizedBox(width: 16),
```

```
                    ElevatedButton(
                      onPressed: _uploadProfileAndCoverImages,
                      style: ElevatedButton.styleFrom(
                        backgroundColor: const Color(0xFFFFE5B4),
                        padding: const EdgeInsets.symmetric(
                            vertical: 8, horizontal: 12),
                      ),
                      child: _isUploading
                          ? CircularProgressIndicator(
                              valueColor: AlwaysStoppedAnimation<Color>(
                                  Colors.black),
                            )
                          : Text(
                              'Update Profile& Cover Picture',
                              style: TextStyle(
                                  fontSize: 12, color: Colors.black),
                            ),
                    ),
                  ],
                ),
                const SizedBox(height: 32),
                FutureBuilder<DocumentSnapshot>(
                  future: _fetchUserInfo(),
                  builder: (context, snapshot) {
                    if (snapshot.connectionState ==
                        ConnectionState.waiting) {
                      return CircularProgressIndicator();
                    } else if (snapshot.hasError) {
                      return Text('Error: ${snapshot.error}');
                    } else if (snapshot.hasData && snapshot.data != null) {
                      final userData = snapshot.data!;
                      final firstName = userData['firstName'] ?? '';
                      final secondName = userData['secondName'] ?? '';
                      final phoneNumber = userData['phoneNumber'] ?? '';
                      final email = userData['email'] ?? '';

                      return Padding(
                        padding: const EdgeInsets.symmetric(horizontal: 16),
                        child: Column(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          children: [
                            _buildInfoRow('First Name:', firstName),
                            const SizedBox(height: 16),
                            _buildInfoRow('Last Name:', secondName),
                            const SizedBox(height: 16),
                            _buildInfoRow('Phone:', phoneNumber),
                            const SizedBox(height: 16),
                            _buildInfoRow('Email:', email),
                            const SizedBox(height: 16),
                            FutureBuilder<QuerySnapshot>(
```

```
                    future: _fetchUserVehicleInfo(),
                    builder: (context, snapshot) {
                      if (snapshot.connectionState ==
                          ConnectionState.waiting) {
                        return CircularProgressIndicator();
                      } else if (snapshot.hasError) {
                        return Text('Error: ${snapshot.error}');
                      } else if (snapshot.hasData &&
                          snapshot.data != null) {
                        final vehicleData = snapshot.data!;
                        final vehicleWidgets = <Widget>[];

                        for (var i = 0;
                            i < vehicleData.docs.length;
                            i++) {
                          final vehicle = vehicleData.docs[i];
                          final brand = vehicle['brand'] ?? '';
                          final model = vehicle['model'] ?? '';
                          final carImageUrl =
                              vehicle['carImageUrl'] ?? '';
                          final carNumber = i + 1;

                          vehicleWidgets.add(
                            Container(
                              padding: const EdgeInsets.all(16),
                              margin: const EdgeInsets.symmetric(
                                  vertical: 8),
                              decoration: BoxDecoration(
                                color: const Color(0xFFFFE5B4),
                                borderRadius:
                                    BorderRadius.circular(10),
                              ),
                              child: Column(
                                crossAxisAlignment:
                                    CrossAxisAlignment.start,
                                children: [
                                  _buildInfoRow(
                                      'Car $carNumber Brand:',
                                      brand),
                                  const SizedBox(height: 8),
                                  _buildInfoRow(
                                      'Car $carNumber Model:',
                                      model),
                                  Padding(
                                    padding:
                                        const EdgeInsets.only(
                                            top: 8),
                                    child: ClipRRect(
                                      borderRadius:
                                          BorderRadius.circular(
```

```
                                        10),
                                    child: CachedNetworkImage(
                                      imageUrl: carImageUrl,
                                      width: double.infinity,
                                      height: 200,
                                      fit: BoxFit.cover,
                                      placeholder: (context,
                                              url) =>
                                          Center(
                                              child:
                                                  CircularProgressIndicator()
),
                                      errorWidget: (context,
                                              url, error) =>
                                          Icon(Icons.error),
                                    ),
                                  ),
                                ),
                                const SizedBox(height: 8),
                                Row(
                                  mainAxisAlignment:
                                      MainAxisAlignment
                                          .spaceBetween,
                                  children: [
                                    Text(
                                      'Subscribed to:',
                                      style: TextStyle(
                                        fontSize: 16,
                                        fontWeight:
                                            FontWeight.w500,
                                      ),
                                    ),
                                    Text(
                                      'Gold Package',
                                      style: TextStyle(
                                        color: const Color(
                                            0xFFF76B1C),
                                        fontSize: 16,
                                        fontWeight:
                                            FontWeight.w500,
                                      ),
                                    ),
                                  ],
                                ),
                              ],
                            ),
                          ),
                        );
                      }
```

```
                                    return Column(
                                      crossAxisAlignment:
                                          CrossAxisAlignment.start,
                                      children: vehicleWidgets,
                                    );
                                  } else {
                                    return SizedBox.shrink();
                                  }
                                },
                              ),
                            ],
                          ),
                        );
                      } else {
                        return SizedBox.shrink();
                      }
                    },
                  ),
                ],
              );
            }
          },
        ),
      ),
    ),
  );
}

Widget _buildInfoRow(String label, String value) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Text(
            label,
            style: TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.w500,
            ),
          ),
          Text(
            value,
            style: TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.w500,
            ),
          ),
        ],
```

```
      ),
      const SizedBox(height: 8),
      Container(
        height: 1,
        color: Colors.grey.withOpacity(0.5),
      ),
    ],
  );
}

Future<DocumentSnapshot> _fetchUserInfo() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    return await FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .get();
  }
  throw Exception('User not found');
}

Future<QuerySnapshot> _fetchUserVehicleInfo() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    return await FirebaseFirestore.instance
        .collection('vehicles')
        .where('userId', isEqualTo: user.uid)
        .get();
  }
  throw Exception('User not found');
}

Future<void> _updateFirstName(String firstName) async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    await FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .update({'firstName': firstName});
  }
}

Future<void> _updateSecondName(String secondName) async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    await FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .update({'secondName': secondName});
  }
```

```
  }
}
```

Services.dart:

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:safe_zone/home.dart';

void main() {
  runApp(Services());
}

class Services extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text(
            'Our Services',
            style: TextStyle(
              color: Color(0xFFF86C1D),
              fontFamily: 'Roboto Serif',
              fontSize: 40,
              fontStyle: FontStyle.normal,
              fontWeight: FontWeight.w500,
              height: 0.5,
            ),
          ),
          centerTitle: true,
          backgroundColor: Color.fromRGBO(237, 237, 237, 0.5),
          toolbarHeight: 80,
        ),
        body: ServicesList(),
      ),
    );
  }
}

class ServicesList extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Column(
        children: [
          SizedBox(height: 35),
```

```
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 30),
            child: Container(
              width: 350,
              height: 264,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(10),
                border: Border.all(color: Colors.black, width: 0.5),
                color: Colors.white,
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withOpacity(0.3),
                    blurRadius: 20,
                    offset: Offset(4, 4),
                  ),
                ],
              ),
              child: Padding(
                padding: EdgeInsets.only(left: 33, right: 14, top: 15),
                child: Text(
                  'Welcome to our comprehensive car services and storage facility! We are
dedicated to providing exceptional automotive services and secure storage solutions for our
valued customers. Whether you need routine maintenance, repairs, or a safe place to store
your vehicle, we have you covered.',
                  style: TextStyle(
                    color: Color(0xFF040415),
                    fontFamily: 'Roboto Serif',
                    fontSize: 15,
                    fontStyle: FontStyle.normal,
                    fontWeight: FontWeight.w400,
                    height: 1.5,
                  ),
                ),
              ),
            ),
          ),
          SizedBox(height: 20),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 30),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                _buildServiceBox('1.jpg', 'Car Washing', context),
                SizedBox(height: 20),
                _buildServiceBox('2.jpg', 'Car Detailing', context),
                SizedBox(height: 20),
                _buildServiceBox('3.jpg', 'Car Tuning', context),
                SizedBox(height: 20),
                _buildServiceBox('4.jpeg', 'Oil Change', context),
                SizedBox(height: 20),
```

```
              _buildServiceBox('5.jpg', 'Tire Rotation', context),
            ],
          ),
        ),
      ],
    ),
  );
}

Widget _buildServiceBox(
    String imagePath, String title, BuildContext context) {
  return FutureBuilder<DocumentSnapshot>(
    future: FirebaseFirestore.instance
        .collection('services')
        .where('service_name', isEqualTo: title)
        .limit(1)
        .get()
        .then((value) => value.docs.first),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return CircularProgressIndicator();
      }
      if (!snapshot.hasData || !snapshot.data!.exists) {
        return SizedBox(); // Return an empty widget if data doesn't exist
      }

      var serviceId = snapshot.data!.id;
      var serviceName = snapshot.data!['service_name'];

      return Container(
        height: 62,
        width: 309,
        margin: EdgeInsets.symmetric(vertical: 2),
        padding: EdgeInsets.only(
            left: 0), // Add padding to move content to the left
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(60),
          border: Border.all(color: Color(0xFFF86C1D), width: 2),
          color: Colors.white,
          boxShadow: [
            BoxShadow(
              color: Colors.black.withOpacity(0.25),
              blurRadius: 10,
              offset: Offset(8, 8),
            ),
          ],
        ),
        child: Row(
          mainAxisAlignment:
              MainAxisAlignment.spaceBetween, // Align children to the ends
```

```
          children: [
            Container(
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(26),
                border: Border.all(
                    color: Color(0xFFF86C1D), width: 2), // Border color
              ),
              child: ClipRRect(
                borderRadius: BorderRadius.circular(
                    24), // Adjust border radius to accommodate the border width
                child: Image.asset(
                  'assets/$imagePath',
                  width: 52,
                  height: 52,
                  fit: BoxFit.cover,
                ),
              ),
            ),
            SizedBox(width: 10), // Add spacing between image and text
            Text(
              serviceName,
              style: TextStyle(
                color: Color(0xFF040415),
                fontFamily: 'Roboto Serif',
                fontSize: 16,
                fontStyle: FontStyle.normal,
                fontWeight: FontWeight.w400,
                height: 1.25,
              ),
            ),
            GestureDetector(
              onTap: () {},
              child: SizedBox(
                width: 80,
                height: 55,
                child: ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) =>
                            ServiceDetailPage(serviceId: serviceId),
                      ),
                    );
                  },
                  child: Text(
                    'Subscribe',
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 10,
```

```
                    ),
                  ),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Color(0xFFF86C1D),
                    padding: EdgeInsets.symmetric(horizontal: 8),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  },
 );
 }
}

class ServiceDetailPage extends StatelessWidget {
  final String serviceId;

  ServiceDetailPage({required this.serviceId});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Service Details'),
        backgroundColor: Color(0xFFF86C1D),
      ),
      body: FutureBuilder<DocumentSnapshot>(
        future: FirebaseFirestore.instance
            .collection('services')
            .doc(serviceId)
            .get(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          if (!snapshot.hasData || !snapshot.data!.exists) {
            return Center(child: Text('Service not found'));
          }

          var serviceData = snapshot.data!.data() as Map<String, dynamic>;
          var serviceName = serviceData['service_name'];
          var serviceDetails = serviceData['service_details'];
          var servicePrice = serviceData['service_price'];
          var times = serviceData['times'];

          return Padding(
            padding: const EdgeInsets.all(16.0),
```

```
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text('Service Name: $serviceName',
                  style:
                      TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
              SizedBox(height: 8),
              Text('\nDetails: $serviceDetails',
                  style: TextStyle(fontSize: 16)),
              SizedBox(height: 8),
              Text('\nPrice: ${servicePrice.toString()} EGP',
                  style: TextStyle(fontSize: 16)),
              SizedBox(height: 8),
              Text('Times: $times', style: TextStyle(fontSize: 16)),
              SizedBox(height: 16),
              Center(
                child: ElevatedButton(
                  onPressed: () async {
                    var user = FirebaseAuth.instance.currentUser;
                    if (user != null) {
                      var vehicles = await FirebaseFirestore.instance
                          .collection('vehicles')
                          .where('userId', isEqualTo: user.uid)
                          .get();
                      if (vehicles.docs.isEmpty) {
                        ScaffoldMessenger.of(context).showSnackBar(
                            SnackBar(content: Text('No vehicles found')));
                        return;
                      }

                      var selectedVehicleId = await showDialog<String>(
                        context: context,
                        builder: (context) =>
                            ChooseVehicleDialog(vehicles: vehicles.docs),
                      );

                      if (selectedVehicleId != null) {
                        var confirmed = await showDialog<bool>(
                          context: context,
                          builder: (context) => AlertDialog(
                            title: Text('Confirm Request'),
                            content:
                                Text('Do you want to request this service?'),
                            actions: [
                              TextButton(
                                onPressed: () =>
                                    Navigator.pop(context, false),
                                child: Text('Cancel'),
                              ),
                              TextButton(
```

```
                    onPressed: () => Navigator.pop(context, true),
                    child: Text('Confirm'),
                  ),
                ],
              ),
            );

            if (confirmed == true) {
              await FirebaseFirestore.instance
                  .collection('requests')
                  .add({
                'serviceId': serviceId,
                'userId': user.uid,
                'vehicleId': selectedVehicleId,
                'timestamp': FieldValue.serverTimestamp(),
              });
              await FirebaseFirestore.instance
                  .collection('adminNotifications')
                  .add({
                'userId': user.uid,
                'serviceId': serviceId,
                'vehicleId': selectedVehicleId,
                'timestamp': FieldValue.serverTimestamp(),
              });

              ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Request submitted')));
              Navigator.pop(context);
            }
          }
        } else {
          ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(content: Text('User not logged in')));
        }
      },
      child: Text(
        'Request This Service',
        style: TextStyle(color: Colors.white),
      ),
      style: ElevatedButton.styleFrom(
        backgroundColor: Color(0xFFF86C1D),
        padding: EdgeInsets.symmetric(
            vertical: 16.0,
            horizontal: 50.0), // Adjust padding for button size
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(
              30.0), // Adjust border radius for button shape
        ),
      ),
    ),
```

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

class ChooseVehicleDialog extends StatelessWidget {
 final List<QueryDocumentSnapshot> vehicles;

 ChooseVehicleDialog({required this.vehicles});

 @override
 Widget build(BuildContext context) {
  return AlertDialog(
   title: Text('Choose Vehicle'),
   content: SingleChildScrollView(
    child: Column(
     children: vehicles.map((vehicle) {
      var vehicleData = vehicle.data() as Map<String, dynamic>;
      var vehicleName = (vehicleData['brand'] ?? 'Unknown Brand') +
          ' - ' +
          (vehicleData['model'] ?? 'Unknown Model');
      return ListTile(
       title: Text(vehicleName),
       onTap: () => Navigator.pop(context, vehicle.id),
      );
     }).toList(),
    ),
   ),
  );
 }
}
```

Signup.dart:

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart'; // Added Firestore import
import 'package:safe_zone/login.dart';
import 'home.dart';
import 'terms_and_conditions.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: SignUpScreen(),
    );
  }
}

class SignUpScreen extends StatefulWidget {
  @override
  _SignUpScreenState createState() => _SignUpScreenState();
}

class _SignUpScreenState extends State<SignUpScreen> {
  bool _isChecked = false;
  late TextEditingController _emailController;
  late TextEditingController _passwordController;
  late TextEditingController _confirmPasswordController;
  late TextEditingController _birthdateController;
  late TextEditingController _firstNameController;
  late TextEditingController _secondNameController;
  late TextEditingController _phoneNumberController;
  late TextEditingController _areaController; // New controller for area
  late TextEditingController _streetController; // New controller for street
  late String _selectedGender = 'Male';
  late String _selectedCity = 'Cairo';
  String? _errorMessage;

  @override
  void initState() {
    super.initState();
    _emailController = TextEditingController();
    _passwordController = TextEditingController();
    _confirmPasswordController = TextEditingController();
    _birthdateController = TextEditingController();
    _firstNameController = TextEditingController();
```

```
    _secondNameController = TextEditingController();
    _phoneNumberController = TextEditingController();
    _areaController = TextEditingController(); // Initialize area controller
    _streetController = TextEditingController(); // Initialize street controller
  }

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    _confirmPasswordController.dispose();
    _birthdateController.dispose();
    _firstNameController.dispose();
    _secondNameController.dispose();
    _phoneNumberController.dispose();
    _areaController.dispose(); // Dispose of area controller
    _streetController.dispose(); // Dispose of street controller
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              SizedBox(height: 48),
              Text(
                'Sign up',
                style: TextStyle(
                  color: Color(0xFFF86C1D),
                  fontSize: 24,
                  fontWeight: FontWeight.w600,
                  fontFamily: 'Roboto Serif',
                ),
              ),
              SizedBox(height: 24),
              Text(
                'Find your dream park!',
                style: TextStyle(
                  color: Color(0xFF040415),
                  fontSize: 20,
                  fontWeight: FontWeight.normal,
                  fontFamily: 'Roboto Serif',
                ),
              ),
              InputField(
                labelText: 'First Name',
```

```
            iconData: Icons.person,
            controller: _firstNameController,
          ),
          InputField(
            labelText: 'Second Name',
            iconData: Icons.person,
            controller: _secondNameController,
          ),
          InputField(
            labelText: 'Email',
            iconData: Icons.email,
            controller: _emailController,
            keyboardType: TextInputType.emailAddress,
          ),
          InputField(
            labelText: 'Phone Number',
            iconData: Icons.phone,
            controller: _phoneNumberController,
          ),
          InputField(
            labelText: 'Password',
            iconData: Icons.lock,
            controller: _passwordController,
            isPassword: true,
          ),
          InputField(
            labelText: 'Confirm Password',
            iconData: Icons.lock,
            controller: _confirmPasswordController,
            isPassword: true,
          ),
          DropdownField(
            labelText: 'Gender',
            iconData: Icons.person,
            value: _selectedGender,
            onChanged: (String? newValue) {
              setState(() {
                _selectedGender = newValue!;
              });
            },
            items: <String>['Male', 'Female']
                .map<DropdownMenuItem<String>>((String value) {
              return DropdownMenuItem<String>(
                value: value,
                child: Text(value),
              );
            }).toList(),
          ),
          GestureDetector(
            onTap: () async {
```

```
            final DateTime? picked = await showDatePicker(
              context: context,
              initialDate: DateTime.now(),
              firstDate: DateTime(1900),
              lastDate: DateTime.now(),
            );
            if (picked != null) {
              setState(() {
                _birthdateController.text = picked.day.toString() +
                    '/' +
                    picked.month.toString() +
                    '/' +
                    picked.year.toString();
              });
            }
          },
          child: InputField(
            labelText: 'Birthdate',
            iconData: Icons.cake,
            controller: _birthdateController,
            enabled: false,
          ),
        ),
        DropdownField(
          labelText: 'City',
          iconData: Icons.location_city,
          value: _selectedCity,
          onChanged: (String? newValue) {
            setState(() {
              _selectedCity = newValue!;
            });
          },
          items: <String>['Cairo', 'Alexandria', 'Luxor', 'Aswan']
              .map<DropdownMenuItem<String>>((String value) {
            return DropdownMenuItem<String>(
              value: value,
              child: Text(value),
            );
          }).toList(),
        ),
        InputField(
            labelText: 'Area',
            iconData: Icons.location_on,
            controller: _areaController), // Add area input field
        InputField(
            labelText: 'Street',
            iconData: Icons.location_on,
            controller: _streetController), // Add street input field
        SizedBox(height: 24),
        ElevatedButton(
```

```
              onPressed: () async {
                if (_isInputValid()) {
                  try {
                    final credential = await FirebaseAuth.instance
                        .createUserWithEmailAndPassword(
                      email: _emailController.text,
                      password: _passwordController.text,
                    );

                    // Log in the user immediately after sign-up
                    await FirebaseAuth.instance.signInWithEmailAndPassword(
                      email: _emailController.text,
                      password: _passwordController.text,
                    );

                    await FirebaseFirestore.instance
                        .collection('users')
                        .doc(credential.user!.uid)
                        .set({
                      'email': _emailController.text,
                      'firstName': _firstNameController.text,
                      'secondName': _secondNameController.text,
                      'phoneNumber': _phoneNumberController.text,
                      'gender': _selectedGender,
                      'birthdate': _birthdateController.text,
                      'city': _selectedCity,
                      'area': _areaController.text,
                      'street': _streetController.text,
                      'isAdmin': false,
                      'profileImgUrl': '',
                      'coverImgUrl': '',
                    });
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(builder: (context) => HomeScreen()),
                    );
                  } on FirebaseAuthException catch (e) {
                    if (e.code == 'weak-password') {
                      setState(() {
                        _errorMessage = 'The password provided is too weak.';
                      });
                    } else if (e.code == 'email-already-in-use') {
                      setState(() {
                        _errorMessage =
                            'The account already exists for that email.';
                      });
                    }
                  } catch (e) {
                    print(e);
                  }
```

```
            }
          },
          style: ButtonStyle(
            backgroundColor: MaterialStateProperty.all(Color(0xFFF86C1D)),
            elevation: MaterialStateProperty.all(4),
            minimumSize: MaterialStateProperty.all(Size(360, 56)),
          ),
          child: Text(
            'Sign Up',
            style: TextStyle(
              color: Color(0xFFFBFBFB),
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
        if (_errorMessage != null)
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 8),
            child: Text(
              _errorMessage!,
              style: TextStyle(color: Colors.red),
            ),
          ),
        SizedBox(height: 16),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Already have an account? ',
              style: TextStyle(
                color: Color(0xFF040415),
                fontSize: 14,
                fontWeight: FontWeight.bold,
              ),
            ),
            TextButton(
              onPressed: () {
                Navigator.pushReplacement(
                  context,
                  MaterialPageRoute(builder: (context) => LoginPage()),
                  // Remove all routes from the stack
                );
              },
              child: Text(
                'Log In',
                style: TextStyle(
                  color: Color(0xFFF86C1D),
                  fontSize: 14,
                  fontWeight: FontWeight.bold,
```

```
            ),
          ),
        ),
      ],
    ),
    GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) => TermsAndConditionsPage()),
        );
      },
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            'Terms & Conditions ',
            style: TextStyle(
              color: Color(0xFFF86C1D),
              fontSize: 14,
              fontWeight: FontWeight.bold,
            ),
          ),
          Icon(
            Icons.arrow_forward,
            color: Color(0xFFF86C1D),
          ),
        ],
      ),
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Checkbox(
          value: _isChecked,
          onChanged: (bool? value) {
            setState(() {
              _isChecked = value ?? false;
            });
          },
          materialTapTargetSize: MaterialTapTargetSize.shrinkWrap,
          checkColor: Colors.white,
        ),
        Text(
          'I agree to the ',
          style: TextStyle(
            color: Color(0xFF040415),
            fontSize: 14,
            fontWeight: FontWeight.bold,
```

```
                ),
              ),
              GestureDetector(
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => TermsAndConditionsPage()),
                  );
                },
                child: Text(
                  'Terms & Conditions',
                  style: TextStyle(
                    color: Color(0xFFF86C1D),
                    fontSize: 14,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  );
}

bool _isInputValid() {
  if (_emailController.text.isEmpty ||
      _passwordController.text.isEmpty ||
      _confirmPasswordController.text.isEmpty ||
      _birthdateController.text.isEmpty ||
      _firstNameController.text.isEmpty ||
      _secondNameController.text.isEmpty ||
      _phoneNumberController.text.isEmpty) {
    setState(() {
      _errorMessage = 'All fields are required.';
    });
    return false;
  } else if (_passwordController.text != _confirmPasswordController.text) {
    setState(() {
      _errorMessage = 'Passwords do not match.';
    });
    return false;
  } else if (!_isChecked) {
    setState(() {
      _errorMessage = 'Please accept the terms and conditions.';
    });
    return false;
```

```
    }
    return true;
  }
}

class InputField extends StatefulWidget {
  final String labelText;
  final IconData iconData;
  final TextEditingController? controller;
  final bool isPassword;
  final TextInputType? keyboardType;
  final bool enabled;

  InputField({
    required this.labelText,
    required this.iconData,
    this.controller,
    this.isPassword = false,
    this.keyboardType,
    this.enabled = true,
  });

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

class _InputFieldState extends State<InputField> {
  late FocusNode _focusNode;
  String _errorMessage = '';

  @override
  void initState() {
    super.initState();
    _focusNode = FocusNode();
  }

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

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: EdgeInsets.symmetric(vertical: 12, horizontal: 24),
      padding: EdgeInsets.symmetric(horizontal: 16),
      decoration: BoxDecoration(
        color: Colors.white,
        boxShadow: [
```

```
          BoxShadow(
            color: Color(0xFF000000).withOpacity(0.25),
            offset: Offset(0, 4),
            blurRadius: 4,
          ),
        ],
        borderRadius: BorderRadius.circular(8),
        border: _focusNode.hasFocus || _errorMessage.isNotEmpty
            ? Border.all(
                color:
                    _errorMessage.isNotEmpty ? Colors.red : Color(0xFFF86C1D),
              )
            : null,
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              Icon(
                widget.iconData,
                size: 32,
                color:
                    _focusNode.hasFocus ? Color(0xFFF86C1D) : Color(0xFFA8AFB9),
              ),
              SizedBox(width: 16),
              Expanded(
                child: TextField(
                  focusNode: _focusNode,
                  controller: widget.controller,
                  obscureText: widget.isPassword,
                  keyboardType: widget.keyboardType,
                  enabled: widget.enabled,
                  onChanged: (value) {
                    setState(() {
                      _errorMessage =
                          value.isEmpty ? 'This field is required.' : '';
                    });
                  },
                  decoration: InputDecoration(
                    labelText: widget.labelText,
                    labelStyle: TextStyle(
                      fontSize: _focusNode.hasFocus ? 16 : 14,
                      fontWeight: FontWeight.w500,
                      color: _focusNode.hasFocus
                          ? Color(0xFFF86C1D)
                          : Color(0xFFA8AFB9),
                    ),
                    border: InputBorder.none,
                  ),
```

```
              ),
            ),
          ],
        ),
        if (_errorMessage.isNotEmpty)
          Padding(
            padding: EdgeInsets.only(left: 48),
            child: Text(
              _errorMessage,
              style: TextStyle(color: Colors.red),
            ),
          ),
      ],
    ),
  );
 }
}

class DropdownField extends StatelessWidget {
 final String labelText;
 final IconData iconData;
 final String? value;
 final Function(String?)? onChanged;
 final List<DropdownMenuItem<String>> items;

 DropdownField({
   required this.labelText,
   required this.iconData,
   this.value,
   this.onChanged,
   required this.items,
 });

 @override
 Widget build(BuildContext context) {
   return Container(
     margin: EdgeInsets.symmetric(vertical: 12, horizontal: 24),
     padding: EdgeInsets.symmetric(horizontal: 16),
     decoration: BoxDecoration(
       color: Colors.white,
       boxShadow: [
         BoxShadow(
           color: Color(0xFF000000).withOpacity(0.25),
           offset: Offset(0, 4),
           blurRadius: 4,
         ),
       ],
       borderRadius: BorderRadius.circular(8),
     ),
     child: Column(
```

```
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              Icon(
                iconData,
                size: 32,
                color: Color(0xFFA8AFB9),
              ),
              SizedBox(width: 16),
              Expanded(
                child: DropdownButtonFormField<String>(
                  value: value,
                  onChanged: onChanged,
                  items: items,
                  decoration: InputDecoration(
                    labelText: labelText,
                    border: InputBorder.none,
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

Terms_and_conditions.dart:

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

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

  @override
  // ignore: library_private_types_in_public_api
  _TermsAndConditionsPageState createState() => _TermsAndConditionsPageState();
}

class _TermsAndConditionsPageState extends State<TermsAndConditionsPage> {
  late ScrollController _scrollController;
  bool _showScrollButton = false;

  @override
  void initState() {
    super.initState();
    _scrollController = ScrollController();
    _scrollController.addListener(_onScroll);
  }

  void _onScroll() {
    if (_scrollController.offset > 200 && !_showScrollButton) {
      setState(() {
        _showScrollButton = true;
      });
    } else if (_scrollController.offset <= 200 && _showScrollButton) {
      setState(() {
        _showScrollButton = false;
      });
    }
  }

  void _scrollToTop() {
    _scrollController.animateTo(
      0,
      duration: const Duration(seconds: 1),
      curve: Curves.easeInOut,
    );
  }

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

  @override
```

```dart
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(),
    body: Stack(
      children: [
        SingleChildScrollView(
          controller: _scrollController,
          padding: const EdgeInsets.symmetric(horizontal: 25, vertical: 16),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // Header Section
              Container(
                width: double.infinity,
                decoration: const ShapeDecoration(
                  color: Colors.white,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.only(
                      topLeft: Radius.circular(32),
                      topRight: Radius.circular(32),
                    ),
                  ),
                ),
                child: const Padding(
                  padding: EdgeInsets.all(20),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
                        'AGREEMENT',
                        style: TextStyle(
                          color: Color(0xFF9F9F9F),
                          fontSize: 16,
                          fontFamily: 'Montserrat',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                      SizedBox(height: 6),
                      Text(
                        'Terms of Service',
                        style: TextStyle(
                          color: Color(0xFF494949),
                          fontSize: 30,
                          fontFamily: 'Montserrat',
                          fontWeight: FontWeight.w700,
                        ),
                      ),
                      SizedBox(height: 6),
                      Text(
                        'Last updated on 5/12/2022',
```

```
                    style: TextStyle(
                      color: Color(0xFF7C7C7C),
                      fontSize: 16,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w600,
                    ),
                  ),
                  Divider(
                    color: Color(0xFFD9D9D9),
                    thickness: 1,
                  ),
                ],
              ),
            ),
          ),

          // Clause 1
          const Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                '1. Acceptance of Terms',
                style: TextStyle(
                  color: Color(0xFF494949),
                  fontSize: 20,
                  fontFamily: 'Lato',
                  fontWeight: FontWeight.w700,
                ),
              ),
              SizedBox(height: 9),
              SizedBox(
                width: 312.69,
                child: Text(
                  'These terms and conditions constitute a binding legal agreement between you and Safe Zone ("Company," "we," "us," or "our"). By accessing or using the Safe Zone app, you agree to abide by these terms.',
                  style: TextStyle(
                    color: Color(0xFF494949),
                    fontSize: 16,
                    fontFamily: 'Lato',
                    fontWeight: FontWeight.w400,
                  ),
                ),
              ),
            ),
          ],
        ),

        const SizedBox(height: 16), // Spacing between clauses

        // Clause 2
```

```
                const Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      '2. App Services',
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 20,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w700,
                      ),
                    ),
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
                        'The Safe Zone app provides a platform for users to book vehicle
services such as routine maintenance, repairs, and customization, as well as reserve secure
parking spaces for short-term or long-term use. The specific services are subject to change
or discontinuation at our discretion.',
                        style: TextStyle(
                          color: Color(0xFF494949),
                          fontSize: 16,
                          fontFamily: 'Lato',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                    ),
                  ],
                ),

                const SizedBox(height: 16), // Spacing between clauses

                // Clause 3
                const Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      '3. User Accounts',
                      style: TextStyle(
                        color: Color.fromRGBO(73, 73, 73, 1),
                        fontSize: 20,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w700,
                      ),
                    ),
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
```

```
                      "To use the app's services, you must create a user account. You agree
to provide accurate and current information during the registration process. You are
responsible for maintaining the confidentiality of your account information, including your
password, and for all activities that occur under your account.",
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 16,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w400,
                      ),
                    ),
                  ),
                ],
              ),
              // Clause 4
              const Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    '4. Service Booking',
                    style: TextStyle(
                      color: Color.fromRGBO(73, 73, 73, 1),
                      fontSize: 20,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w700,
                    ),
                  ),
                  SizedBox(height: 9),
                  SizedBox(
                    width: 312.69,
                    child: Text(
                      "When you book a vehicle service through the app, you agree to pay
the applicable fees. All bookings are subject to availability and confirmation. We reserve
the right to cancel or refuse any booking at our discretion.",
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 16,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w400,
                      ),
                    ),
                  ),
                ],
              ),
              // Clause 5
              const Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    '5. Payment and Refunds',
```

```
                    style: TextStyle(
                      color: Color.fromRGBO(73, 73, 73, 1),
                      fontSize: 20,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w700,
                    ),
                  ),
                  SizedBox(height: 9),
                  SizedBox(
                    width: 312.69,
                    child: Text(
                      "All payments for services must be made through the app using the accepted payment methods. You are responsible for providing accurate payment information. We do not store your payment details. Refunds are subject to our refund policy, which is available on the app or upon request.",
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 16,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w400,
                      ),
                    ),
                  ),
                ],
              ),
              // Clause 6
              const Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    '6. Vehicle Maintenance and Repair',
                    style: TextStyle(
                      color: Color.fromRGBO(73, 73, 73, 1),
                      fontSize: 20,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w700,
                    ),
                  ),
                  SizedBox(height: 9),
                  SizedBox(
                    width: 312.69,
                    child: Text(
                      "We take care to ensure that all services are performed by skilled technicians and mechanics. However, we are not liable for any damage or loss resulting from vehicle maintenance, repairs, or customization unless caused by our gross negligence or intentional misconduct.",
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 16,
                        fontFamily: 'Lato',
```

```
                      fontWeight: FontWeight.w400,
                    ),
                  ),
                ),
              ],
            ),
            // Clause 7
            const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  '7. Parking Services',
                  style: TextStyle(
                    color: Color.fromRGBO(73, 73, 73, 1),
                    fontSize: 20,
                    fontFamily: 'Lato',
                    fontWeight: FontWeight.w700,
                  ),
                ),
                SizedBox(height: 9),
                SizedBox(
                  width: 312.69,
                  child: Text(
                    "Our parking facilities are equipped with advanced security measures, including 24/7 surveillance. However, we do not guarantee against theft, loss, or damage to vehicles or their contents. You are responsible for securing your vehicle and valuables. Our liability for any loss or damage is limited to the value of the parking fee paid.",
                    style: TextStyle(
                      color: Color(0xFF494949),
                      fontSize: 16,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w400,
                    ),
                  ),
                ),
              ],
            ),
            // Clause 8
            const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  '8. Limitation of Liability',
                  style: TextStyle(
                    color: Color.fromRGBO(73, 73, 73, 1),
                    fontSize: 20,
                    fontFamily: 'Lato',
                    fontWeight: FontWeight.w700,
                  ),
                ),
              ),
```

```
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
                        "To the fullest extent permitted by law, the Company shall not be liable for any indirect, incidental, consequential, special, or punitive damages arising out of or related to your use of the app or its services. Our total liability, whether in contract, tort, or otherwise, shall not exceed the amount paid by you for the services.",
                        style: TextStyle(
                          color: Color(0xFF494949),
                          fontSize: 16,
                          fontFamily: 'Lato',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                    ),
                  ],
                ),
                // Clause 9
                const Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      '9. Indemnification',
                      style: TextStyle(
                        color: Color.fromRGBO(73, 73, 73, 1),
                        fontSize: 20,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w700,
                      ),
                    ),
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
                        "You agree to indemnify, defend, and hold the Company, its affiliates, officers, directors, employees, and agents harmless from any claims, damages, liabilities, or expenses arising out of your use of the app, your breach of these terms, or your violation of any applicable laws.",
                        style: TextStyle(
                          color: Color(0xFF494949),
                          fontSize: 16,
                          fontFamily: 'Lato',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                    ),
                  ],
                ),
                // Clause 10
```

```
                const Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      '10. Privacy Policy',
                      style: TextStyle(
                        color: Color.fromRGBO(73, 73, 73, 1),
                        fontSize: 20,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w700,
                      ),
                    ),
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
                        "Your use of the app is subject to our privacy policy, which outlines how we collect, use, and protect your personal information. By using the app, you agree to the terms of the privacy policy.",
                        style: TextStyle(
                          color: Color(0xFF494949),
                          fontSize: 16,
                          fontFamily: 'Lato',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                    ),
                  ],
                ),
                // Clause 11
                const Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      '11. Termination',
                      style: TextStyle(
                        color: Color.fromRGBO(73, 73, 73, 1),
                        fontSize: 20,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w700,
                      ),
                    ),
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
                        "We reserve the right to terminate your account and access to the app at any time, with or without cause. If you breach these terms, we may terminate your account without notice.",
                        style: TextStyle(
```

```
                      color: Color(0xFF494949),
                      fontSize: 16,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w400,
                    ),
                  ),
                ),
              ],
            ),
            // Clause 12
            const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  '12. Governing Law and Dispute Resolution',
                  style: TextStyle(
                    color: Color.fromRGBO(73, 73, 73, 1),
                    fontSize: 20,
                    fontFamily: 'Lato',
                    fontWeight: FontWeight.w700,
                  ),
                ),
                SizedBox(height: 9),
                SizedBox(
                  width: 312.69,
                  child: Text(
                    "These terms shall be governed by and construed in accordance with the laws of the jurisdiction where Safe Zone is based. Any disputes arising out of or related to these terms shall be resolved through arbitration in accordance with the rules of the relevant arbitration body.",
                    style: TextStyle(
                      color: Color(0xFF494949),
                      fontSize: 16,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w400,
                    ),
                  ),
                ),
              ],
            ),
            // Clause 13
            const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  '13. Changes to Terms',
                  style: TextStyle(
                    color: Color.fromRGBO(73, 73, 73, 1),
                    fontSize: 20,
                    fontFamily: 'Lato',
```

```
                      fontWeight: FontWeight.w700,
                    ),
                  ),
                  SizedBox(height: 9),
                  SizedBox(
                    width: 312.69,
                    child: Text(
                      "We may update or modify these terms at any time. We will notify you
of any significant changes, and your continued use of the app after such changes indicates
your acceptance of the revised terms.",
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 16,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w400,
                      ),
                    ),
                  ),
                ],
              ),
              // Clause 14
              const Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    '14. Contact Us',
                    style: TextStyle(
                      color: Color.fromRGBO(73, 73, 73, 1),
                      fontSize: 20,
                      fontFamily: 'Lato',
                      fontWeight: FontWeight.w700,
                    ),
                  ),
                  SizedBox(height: 9),
                  SizedBox(
                    width: 312.69,
                    child: Text(
                      "If you have any questions about these terms, please contact us at
[contact email or address].",
                      style: TextStyle(
                        color: Color(0xFF494949),
                        fontSize: 16,
                        fontFamily: 'Lato',
                        fontWeight: FontWeight.w400,
                      ),
                    ),
                  ),
                ],
              ),
              // Ending
```

```
                const Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    SizedBox(height: 9),
                    SizedBox(
                      width: 312.69,
                      child: Text(
                        "By using the Safe Zone app, you acknowledge that you have read,
understood, and agreed to these terms and conditions. Thank you for choosing Safe Zone.",
                        style: TextStyle(
                          color: Color(0xFF494949),
                          fontSize: 16,
                          fontFamily: 'Lato',
                          fontWeight: FontWeight.w400,
                        ),
                      ),
                    ),
                  ],
                ),

                const SizedBox(height: 16), // Spacing between clauses

                // Accept & Continue Button
                Center(
                  child: Container(
                    width: 183,
                    height: 47,
                    clipBehavior: Clip.antiAlias,
                    decoration: ShapeDecoration(
                      color: const Color(0xFFFF5000),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(100),
                      ),
                      shadows: const [
                        BoxShadow(
                          color: Color(0xFF8E6DE9),
                          blurRadius: 25,
                          offset: Offset(0, 5),
                          spreadRadius: -10,
                        ),
                      ],
                    ),
                    child: Stack(
                      children: [
                        Positioned(
                          left: -162,
                          top: 0,
                          child: Container(
                            width: 162,
                            height: 47,
```

```
                        decoration: const BoxDecoration(
                          color: Colors.white,
                        ),
                      ),
                    ),
                    const Positioned(
                      left: 16,
                      top: 12.5,
                      child: Text(
                        'Accept & Continue',
                        textAlign: TextAlign.center,
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 16,
                          fontFamily: 'Lato',
                          fontWeight: FontWeight.w700,
                        ),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
      if (_showScrollButton)
        Positioned(
          bottom: 16,
          right: 16,
          child: FloatingActionButton(
            onPressed: _scrollToTop,
            child: const Icon(Icons.arrow_upward),
          ),
        ),
    ],
  ),
);
  }
}

void main() {
  runApp(const MaterialApp(
    home: TermsAndConditionsPage(),
  ));
}
```

Users.dart:

```dart
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:get/get.dart';
import 'package:safe_zone/admin_notifications.dart';

void main() {
  runApp(const Users());
}

class Users extends StatelessWidget {
  const Users({Key? key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: 'Users',
      home: UserPage(),
    );
  }
}

class UserPage extends StatelessWidget {
  const UserPage({Key? key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Align(
          alignment: Alignment.topCenter,
          child: Text(
            'Users',
            textAlign: TextAlign.center,
            style: TextStyle(
              color: Color(0xFFF76B1C),
              fontSize: 17,
              fontFamily: 'Roboto',
              fontWeight: FontWeight.w700,
              height: 0,
              letterSpacing: -0.41,
            ),
          ),
        ),
      ),
      body: SingleChildScrollView(
        child: Align(
          alignment: Alignment.topCenter,
```

```
          child: Padding(
            padding: const EdgeInsets.only(top: 10.0, left: 23.5, right: 23.5),
            child: StreamBuilder<QuerySnapshot>(
              stream:
                  FirebaseFirestore.instance.collection('users').snapshots(),
              builder: (context, snapshot) {
                if (!snapshot.hasData) {
                  return const CircularProgressIndicator(); // Loading indicator
                }

                final List<DocumentSnapshot> documents = snapshot.data!.docs;

                final filteredDocuments =
                    documents.where((doc) => !(doc['isAdmin'] ?? false));

                return Column(
                  children: List.generate(
                    filteredDocuments.length,
                    (index) {
                      final userData = filteredDocuments.elementAt(index).data()
                          as Map<String, dynamic>;

                      final String firstName = userData['firstName'];
                      final String secondName = userData['secondName'] ?? '';
                      final String gender = userData['gender'];
                      final String userId =
                          filteredDocuments.elementAt(index).id;

                      return UserBox(
                        userIndex: index + 1,
                        firstName: firstName,
                        secondName: secondName,
                        gender: gender,
                        userId: userId,
                      );
                    },
                  ),
                );
              },
            ),
          ),
        ),
      ),
    );
  }
}

class UserBox extends StatelessWidget {
  final int userIndex;
  final String firstName;
```

```
final String secondName;
final String gender;
final String userId;

const UserBox({
  Key? key,
  required this.userIndex,
  required this.firstName,
  required this.secondName,
  required this.gender,
  required this.userId,
});

@override
Widget build(BuildContext context) {
  final String salutation = gender == 'Male' ? 'Mr. ' : 'Mrs. ';

  return GestureDetector(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => UserCarsPage(userId: userId),
        ),
      );
    },
    child: Padding(
      padding: const EdgeInsets.only(top: 20.0),
      child: SizedBox(
        width: 435,
        height: 185,
        child: Stack(
          children: [
            Container(
              clipBehavior: Clip.antiAlias,
              decoration: BoxDecoration(
                color: Colors.white,
                shape: BoxShape.rectangle,
                borderRadius: BorderRadius.circular(32),
                border: Border.all(width: 2, color: const Color(0xFFF76B1C)),
                boxShadow: const [
                  BoxShadow(
                    color: Color(0x0F000000),
                    blurRadius: 12,
                    offset: Offset(0, 2),
                  ),
                ],
              ),
              child: Padding(
                padding: const EdgeInsets.all(16.0),
```

```
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Row(
                      children: [
                        const Icon(Icons.person_outline,
                            color: Color(0xFFF76B1C)),
                        const SizedBox(width: 8),
                        Text(
                          '$salutation$firstName $secondName',
                          style: const TextStyle(
                            color: Colors.black,
                            fontSize: 15,
                            fontFamily: 'Roboto',
                            fontWeight: FontWeight.w900,
                          ),
                        ),
                      ],
                    ),
                    const SizedBox(height: 6),
                    Container(
                      width: double.infinity,
                      height: 2,
                      color: const Color(0xFFF76B1C),
                    ),
                    const SizedBox(
                      height: 4,
                    ),
                    const SizedBox(height: 8),
                    Padding(
                      padding: const EdgeInsets.only(left: 28),
                      child: RichText(
                        text: TextSpan(
                          children: [
                            TextSpan(
                              text: '${firstName.capitalize!}\'s Profile:\n',
                              style: const TextStyle(
                                color: Colors.black,
                                fontSize: 20,
                                fontFamily: 'Roboto',
                                fontWeight: FontWeight.w700,
                                decoration: TextDecoration.underline,
                                wordSpacing: 2,
                              ),
                            ),
                            const TextSpan(
                              text: '\n',
                              style: TextStyle(
                                fontSize: 10,
                              ),
```

```
                        ),
                        const TextSpan(
                          text: 'Cars',
                          style: TextStyle(
                            color: Color(0xFFF76B1C),
                            fontSize: 20,
                            fontFamily: 'Roboto',
                            fontWeight: FontWeight.w700,
                            decoration: TextDecoration.underline,
                          ),
                        ),
                        const TextSpan(
                          text: '   ',
                          style: TextStyle(
                            color: Colors.black,
                            fontSize: 20,
                            fontFamily: 'Roboto',
                            fontWeight: FontWeight.w400,
                          ),
                        ),
                        TextSpan(
                          text: '\n\nSend Notifications',
                          style: TextStyle(
                            color: Color(0xFFF76B1C),
                            fontSize: 20,
                            fontFamily: 'Roboto',
                            fontWeight: FontWeight.w700,
                            decoration: TextDecoration.underline,
                          ),
                          recognizer: TapGestureRecognizer()
                            ..onTap = () {
                              Navigator.push(
                                context,
                                MaterialPageRoute(
                                  builder: (context) =>
                                      AdminNotificationPage(),
                                ),
                              );
                            },
                        ),
                      ],
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
```

```
        ),
      ),
    );
  }
}

class UserCarsPage extends StatelessWidget {
  final String userId;

  const UserCarsPage({Key? key, required this.userId}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'User Cars',
          style: TextStyle(
            color: Color(0xFFF76B1C),
            fontSize: 17,
            fontFamily: 'Roboto',
            fontWeight: FontWeight.w700,
            letterSpacing: -0.41,
          ),
        ),
      ),
      body: SingleChildScrollView(
        child: Align(
          alignment: Alignment.topCenter,
          child: Padding(
            padding: const EdgeInsets.only(top: 10.0, left: 23.5, right: 23.5),
            child: StreamBuilder<QuerySnapshot>(
              stream: FirebaseFirestore.instance
                  .collection('vehicles')
                  .where('userId', isEqualTo: userId)
                  .snapshots(),
              builder: (context, snapshot) {
                if (!snapshot.hasData) {
                  return const CircularProgressIndicator(); // Loading indicator
                }

                final List<DocumentSnapshot> documents = snapshot.data!.docs;

                if (documents.isEmpty) {
                  return const Text("This user has no cars");
                }

                return Column(
                  children: List.generate(
                    documents.length,
```

```
                  (index) {
                    final vehicleData =
                        documents[index].data() as Map<String, dynamic>;

                    final String vehicleName = vehicleData['brand'];
                    final String vehicleModel = vehicleData['model'];

                    return VehicleBox(
                      vehicleIndex: index + 1,
                      vehicleName: vehicleName,
                      vehicleModel: vehicleModel,
                    );
                  },
                ),
              );
            },
          ),
        ),
      ),
    ),
  );
  }
}

class VehicleBox extends StatelessWidget {
  final int vehicleIndex;
  final String vehicleName;
  final String vehicleModel;

  const VehicleBox({
    Key? key,
    required this.vehicleIndex,
    required this.vehicleName,
    required this.vehicleModel,
  });

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 20.0),
      child: SizedBox(
        width: 435,
        height: 96,
        child: Container(
          clipBehavior: Clip.antiAlias,
          decoration: BoxDecoration(
            color: Colors.white,
            shape: BoxShape.rectangle,
            borderRadius: BorderRadius.circular(32),
            border: Border.all(width: 2, color: const Color(0xFFF76B1C)),
```

```
          boxShadow: const [
            BoxShadow(
              color: Color(0x0F000000),
              blurRadius: 12,
              offset: Offset(0, 2),
            ),
          ],
        ),
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Row(
                children: [
                  const Icon(Icons.directions_car_filled_sharp,
                      color: Color(0xFFF76B1C)),
                  const SizedBox(width: 8),
                  Text(
                    '$vehicleName $vehicleModel',
                    style: const TextStyle(
                      color: Colors.black,
                      fontSize: 15,
                      fontFamily: 'Roboto',
                      fontWeight: FontWeight.w900,
                    ),
                  ),
                ],
              ),
              const SizedBox(height: 6),
              Container(
                width: double.infinity,
                height: 2,
                color: const Color(0xFFF76B1C),
              ),
              const SizedBox(
                height: 4,
              ),
              const SizedBox(height: 8),
              Padding(
                padding: const EdgeInsets.only(left: 28),
                child: RichText(
                  text: const TextSpan(
                    children: [],
                  ),
                ),
              ),
            ],
          ),
        ),
```

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