

NAME : SOHAM SHETYE
DIV : D15A

ROLL NO. : 54
BATCH : C

MAD PWA LAB

LAB 6

Aim: To connect flutter UI with firebase database

THEORY :

Using Firebase with Flutter provides several advantages, making it a popular choice for mobile app development. Here are some reasons why Firebase is often chosen as a backend solution for Flutter applications:

Real-Time Database:

- Firebase provides a real-time NoSQL database that allows seamless data synchronization across devices. This is particularly useful for apps that require live updates, such as messaging apps, collaborative tools, or any application where real-time data is crucial.

Authentication:

- Firebase Authentication simplifies the process of user authentication with various sign-in methods, including email/password, Google Sign-In, Facebook Login, and more. Integrating Firebase Authentication with Flutter is straightforward and provides a secure way to manage user identities.

Cloud Functions:

- Firebase allows you to deploy serverless functions in the cloud using Cloud Functions for Firebase. This enables you to run backend code in response to events triggered by changes in the database, authentication, or other cloud services. These functions can be written in JavaScript, TypeScript, or Dart.

Cloud Storage:

- Firebase offers Cloud Storage for storing and serving user-generated content, such as images, videos, and other files. Integrating Cloud Storage with Flutter enables efficient handling of multimedia content without the need for a separate server.

Firebase Hosting:

- Firebase Hosting provides a simple and secure way to deploy web apps. If you're building a Flutter web application, Firebase Hosting can be used to deploy and serve your app, ensuring a reliable and scalable hosting solution.

Analytics and Crash Reporting:

- Firebase Analytics helps you understand user behavior, track app performance, and gain insights into how users interact with your app. Firebase Crashlytics provides real-time crash reporting, allowing you to identify and address issues quickly.

Cloud Firestore:

- While the real-time database is a key feature, Firebase also offers Cloud Firestore, a more powerful and scalable NoSQL database. Firestore provides richer query capabilities, hierarchical data structures, and better support for large-scale applications compared to the original real-time database.

Easy Integration with Flutter:

- Google provides official plugins for Flutter that make integrating Firebase services seamless. These plugins are well-maintained, well-documented, and designed to work seamlessly with Flutter apps.

Scalability and Reliability:

- Firebase is a fully managed platform, ensuring scalability and reliability without the need for developers to manage servers or infrastructure. This allows developers to focus more on building features and less on managing backend services.

Community Support:

- Due to its popularity, there is a vibrant community around both Flutter and Firebase. This means a wealth of resources, tutorials, and community support are available, making it easier for developers to find help when needed.

CODE :-

```
class LoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login Page'),
      ),
      body: Stack(
        fit: StackFit.expand,
        children: [
          // Background Image
          Image.asset(
            'assets/images/projects.png', // Replace with your image asset path
            fit: BoxFit.cover,
          ),
          Column(
            mainAxisAlignment: MainAxisAlignment.center,
```

```

crossAxisAlignment: CrossAxisAlignment.center,
children: [
  ElevatedButton(
    onPressed: () async {
      // Sign out the current Google user
      await GoogleSignIn().signOut();

      // Open the Google Sign-In page
      await signInWithGoogle(context);
    },
    child: Text('Sign in with Google'),
  ),
  SizedBox(height: 16.0), // Add some spacing
  ElevatedButton(
    onPressed: () {
      // Navigate to the RegisterPage
      Navigator.pushNamed(context, '/register');
    },
    child: Text('Register'),
  ),
],
),
],
),
);
}
}

```

```

Future<void> signInWithGoogle(BuildContext context) async {
  try {
    final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();
    if (googleUser == null) {
      // User canceled the sign-in
      return;
    }
  }
}

```

```

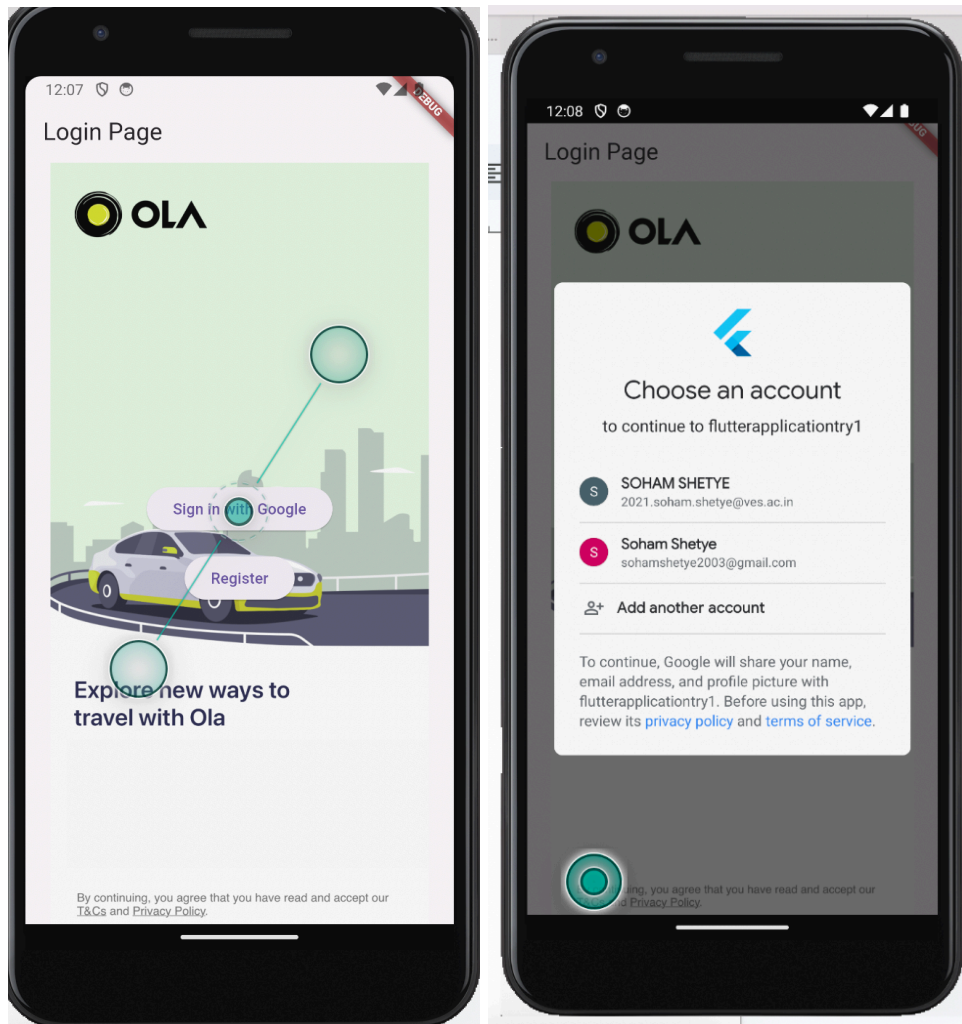
final GoogleSignInAuthentication googleAuth =
  await googleUser.authentication;
final AuthCredential credential = GoogleAuthProvider.credential(
  accessToken: googleAuth.accessToken,
  idToken: googleAuth.idToken,
);
await FirebaseAuth.instance.signInWithCredential(credential);

```

```


// Navigate to the home page after successful sign-in
Navigator.pushReplacementNamed(context, '/home');
} catch (e) {
  print('Error signing in with Google: $e');
}
}

```



Search by email address, phone number, or user UID

Add user

Identifier	Providers	Created <div>↓</div>	Signed In	User UID
sohamshetye2003@gm...		Feb 14, 2024	Feb 14, 2024	magShpgQM6Ph6M3kK0TCO...

Rows per page:

50

▼

1 – 1 of 1

◀

▶

```
final String pickupSpot;  
final String destination;
```

```
DriverListPage({required this.pickupSpot, required this.destination});
```

```
@override  
_DriverListPageState createState() => _DriverListPageState();  
}
```

```
class _DriverListPageState extends State<DriverListPage> {  
  int? selectedDriverIndex;  
  List<TaxiDriver> taxiDrivers = [];
```

```
@override  
void initState() {  
  super.initState();  
  // Fetch data from Firestore when the widget is initialized  
  fetchData();  
}
```

```
Future<void> fetchData() async {  
  try {  
    QuerySnapshot<Map<String, dynamic>> querySnapshot =  
      await FirebaseFirestore.instance.collection('taxiDrivers').get();  
  
    setState(() {  
      taxiDrivers = querySnapshot.docs  
        .map((doc) => TaxiDriver.fromMap(doc.data() as Map<String, dynamic>))  
        .toList();  
    });  
  } catch (e) {  
    print('Error fetching data: $e');  
  }  
}
```

```
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(  
      title: Text('Taxi Drivers'),  
      backgroundColor: Colors.green,  
    ),  
    body: Column(  
      children: [
```

```

Container(
  height: MediaQuery.of(context).size.height * 0.4,
  decoration: BoxDecoration(
    image: DecorationImage(
      image: AssetImage("assets/images/map.jpg"),
      fit: BoxFit.cover,
    ),
  ),
),
Expanded(
  child: taxiDrivers.isEmpty
    ? Center(
        child: CircularProgressIndicator(),
      )
    : ListView.builder(
        itemCount: taxiDrivers.length,
        itemBuilder: (context, index) {
          return GestureDetector(
            onTap: () {
              setState(() {
                selectedDriverIndex = index;
              });
            },
            child: Card(
              elevation: 4.0,
              margin: EdgeInsets.symmetric(
                vertical: 8.0, horizontal: 16.0),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(12.0),
                side: BorderSide(
                  color: selectedDriverIndex == index
                    ? Colors.green
                    : Colors.transparent,
                  width: 2.0,
                ),
              ),
            ),
            child: ListTile(
              title: Text(
                taxiDrivers[index].name,
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Colors.black,
                ),
              ),
            ),
          ),
        ),
      ),
    ),
  ),

```

```

        subtitle: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Taxi: ${taxiDrivers[index].taxiName}',
              style: TextStyle(
                color: Colors.green,
              ),
            ),
            Text(
              'Rate: ${taxiDrivers[index].rate}',
              style: TextStyle(
                color: Colors.green,
              ),
            ),
          ],
        ),
        leading: CircleAvatar(
          backgroundImage:
            AssetImage(taxiDrivers[index].carPhoto),
          backgroundColor: Colors.white,
        ),
      ),
    );
  },
),
ElevatedButton(
  onPressed: selectedDriverIndex != null
    ? () {
        // Implement booking logic here
        // You can use taxiDrivers[selectedDriverIndex] to get the selected driver details
        // For example: taxiDrivers[selectedDriverIndex].name
        showDialog(
          context: context,
          builder: (context) => AlertDialog(
            title: Text('Booking Confirmation'),
            content: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              mainAxisAlignment: MainAxisAlignment.min,
              children: [
                Image.asset(
                  taxiDrivers[selectedDriverIndex!].carPhoto,

```

```

        height: 100,
        width: 100,
        fit: BoxFit.cover,
      ),
      SizedBox(height: 8),
      Text(
        'Driver: ${taxiDrivers[selectedDriverIndex!].name}',
      ),
      Text(
        'Taxi: ${taxiDrivers[selectedDriverIndex!].taxiName}',
      ),
      Text(
        'Rate: ${taxiDrivers[selectedDriverIndex!].rate}',
      ),
    ],
  ),
  actions: [
    TextButton(
      onPressed: () {
        Navigator.pop(context);
      },
      child: Text('OK'),
    ),
  ],
),
);
}
: null,
style: ElevatedButton.styleFrom(
  primary: Colors.green,
),
child: Text(
  'Book Now',
  style: TextStyle(color: Colors.white),
),
),
],
),
);
}
}

```

```

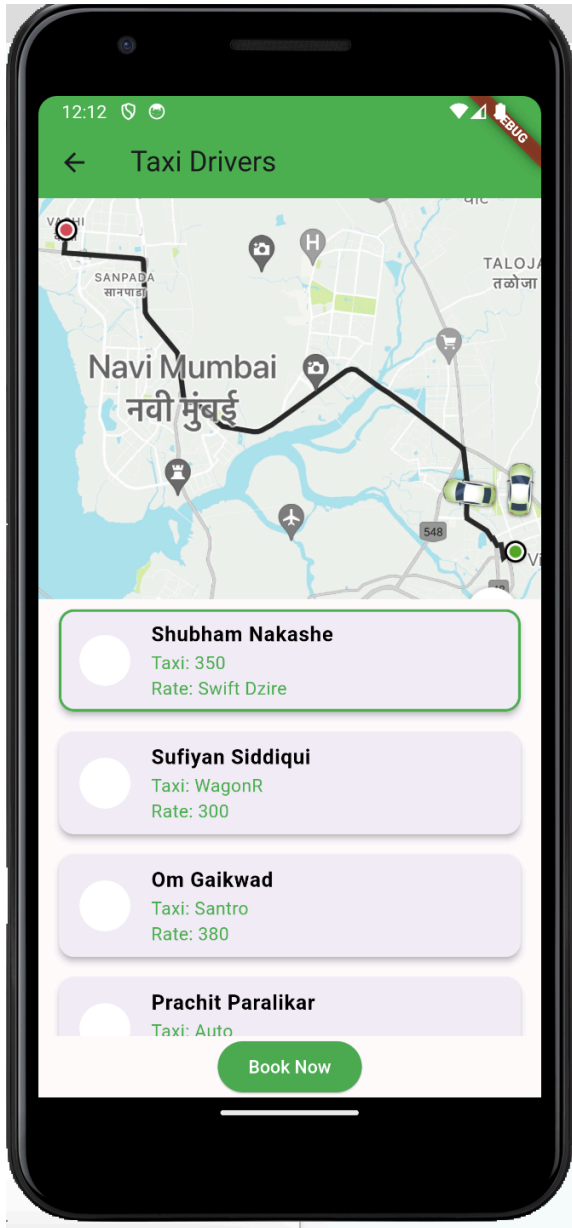
class TaxiDriver {
  final String name;
  final String taxiName;
  final String rate;
  final String carPhoto;
}

```



```
TaxiDriver({  
  required this.name,  
  required this.taxiName,  
  required this.rate,  
  required this.carPhoto,  
});
```

```
factory TaxiDriver.fromMap(Map<String, dynamic> map) {  
  print('Mapping Firestore data: $map');  
  return TaxiDriver(  
    name: map['name'] ?? "",  
    taxiName: map['taxiName'] ?? "",  
    rate: map['rate'] ?? "",  
    carPhoto: map['carPhoto'] ?? "",  
  );  
}  
}
```



🏠 > taxiDrivers > BlmTRGyocD3x...		
☰ (default)	📁 taxiDrivers	📄 BlmTRGyocD3x6xX5FbIY
+ Start collection	+ Add document	+ Start collection
taxiDrivers >	3XWxWNKley2A0Y1120qP 7PFS8BpqAEqE8DzyZIOS 8jmGR10ApHiyGE8GgzNH ⋮ BlmTRGyocD3x6xX5FbIY > xj9zbDKQdRu10Zcz60Fq	+ Add field carPhoto: <i>null</i> name: "Prachit Paralikar" rate: "200" taxiName: "Auto"

CONCLUSION :-

Firebase provides a comprehensive set of backend services that complement Flutter's capabilities, making it a convenient and powerful choice for building mobile and web applications with real-time features, authentication, and more.