Experiment No.6

- **Aim:** To connect flutter UI with firebase database.

- **Theory:**

Firebase:

Firebase is a comprehensive mobile and web development platform provided by Google. It offers a set of tools and services to streamline various aspects of app development, including real-time databases, authentication, cloud functions, hosting, and more. One of its key features is the Firebase Realtime Database, a NoSQL cloud-based database that allows developers to store and sync data in real-time.

Firebase Use:

1. **Realtime Database:** Firebase provides a scalable and real-time cloud database, allowing developers to build responsive applications with synchronized data across clients.
2. **Authentication:** Firebase simplifies user authentication, offering ready-to-use authentication services, such as email/password, Google Sign-In, and more.
3. **Cloud Functions:** Serverless computing with Firebase Cloud Functions enables developers to run backend code in response to events triggered by Firebase features.
4. **Cloud Firestore:** Firestore is another Firebase database option, offering more advanced querying capabilities and scalability.
5. **Hosting:** Firebase Hosting allows developers to deploy and host web applications, ensuring fast and secure content delivery to users.

Steps to use firebase in flutter:

1. Create a project in the Firebase Console.
2. Register your app in the Firebase project.
3. Download and add configuration files to your Flutter project.
4. Add Firebase dependencies (e.g., firebase_core, firebase_auth) in pubspec.yaml.
5. Initialize Firebase in the main.dart file.
6. Access Firebase services in your Flutter code.
7. Add authentication services if needed.
8. Run and test your Flutter app.

- **Code:-**

Integration in Flutter:

To integrate Firebase with a Flutter app:

1. **Add Firebase to Your Project:**

Create a Firebase project on the Firebase Console (https://console.firebase.google.com/). Register your app and download the google-services.json (Android) or GoogleService-Info.plist (iOS) configuration files.

2. **Flutter Firebase Plugins:**

Add the necessary Flutter Firebase plugins to your pubspec.yaml file, such as firebase_core, cloud_firestore for Firestore, or firebase_database for the Realtime Database.

```
pubspec.yaml
29    dependencies:
30      flutter:
32        firebase_core: ^1.10.6
33        cloud_firestore: ^3.4.1
34
```

3. **Initialize Firebase:**

Initialize Firebase in your Flutter app using Firebase.initializeApp() in the main.dart file.

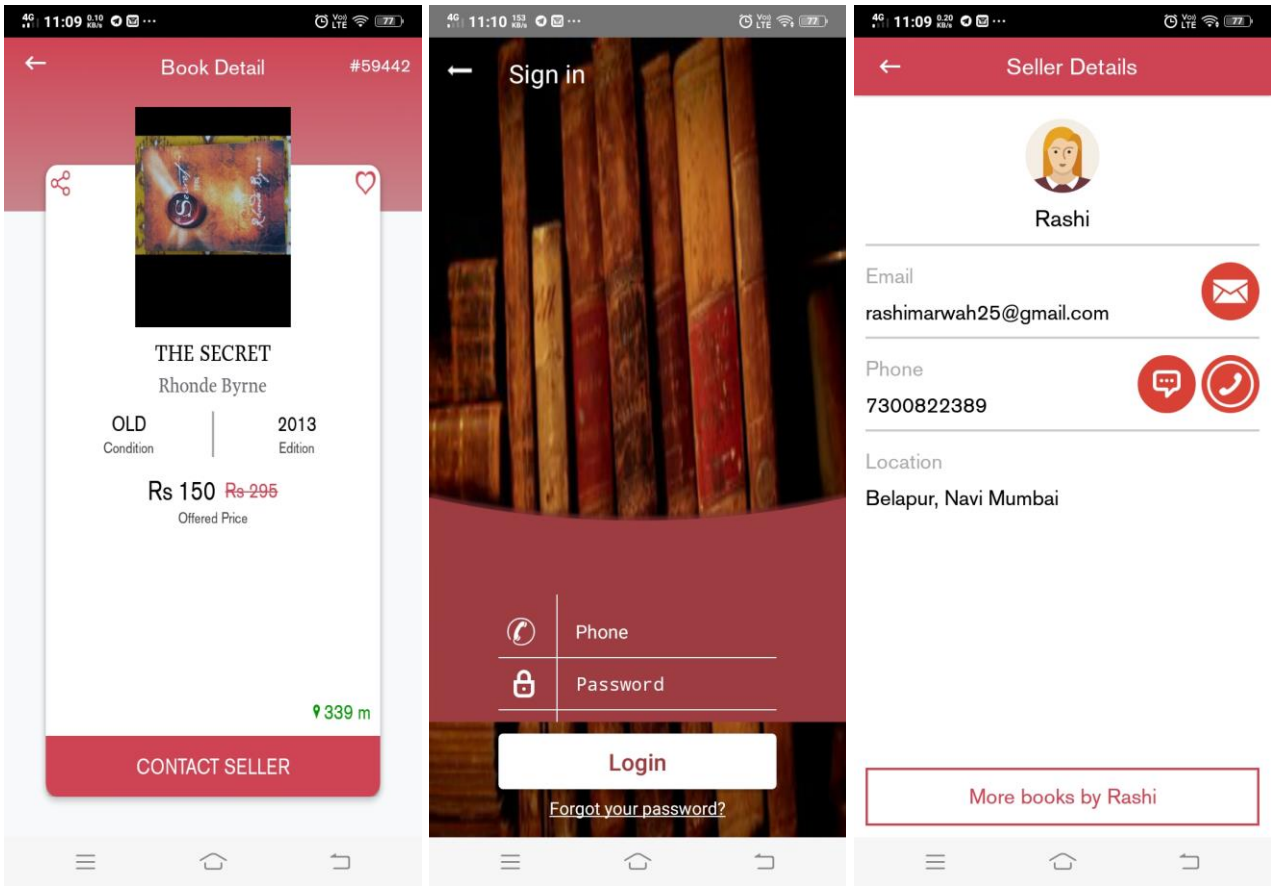
```
main.dart

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: FirebaseOptions(
      apiKey: "AIzaSyDcvOdEQ[illegible]mx4Rg8jvgCDm27EE",
      appId: "1:310036739787[illegible]54db875786c6c8bd68d4e",
      projectId: "signin-fe6e2",
      messagingSenderId: "1234567890", // Provide a dummy value
    ),
  );
  runApp(MyApp());
}
```

Loading...

4. **Database Operations:**
   Use Flutter Firebase plugins to perform database operations. For example, with Firestore:

```
class _SignInScreenState extends State<SignInScreen> {
  Widget build(BuildContext context) {
            padding: const EdgeInsets.all(10.0),
            child: ElevatedButton(
              onPressed: () async {
                Map res = await FirebaseAuthHelper.firebaseAuthHelper
                    .SignInAnonymously();
                if (res['user'] != null) {
                  person = res['email'];
                  ScaffoldMessenger.of(context).showSnackBar(
                    const SnackBar(
                      behavior: SnackBarBehavior.floating,
                      backgroundColor: Colors.green,
                      content: Text(
                        'Successfully Sign in as a guest',
                        style: TextStyle(
                          color: Colors.white,
                          fontWeight: FontWeight.bold,
                        ), // TextStyle
                      ), // Text
                    ), // SnackBar
                  );
```

- **Output:-**



- **Conclusion:**

Integrating Flutter UI with Firebase database in my book-selling app has transformed user interactions. By incorporating Firebase services, I efficiently manage user authentication and securely store data. This hands-on experience has deepened my understanding of connecting Flutter apps to dynamic backend systems, enabling seamless data exchange and enhancing the overall functionality and responsiveness of my book-selling platform.