MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	37
Name	Nihal Nagdev
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

Aim:

To connect a Flutter application to Firebase by integrating Firebase Core and initializing Firebase services in a Flutter project.

Theory:

Firebase offers a suite of backend services such as Authentication, Firestore, Realtime Database, Storage, and more to aid in mobile and web app development. To use these services in Flutter, the firebase_core package is essential. It acts as a bridge between your Flutter app and Firebase.

This experiment focuses on:

1. Setting up Firebase for your Flutter project.
2. Integrating firebase_core into a Flutter application.
3. Configuring platform-specific settings for Android and iOS.
4. Initializing Firebase to enable other services.

Steps to Connect Firebase with Flutter:

1. Firebase Console Setup:

- Go to Firebase Console.
- Click "Create a Project" and follow the setup instructions.
- Enable required Firebase services (e.g., Firestore, Authentication, Storage).

2. Add Your Flutter App to Firebase:

For Android:

1. In Firebase Console → Click Add App → Choose Android.
2. Enter your app's Android package name (check android/app/src/main/AndroidManifest.xml).
3. Download google-services.json.
4. Place the file inside android/app/.
5. Modify Android build files:

In android/build.gradle:

```
    dependencies {
  classpath 'com.google.gms:google-services:4.3.3' // Add this
}
```

In android/app/build.gradle (bottom of file):

```
apply plugin: 'com.google.gms.google-services' // Add this
```

For iOS:

1. In Firebase Console → Add App → Choose iOS.
2. Enter your iOS Bundle ID.
3. Download GoogleService-Info.plist.
4. Open the project in Xcode and drag the .plist file into the Runner project. Check "Copy items if needed".

In ios/Podfile, ensure:

```
platform :ios, '10.0'
```

3. Add Firebase Dependencies in Flutter:

In pubspec.yaml:

```
dependencies:
flutter:
   sdk: flutter
firebase_core: ^1.10.0 Run:
flutter pub get
```

4. Initialize Firebase in Flutter:

```
In main.dart: import
'package:flutter/material.dart'; import
'package:firebase_core/firebase_core.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {     return MaterialApp(
title: 'Flutter Firebase Connection',       theme:
ThemeData(primarySwatch: Colors.blue),       home: Scaffold(
appBar: AppBar(title: Text('Firebase Initialized')),        body:
Center(child: Text('Firebase Connected Successfully!')),
      ),
  );
 } }
```

5. Platform-Specific Configuration:

Android:

In AndroidManifest.xml, add inside <application>:

```
<meta-data
android:name="com.google.firebase.messaging.default_notification_icon"
android:resource="@drawable/ic_notification" />
```

iOS:

In Info.plist, add:

```
<key>UIBackgroundModes</key>
<array>
  <string>fetch</string>
  <string>remote-notification</string>
</array>
<key>NSLocationWhenInUseUsageDescription</key>
<string>Your app requires access to location</string>
```

6. Test Firebase Connection:

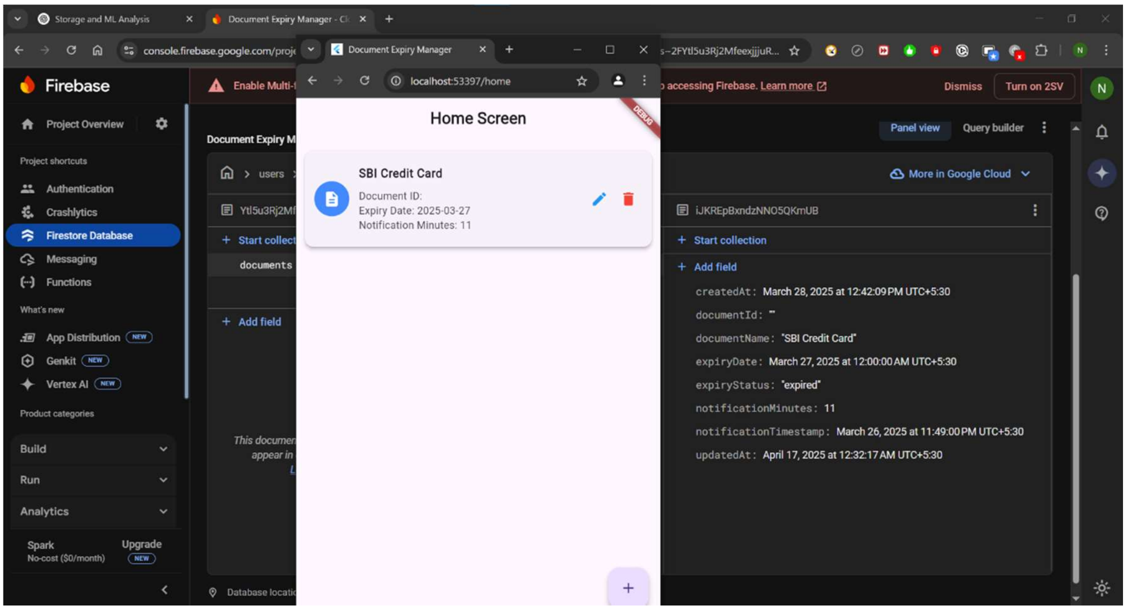
- Run your app on a physical device or emulator.
- If everything is configured correctly, the app should display:

"Firebase Connected Successfully!"

7. Optional: Add Other Firebase Services

- Firestore:
  Add cloud_firestore package for database operations.
- Authentication:
  Use firebase_auth for email/password or social sign-ins.
- Storage:
  Add firebase_storage to handle file uploads and downloads.

Output:



Home Screen

SBI Credit Card
Document ID:
Expiry Date: 2025-03-27
Notification Minutes: 11

```
createdAt: March 28, 2025 at 12:42:09 PM UTC+5:30
documentId: ""
documentName: "SBI Credit Card"
expiryDate: March 27, 2025 at 12:00:00 AM UTC+5:30
expiryStatus: "expired"
notificationMinutes: 11
notificationTimestamp: March 26, 2025 at 11:49:00 PM UTC+5:30
updatedAt: April 17, 2025 at 12:32:17 AM UTC+5:30
```

Conclusion:

After following the above steps, your Flutter application is successfully connected to Firebase. The Firebase Core integration is the foundation that allows you to utilize Firebase services such as Firestore, Authentication, Cloud Storage, etc., to enhance your app functionality.