

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

Theory:Firebase provides several backend services like Authentication, Firestore, Realtime Database, Storage, etc., to build mobile and web apps. Firebase Core is the essential service that allows Flutter apps to communicate with Firebase. Once Firebase Core is connected, you can add specific Firebase services like Firestore, Firebase Authentication, Firebase Storage, etc.

This experiment focuses on:

1. Setting up Firebase for your project.
2. Integrating Firebase Core with a Flutter app.
3. Platform-specific configurations (Android and iOS).
4. Basic setup for Firebase services.

Steps to Connect Firebase with Flutter:

1. Firebase Console Setup:

To begin with, you'll need a Firebase project.

Create Firebase Project:

1. Go to Firebase Console.
2. Click on Create a New Project and follow the steps.
3. After creating the project, you'll be directed to the Firebase console for your project.
4. Enable the Firebase services you need (e.g., Firestore, Firebase Authentication, Firebase Storage).

2. Add Your Flutter App to Firebase:

For Android:

1. Register Your Android App:
 - In the Firebase Console, click Add app and choose Android.
 - Register your app with your Android package name (you can find it in `android/app/src/main/AndroidManifest.xml`).
 - Download the `google-services.json` file.
2. Place `google-services.json` in the Android directory:
 - Put the downloaded `google-services.json` file in the `android/app/` directory.
3. Configure Android build files:

In `android/build.gradle`, add the following classpath inside the dependencies block:

`classpath 'com.google.gms:google-services:4.3.3'` // Add this line

Then, in the `android/app/build.gradle` file, add the following line at the bottom of the file:

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

For iOS:

1. Register Your iOS App:

- In the Firebase Console, click Add app and choose iOS.
- Register the iOS app with your iOS bundle ID.
- Download the GoogleService-Info.plist file.

2. Add GoogleService-Info.plist to Xcode:

- Open your Flutter project in Xcode.
- Drag the GoogleService-Info.plist into the Runner project inside Xcode.
- Make sure to select Copy items if needed and add it to your app target.

3. Add Firebase SDK in iOS:

In the ios/Podfile, ensure you have the following lines:

```
platform :ios, '10.0' # Firebase SDK requires at least iOS 10
```

Run the following command to install the CocoaPods dependencies:

```
cd ios
pod install
cd ..
```

3. Add Firebase Dependencies in Flutter:

In your pubspec.yaml file, include the following dependencies to initialize Firebase Core dependencies:

```
flutter:
  sdk: flutter
  firebase_core: ^1.10.0 # Firebase Core
```

Run the following command to install the dependencies:

```
flutter pub get
```

4. Initialize Firebase in Flutter:

In the main.dart file, you'll need to initialize Firebase before you use any Firebase service.

Add the following code:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart'; // Import Firebase Core

void main() async {
  WidgetsFlutterBinding.ensureInitialized(); // Ensures Firebase is initialized before app
  starts await Firebase.initializeApp(); // Initialize Firebase
  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. Handle Platform-Specific Setup (for Android and iOS): For Android:

Ensure the following is in your android/app/src/main/AndroidManifest.xml to allow Firebase services:

```

<application
  android:label="your-app-name"
  android:icon="@mipmap/ic_launcher">
  <!-- Add this line for Firebase services --> <meta-data
    android:name="com.google.firebase.messaging.default_notification_icon"
    android:resource="@drawable/ic_notification" /> </application>

```

For iOS:

You may need to request permissions for notifications or other Firebase services in your Info.plist file (if using Firebase Messaging, for example).

```

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

```

6. Testing Firebase Connection:

Run the app on an emulator or physical device. If everything is set up correctly, the app should launch with the message: "Firebase Connected Successfully!"

7. Optional: Firebase Services Integration (Firestore, Firebase Auth, etc.)

After successfully connecting Firebase, you can easily integrate other Firebase services, such as:

1. Firebase Firestore:

- Add the `cloud_firestore` package in `pubspec.yaml` and perform operations (CRUD).

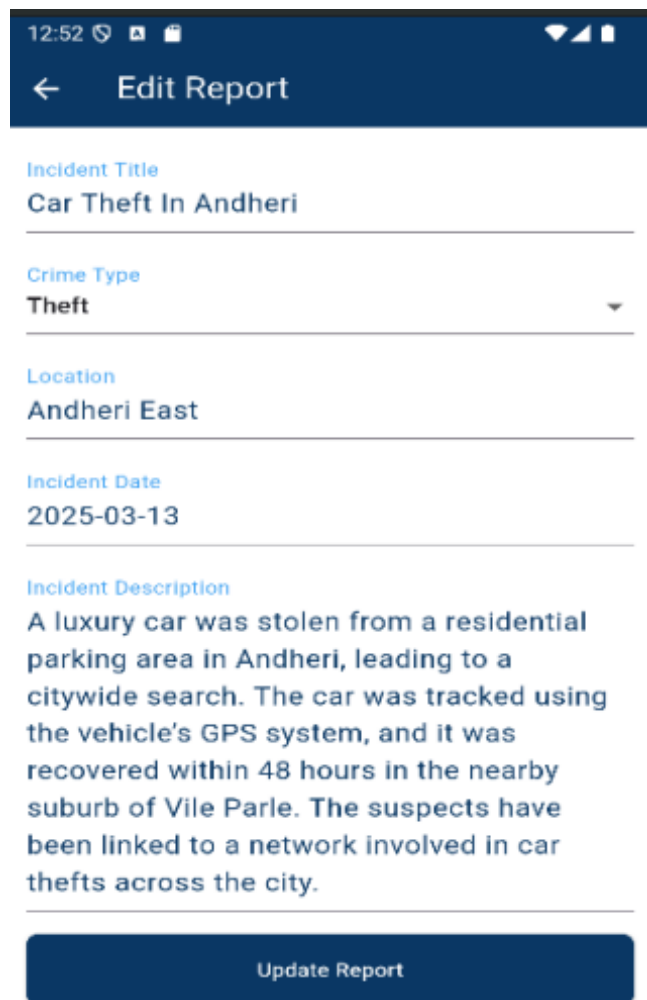
2. Firebase Authentication:

- Add the `firebase_auth` package and enable authentication methods like email/password or Google sign-in.

3. Firebase Cloud Storage:

- Use the `firebase_storage` package for storing and retrieving files.

Output:



12:52

← Edit Report

Incident Title

Car Theft In Andheri

Crime Type

Theft

Location

Andheri East

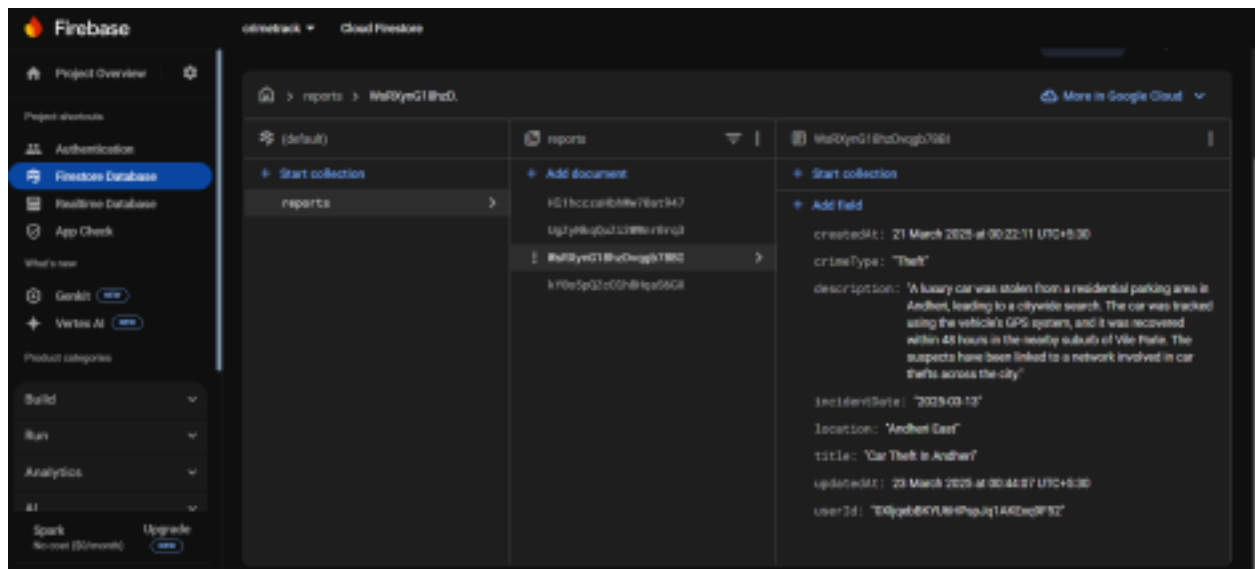
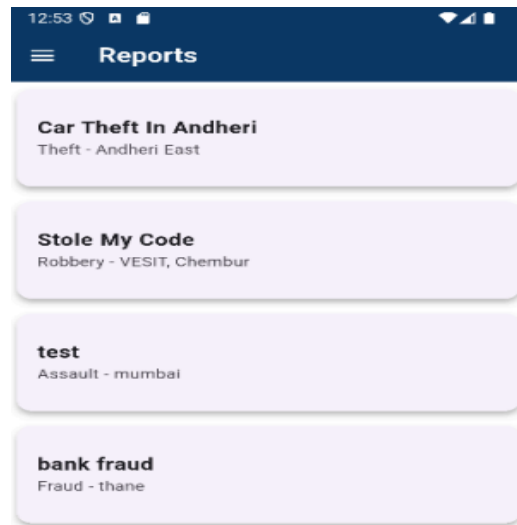
Incident Date

2025-03-13

Incident Description

A luxury car was stolen from a residential parking area in Andheri, leading to a citywide search. The car was tracked using the vehicle's GPS system, and it was recovered within 48 hours in the nearby suburb of Vile Parle. The suspects have been linked to a network involved in car thefts across the city.

Update Report



Conclusion:

After completing the above steps, your Flutter app is successfully connected to Firebase. The Firebase Core initialization allows your Flutter app to communicate with Firebase services. This setup can be expanded to include services like Firestore, Firebase Auth, Cloud Storage, and more based on the app's requirements.