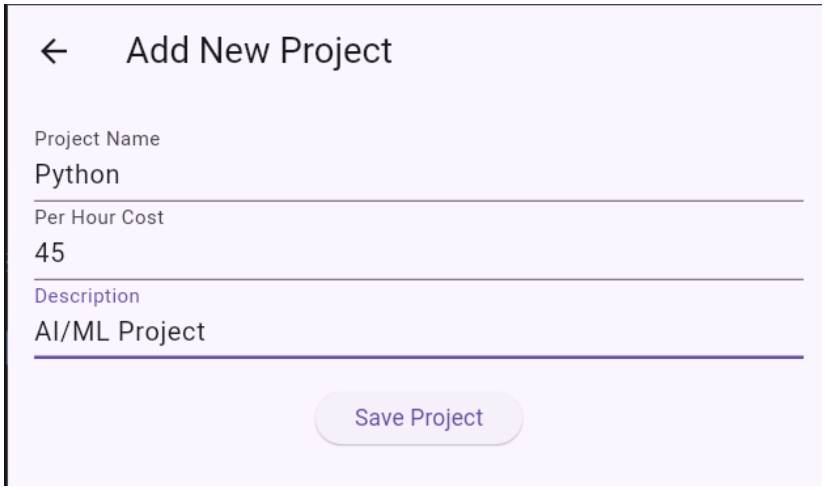
- Aim:To connect a Flutter application to Firebase by integrating Firebase Core, and initializing Firebase services in a Flutter project.
- Theory:Firebase provides several backend services like Authentication, Firestore, Realtime Database, Storage, etc., to build mobile and web apps. Firebase Core is 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:
        - Setting up Firebase for your project.
        - Integrating Firebase Core with a Flutter app.
        - Platform-specific configurations (Android and iOS).
        - Basic setup for Firebase services.
    - Steps to Connect Firebase with Flutter:
        - Go to Firebase Console.
        - Click on Create a New Project and follow the steps.
        - After creating the project, you'll be directed to the Firebase console for your project.
        - Enable the Firebase services you need (e.g., Firestore, Firebase Authentication, Firebase Storage).
        - Add Your Flutter App to Firebase: For Android:
        - . 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.
        - . Place google-services.json in the Android directory
        - Put the downloaded google-services.json file in the android/app/ directory
        - 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:
            - 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.
            - 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.

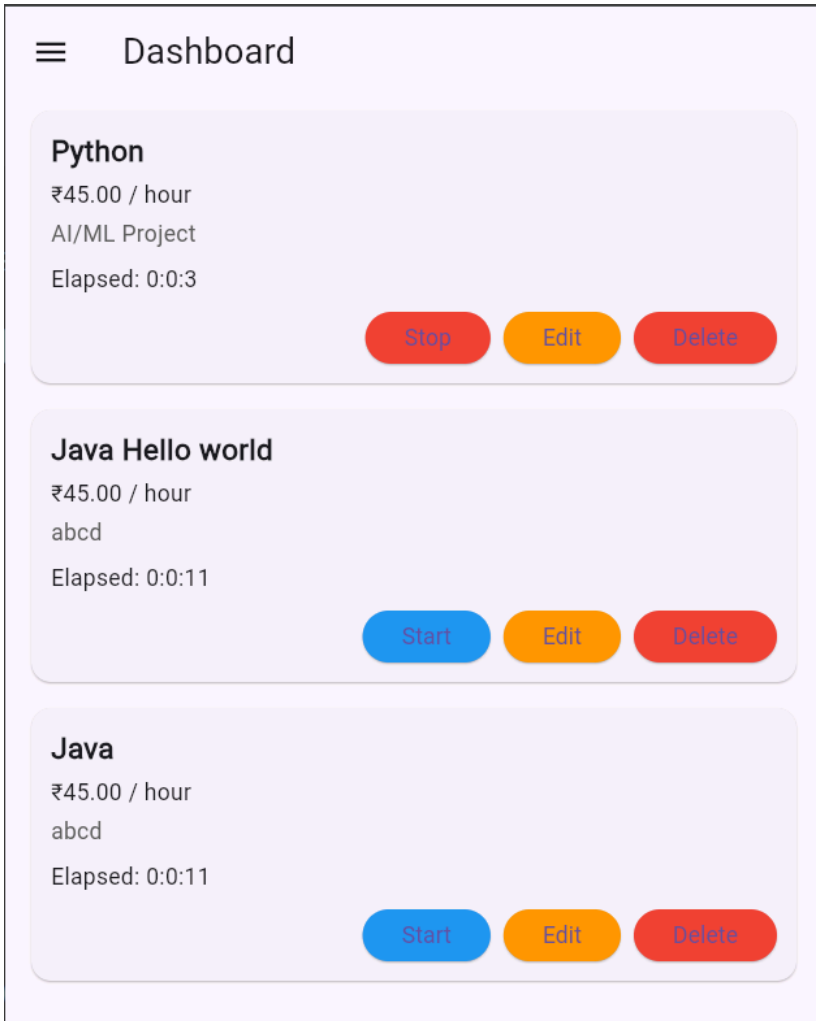
- .Add Firebase SDK in iOS: In the ios/Podfile, ensure you have the following lines: platform :ios, '10.0' # Firebase SDK requires at least iOS 10 Run the following command to install the CocoaPods dependencies:
- cd ios pod install
- cd ..
- Add Firebase Dependencies in Flutter: In your pubspec.yaml 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

- 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!'),
),
),
);
}
}
```

- Output :





- 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.