

Documentation for GoRide

The online cab booking service providers care the price of traveling based on the distance of travel trip and type of car, traffic, and waiting prices. There are numerous apps available in the android play store and apple app store for cab booking in India.

Selecting the greatest taxi booking apps in India is tough, which may vary from town to town.

Basic knowledge required to setup

1. Real server Server related knowledge like apache or local machine server, we preferred to use a real server
2. Server related knowledge and we preferred cPanel in your server for quick installation
3. Basic knowledge in PHP, Laravel and Flutter if you want to do some customization yourself (Not compulsory).
4. Basic knowledge about google cloud and firebase

Server Requirement


1. Laravel 10.x requires a minimum PHP version of 8.1
2. Support for MySQL v5.7
3. Apache Server (Recommended)
4. Mod_rewrite Apache
5. PDO Extension and other required modules of PHP
6. Firebase Account
7. Firebase Database
8. Google MAP API Key

Admin Panel Setup

Step 1: After downloading the source code unzip **goride_source_code.zip** and upload the **goride_admin_panel.zip** file and extract to the root directory of your server.

For example: `/var/www/html/` or `/home/yourusername/public_html` or whatever is the root folder of your domain or subdomain. Which will make it reachable as follows: <http://yourdomain.com/>

Step 2: Create a new database from your server MYSQL database



The image shows the cPanel interface for MySQL Databases. At the top, the cPanel logo is visible. Below it, the heading 'MySQL® Databases' is displayed. A descriptive paragraph follows: 'Manage large amounts of information over the web easily. MySQL databases are necessary to run many web-based applications like shopping carts. For more information, read the [documentation](#).' Below this, the section 'Create New Database' is shown. It includes a label 'New Database:' and a text input field containing 'sisappdev_'. A red rectangular box highlights the 'Create Database' button, which is a blue button with white text.

Step 3: Create a DB user to the database and link that database to the DB user
Note: Give all permission to your user by Check on “All PRIVILEGES”

cPanel Search

MySQL Users

Add New User

Username

Password

Password (Again)

Strength Very Weak (0/100) Password Generator

Create User

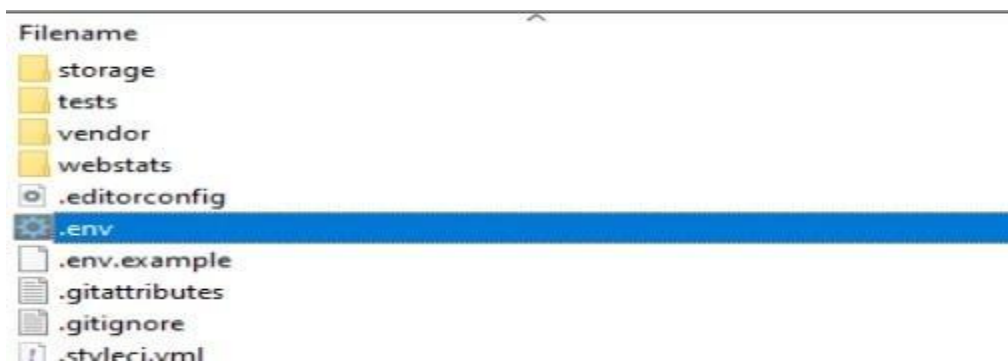
Add User To Database

User

Database

Add

Step 4: Update your database credentials (DB_DATABASE, DB_USERNAME &DB_PASSWORD) to .env file which exist at root path of admin panel



Step 5: Click on the Import Database button and import the admin database `goride_admin_database.sql` file.

Importing into the database "dbqqp7rv0jghni"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in .[format].[compression]. Example: .sql.zip

Browse your computer: No file chosen (Max: 128MiB)

You may also drag and drop a file on any page.

Character set of the file:

Partial import:

☒

Step 6: Update your APP_NAME & APP_URL In .env file

Now you can access your admin panel with your desired domain.

Login with default admin credentials

Username: admin@goride.com

Password: 12345678

Landing Page Setup

Step 1: After downloading the source code unzip **goride_source_code.zip** and upload the **goride_landing_panel.zip** file and extract to the root directory of your server.

Like: <http://landing.yourdomain.com/> or whatever which you like.

Please use subdomain of your main domain for landing page setup
(Recommended)

Step 2: Update firebase credentials at .env file of landing page site.

Notes: Please use this same firebase credentials for landing page setup which you will add for admin panel setup on .env file.

Now you can access your landing page at your desired domain

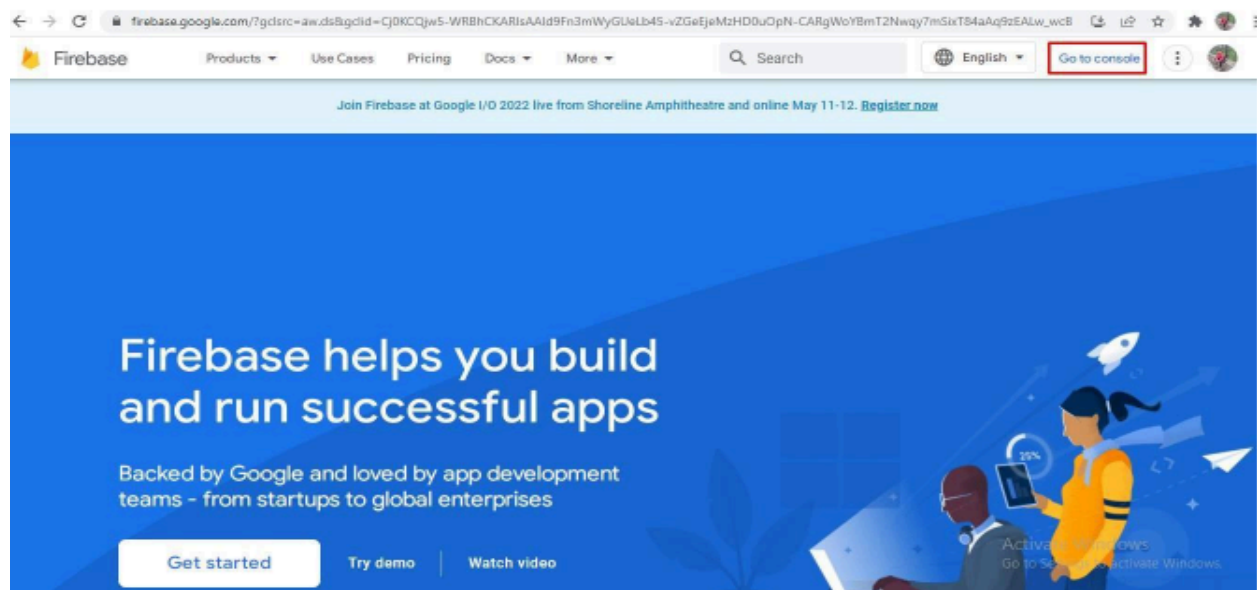
For ex: <http://landing.yourdomain.com/>

Firebase Project Setup/Configuration

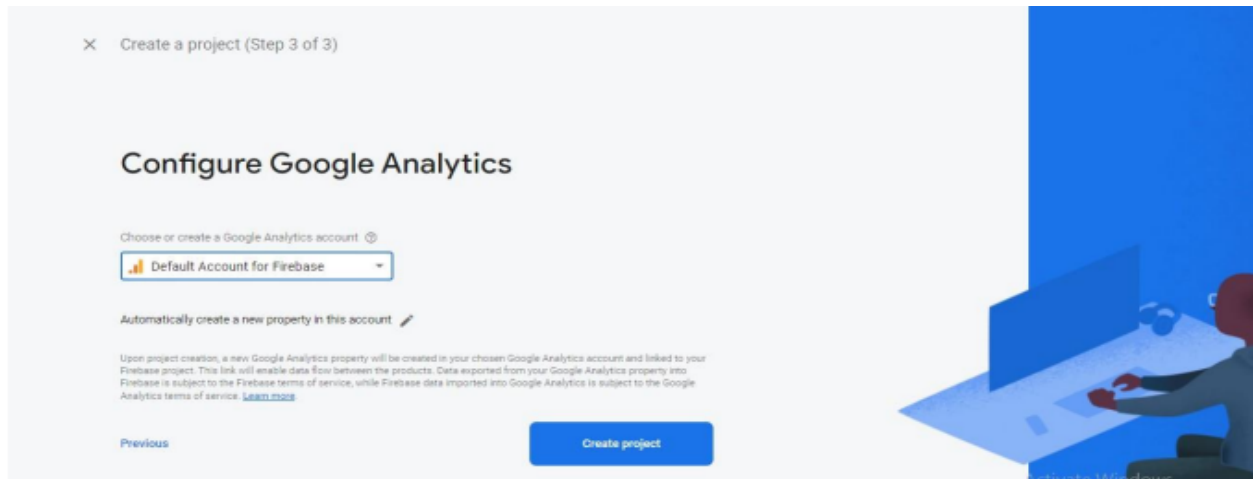
Create Firebase project

Step 1: Go to the firebase console using this link <https://firebase.google.com/>

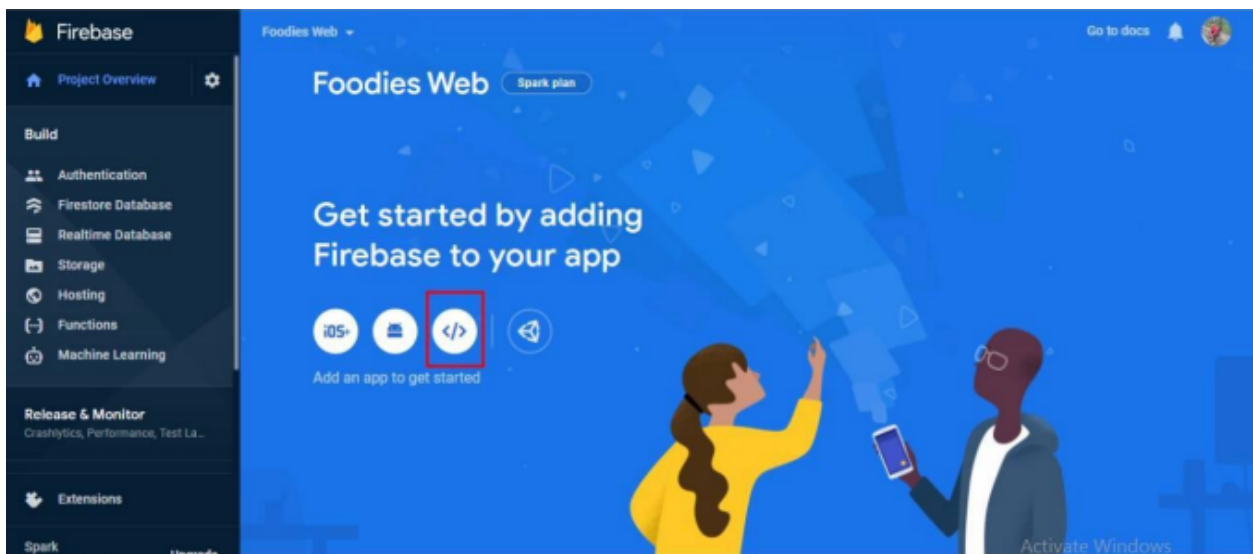
Step 2: Click on “Go to console” in the top right corner.



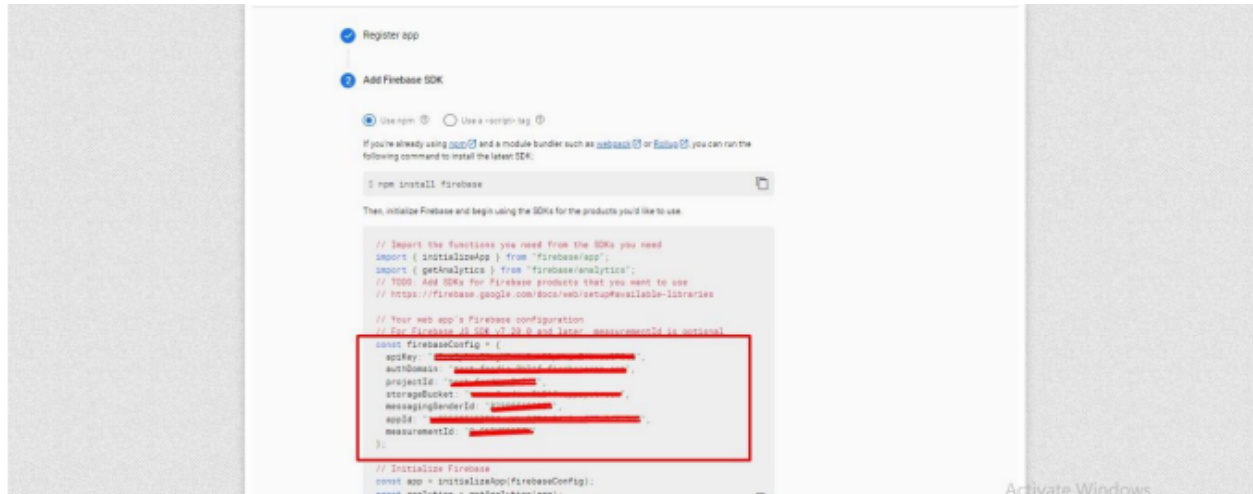
Step 3: Click on “Add project”, it will redirect you to the new project creation page Enter your project name and click on “Continue” again click on “Continue” after that select “Default Account fo firebase” and then click on “Create Project”



Step 4: After creating a new project it will redirect you to the overview page on this page click on icons like

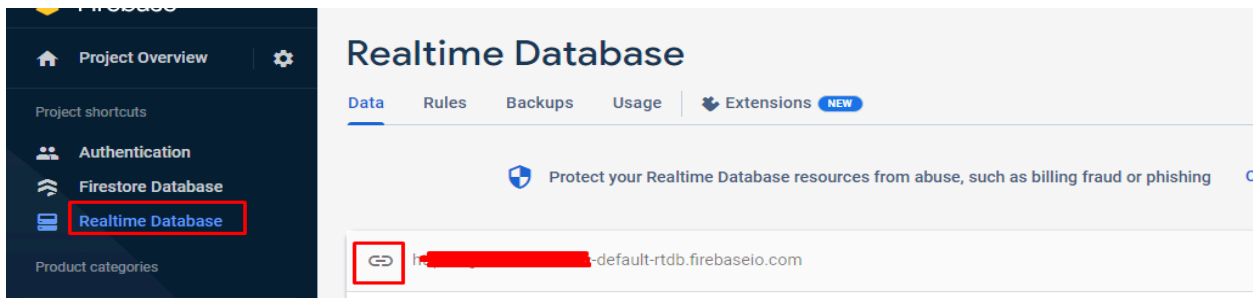


It will redirect you to the “Add Firebase to your web app” page enter your app nickname and click on “Register app” after clicking on it scroll down you can see details like below

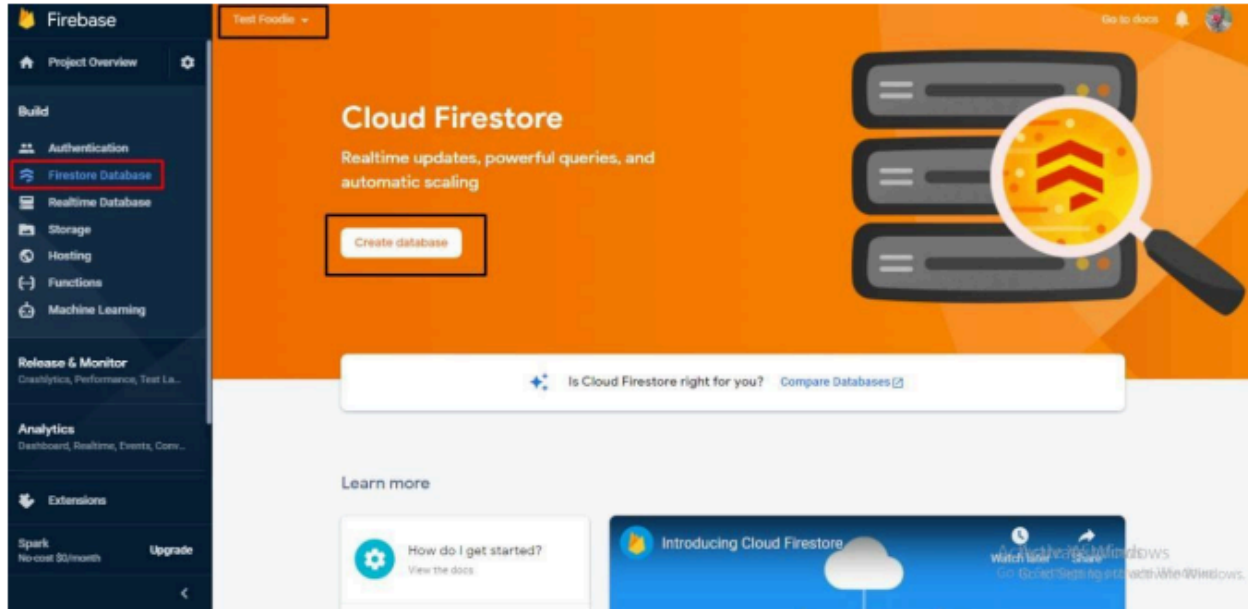


You can add these firebase project details in the project's .env file.

Step 5: Create a RealTime database and copy url and add into project's .env file.

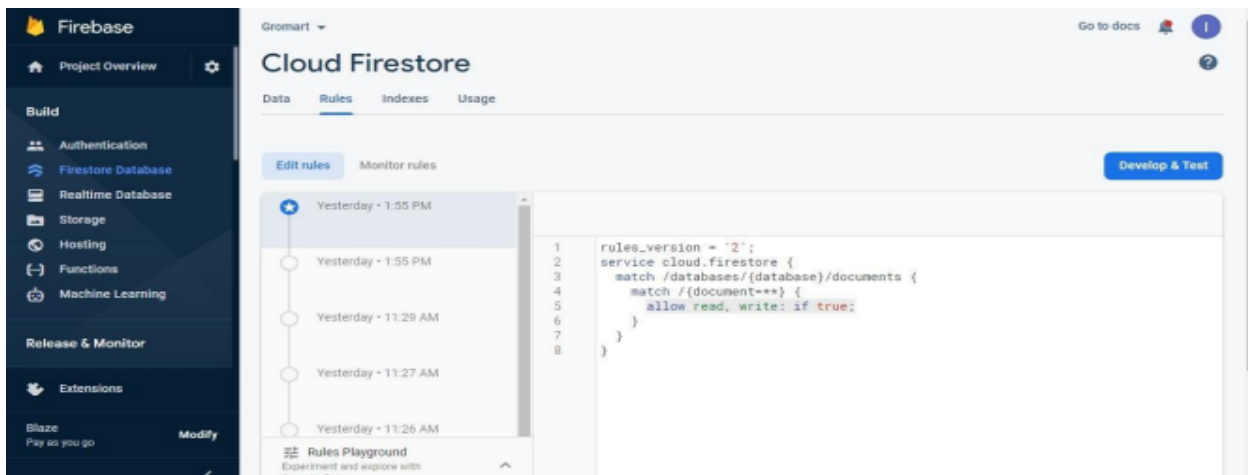


Step 6: After that click on “Firestore Database” in the left sidebar and select your project name from the drop-down and click on “Create database”



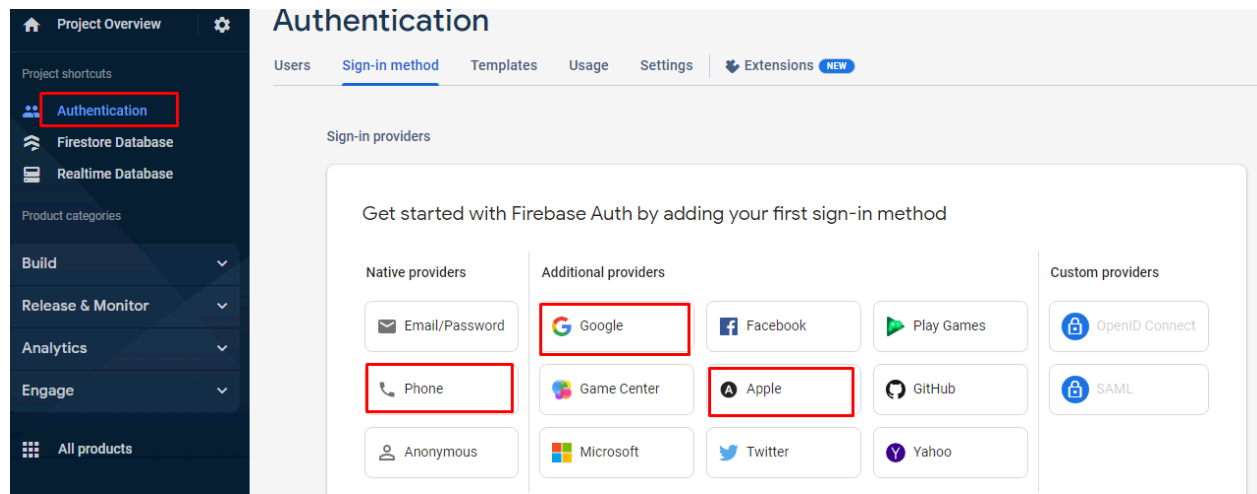
Step 7: Select the preferable option for you and click on the Next button then click on “Enable”.

Step 8: Firstore Database Rules Update.



Step 9: Enable Firebase Authentication Methods

To enable Firebase Authentication, go to Firebase Console -> Authentication -> Sign-in Methods and enable the methods that you are going to support in your app. By default, our Flutter apps have integration with Phone, Google and Apple.



See video for more : [📺 GORIDE - How to setup/configure Firebase Project?](#)

Firestore Database Collection Import/Export

Step 1: Setup NPM in your Computer URL: <https://nodejs.org/en/download>

Step 2: Extract Source Code of firebase-import-export-functions.zip

Step 3: We have to open command prompt on project folder and run command :
“npm install”

Step 4: Setup Firebase Project if you have not created.

Step 5: Configure serviceAccountKey.json file. you can get from forbase
Project settings

Go to ->Service account -> Select Node.js -> Generate new private key Wait untill
create key this will auto download. and Replace with current
serviceAccountKey.json

Step 6: Run command on project folder bellow import commands All Collections are in Folder name “GoRideDataSeed” One By one You have to run command for import each collection

See video for more :  GORIDE : How to import/export collections in firebase?

IMPORT Commands:

node import.js GoRideDataSeed/airports.json

node import.js GoRideDataSeed/banner.json

node import.js GoRideDataSeed/cms_pages.json

node import.js GoRideDataSeed/coupon.json

node import.js GoRideDataSeed/currency.json

node import.js GoRideDataSeed/documents.json

node import.js GoRideDataSeed/driver_rules.json

node import.js GoRideDataSeed/faq.json

node import.js GoRideDataSeed/freight_vehicle.json

node import.js GoRideDataSeed/languages.json

node import.js GoRideDataSeed/on_boarding.json

node import.js GoRideDataSeed/service.json

node import.js GoRideDataSeed/settings.json

node import.js GoRideDataSeed/tax.json

node import.js GoRideDataSeed/vehicle_type.json

Not compulsory to run this commands:

node import.js GoRideDataSeed/bank_details.json

node import.js GoRideDataSeed/driver_document.json

node import.js GoRideDataSeed/driver_users.json

node import.js GoRideDataSeed/intercity_service.json

node import.js GoRideDataSeed/orders.json

node import.js GoRideDataSeed/orders_intercity.json

node import.js GoRideDataSeed/referral.json

node import.js GoRideDataSeed/review_customer.json

node import.js GoRideDataSeed/review_driver.json

node import.js GoRideDataSeed/sos.json

node import.js GoRideDataSeed/users.json

node import.js GoRideDataSeed/wallet_transaction.json

node import.js GoRideDataSeed/withdrawal_history.json

Export commands:

node export.js airports

node export.js bank_details

node export.js banner

node export.js cms_pages

node export.js coupon

node export.js currency

node export.js documents

node export.js driver_document

node export.js driver_rules

node export.js driver_users

node export.js faq

node export.js freight_vehicle

node export.js intercity_service

node export.js languages

node export.js on_boarding

node export.js orders

node export.js orders_intercity

node export.js referral

node export.js review_customer

node export.js review_driver

node export.js service

node export.js settings

node export.js sos

node export.js tax

node export.js users

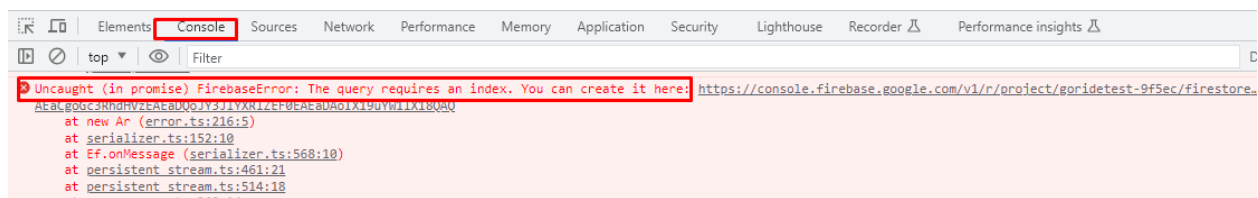
`node export.js vehicle_type`

`node export.js wallet_transaction`

`node export.js withdrawal_history`

Query Indexing

While viewing the admin panel if you see a processing loader then please open your browser console and check the errors. If the errors showing below snap then click on the given URL to do indexing. And it will create index in firebase automatically and errors will be removed from the page.



Deploy Firebase Functions

Note: To deploy Firebase Functions, you need to make sure you upgrade your Firebase Pricing Plan to Blaze.

Step 1: Setting up Node.js and the Firebase CLI

Use the Firebase doc ([https:// rebase.google.com/docs/functions/get-started](https://rebase.google.com/docs/functions/get-started)) if you encounter any issues.

Node.js and Firebase CLI is needed to write functions and deploy them to the Cloud Functions.

Install Node.js (<https://nodejs.org/en/>) and npm (<https://www.npmjs.com/>), Node Version Manager (<https://github.com/nvmsh/nvm/blob/master/README.md>).

After Node.js and npm are installed, install the Firebase CLI.

Use: “npm install -g firebase-tools” command.

Step 2: Initialize your project

An empty project containing sample code is created when you initialize Firebase SDK for cloud functions.

Authenticate the firebase tool by running the firebase login via browser
You need to create a new “MyFirebaseFunctions” empty folder

Go to the source code and find **firebase-cloud-functions.zip** file and extract it and go to
firebase-cloud-functions > functions folder add your firebase credentials in **serviceAccountKey.json**, **index** and **.firebaserc** file.

Step 3: Deploy Firebase Functions

Open powerShell from firebase-cloud-functions > functions and run command

“npm install”

“firebase login” (Select Y(yes) and It will redirect to browser and login with google account which you used for firebase)

Now after successfully login simply run this command

“firebase deploy --only functions”

And it deploy your firebase cloud function and you can also check status in CLI

After successfully deploying the functions you can go to your Firebase Console and check, as the functions have been deployed.

Step 4: Watch Firebase Functions for Errors

It is possible to see the logs for each function, understand the output, and know when it gets called.

Note: To run properly, some firebase functions need the creation of indexes on certain firestore collections.

Use the app and watch the logs for the firebase functions in the console.

If you happen to get an error of missing index, simply click on the URL of that error, and the index will get created automatically.

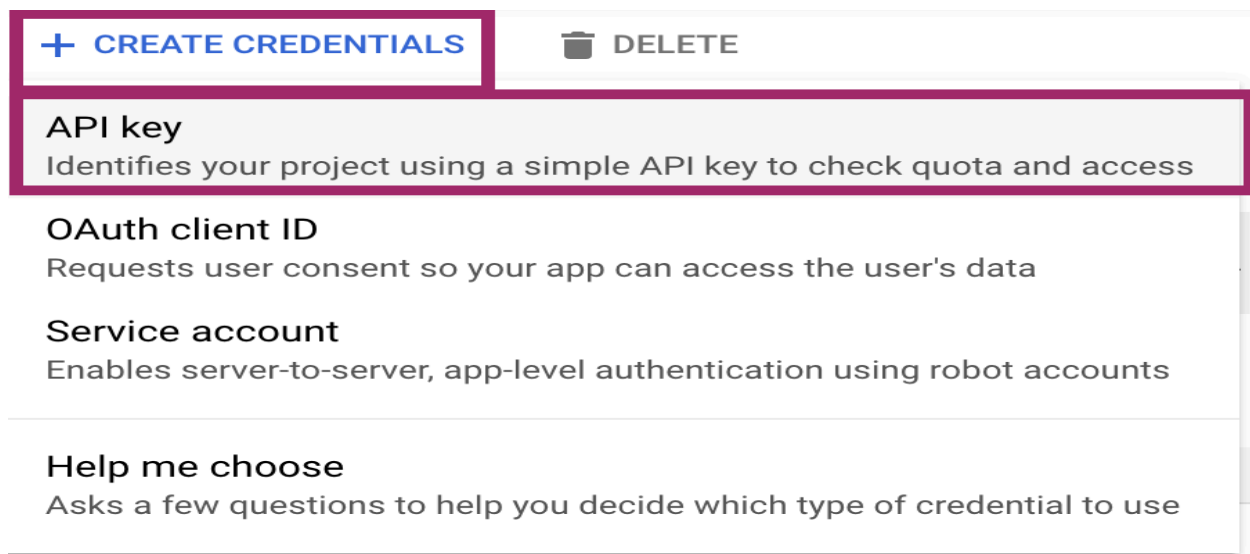
There is no need to deploy functions again, but wait until the index is created, before testing the app again.

See video for more :  **GORIDE - How to setup Cloud function in Firebase**

Create Google Map API Key

1. Go to the [Google Maps Platform](#)
2. Click the Get Started button in the middle of the screen.
3. Click on the Google Cloud Platform home in the upper left corner.
4. Click on Billing to make sure your billing details are up-to-date. If they are not, your Google Maps will not work properly.
5. Once you've confirmed your billing is up-to-date, click on the Google Cloud Platform home in the upper left corner again.

6. Hover to APIs & Services and go to Credentials.
7. If you want to use an existing project, please select it from the list. Otherwise, select 'Create a new project' and enter a project name.
8. Click Create credentials and select API key. You will see a new dialog that displays the newly created API key.



9. Click the Close button in the API key dialogue. Your new API key will be listed on the Credentials page under API keys

Admin Panel - Update

To update the admin panel just upload our latest source code zip file in your project root folder and extract it.

If you customized something on the code and want to update to our latest version then you can follow any one option from following this.

1. First push your code on a git branch then download our latest version source code from codecanyon and push it to another branch. And then you

can merge the changes between current version and previous version code and merge both branches and it is possible to get conflicts on branches.

2. Download the our latest version source code from codecanyon and apply.

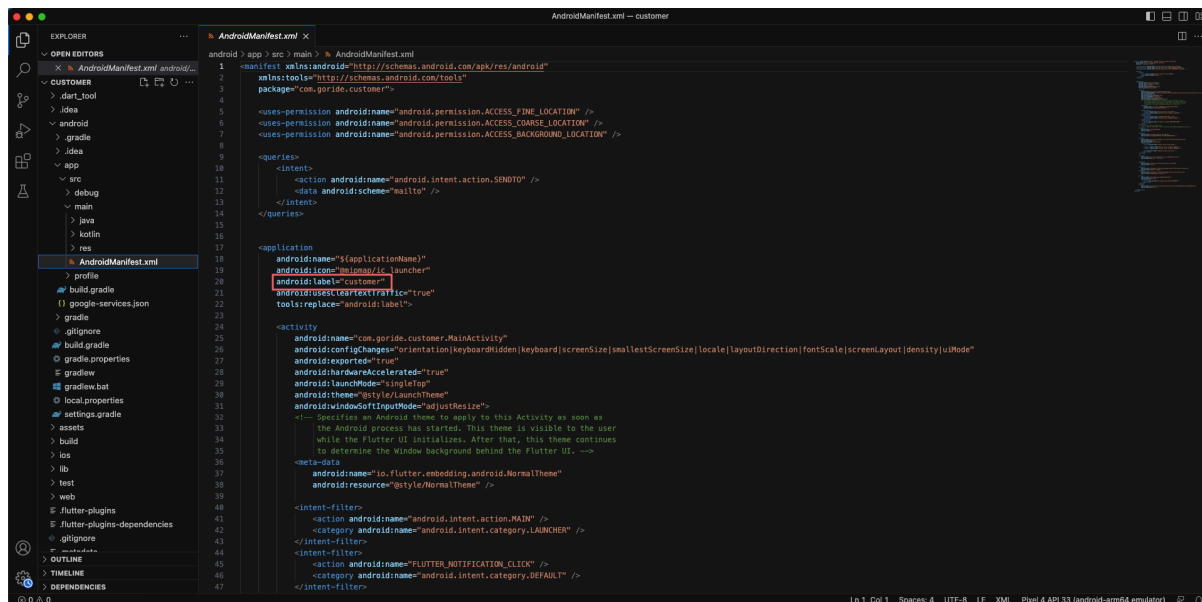
Application Setup

How to set up the project

Change App Name **(For both application customer and driver follow same step)**

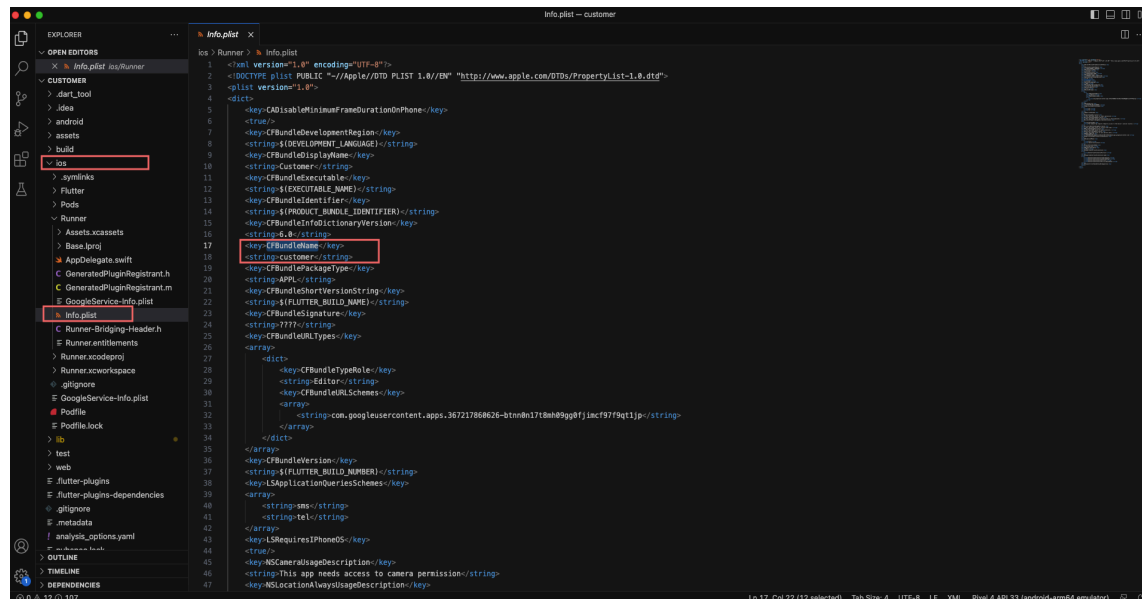
1. Change the value of label from

`<project>/android/app/src/main/AndroidManifest.xml`



2. Change the value of CFBundleName from **(For both application customer and driver follow same step)**

<project>/iOS/Runner/info.plist



Change App Package (For both application customer and driver follow same step)

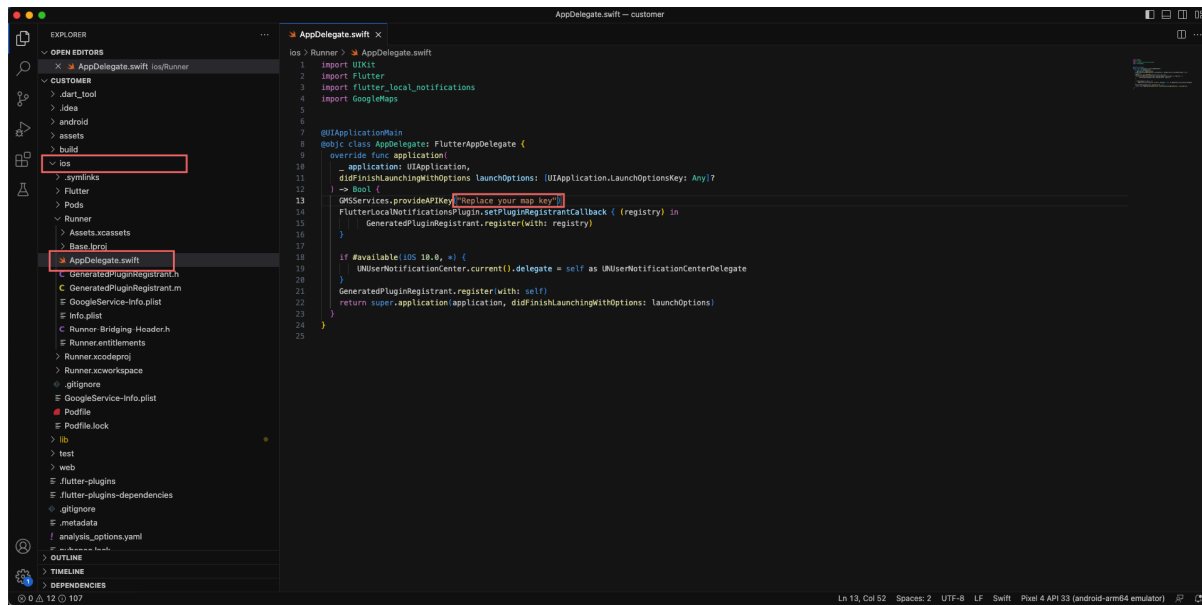
First you have to find out the existing package name. You can find it out from the top of the /app/src/main/AndroidManifest.xml file. Now on the right side of the project folder from VSCode. In the first box you have to put the existing package name that you saw in the AndroidManifest.xml file previously and write down your preferred package name in the second box and then click on Replace All

The image shows the Android Studio IDE interface. On the left, the 'SEARCH' tab is active, displaying a search for 'com.goride.customer'. The search results are listed in a tree view, showing files like 'build.gradle', 'google-services.json', 'AndroidManifest.xml', and 'MainActivity.kt'. The 'MainActivity.kt' file is selected, and its content is displayed in the main editor area on the right. The code in 'MainActivity.kt' shows the package name 'com.goride.customer' and the class name 'MainActivity'. The bottom status bar indicates the device is 'Pixel 4 API 33 (android-arm64 emulator)' and the CPU usage is 107%.

1. Android

[illegible]

<project>/ios/Runner/AppDelegate.swift



Setup with firebase

1. Setup with firebase using flutterFire.

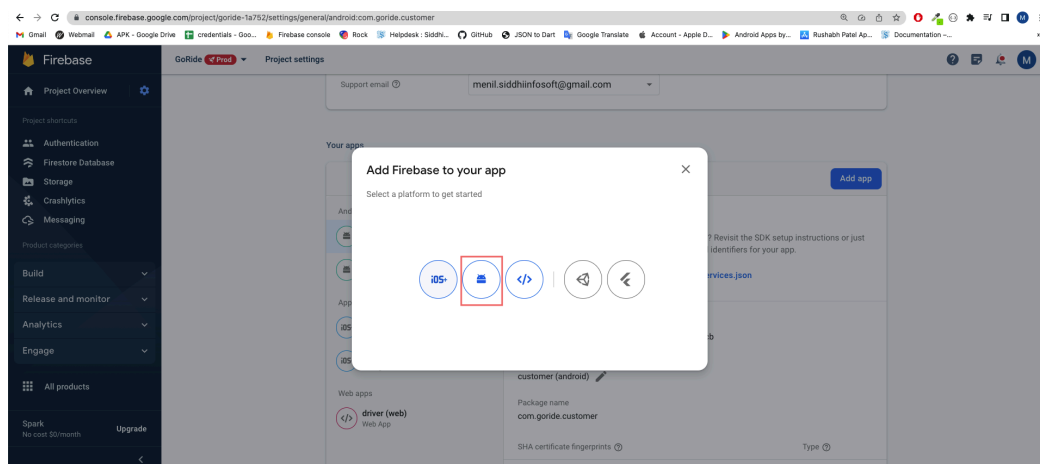
a. Follow this link to setup with firebase :-

<https://firebase.google.com/docs/flutter/setup?platform=ios>

b. Video link :- [Add Firebase to your Flutter app: The fast way](#)

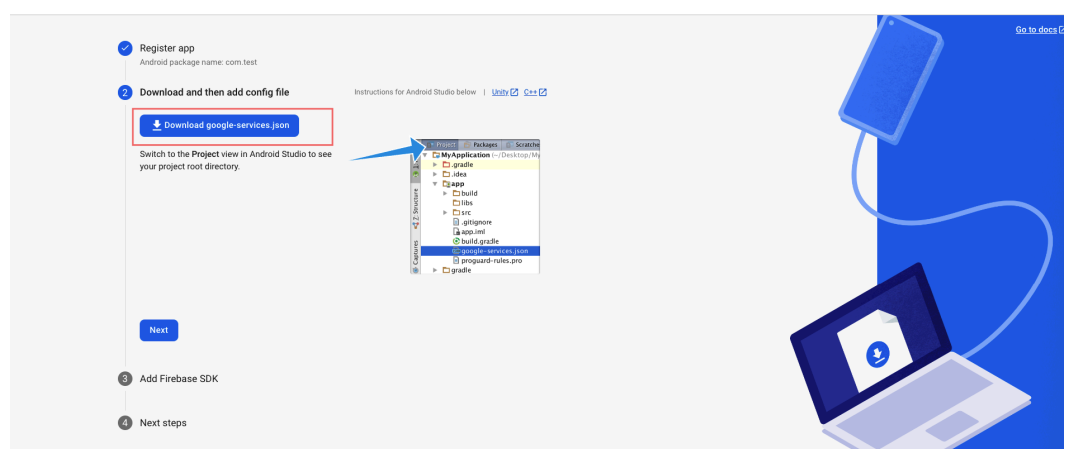
2. Setup manually.

a. For Android

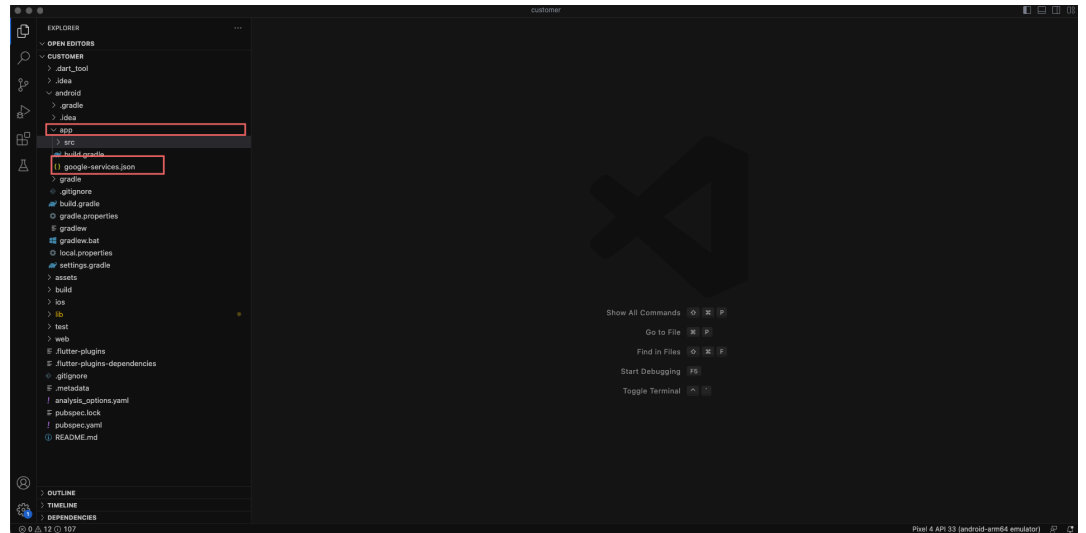


[illegible]

c. Download google-services.json file.

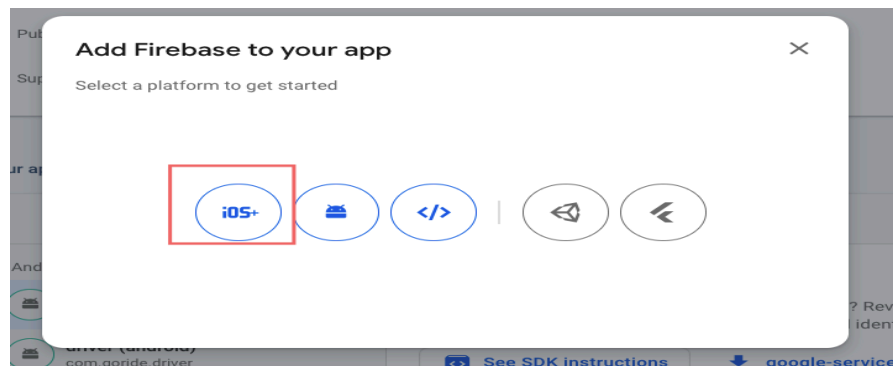


d. Place google-services.json file in <project>/android/app/



e. For IOS.

1. Add App



1(i). Enter package name and download GoogleService-Info.plist

1 Register app

Apple bundle ID ⓘ

com.company.appname

App nickname (optional) ⓘ

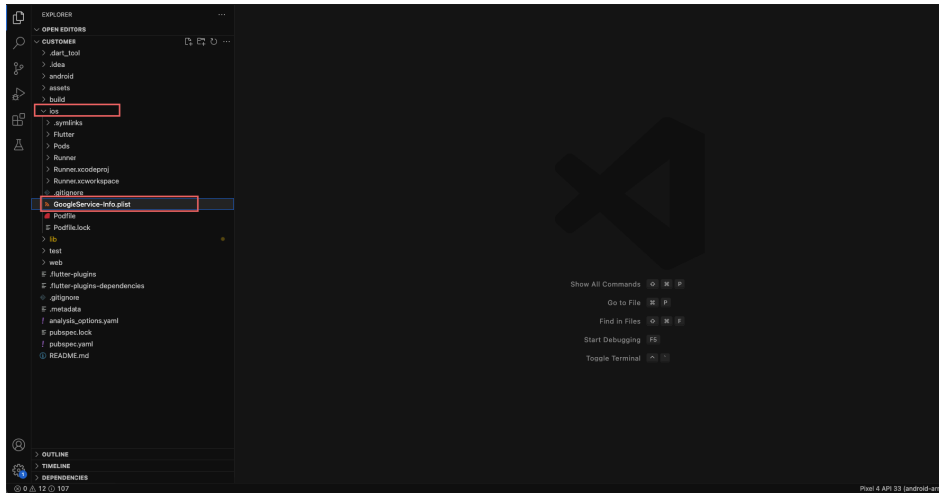
My Apple app

App Store ID (optional) ⓘ

123456789

Register app

1(ii). Place GoogleService-Info.plist in <project>/iOS/



Generate SHA-1 for Flutter

Link :-

<https://stackoverflow.com/questions/51845559/generate-sha-1-for-flutter-react-native-android-native-app>

App build & release

1. Build for Android

For debug build you can run command:

flutter build apk

You will get a larger merged apk with this. But you can split them with this command:

flutter build apk --target-platform android-arm,android-arm64,android-x64 --split-per-abi

Build file location: <project>/build/app/outputs/flutter-apk/ For deploying it please follow this documentation:

<https://docs.flutter.dev/deployment/android>

2. Build for iOS

There is no general way to generate apps for iOS. Apple doesn't allow you to install apps like this. If you want to install it on your iOS device then you have to deploy it on TestFlight or AppStore. For deploying it please follow this documentation: <https://docs.flutter.dev/deployment/ios>

How to Change Flutter Package Name

Here are the steps to change the package name in Flutter for both Android and iOS:

Android (in AndroidManifest.xml and build.gradle):

1. Open the `android/app/src/main/AndroidManifest.xml` file.
2. Locate the `package` attribute in the `<manifest>` element. Change the value to your desired package name.

3. Open the `android/app/build.gradle` file.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.newpackagename">
    <!-- ... other configurations ... -->
</manifest>
```

4. Locate the `applicationId` field inside the `defaultConfig` block. Update it with the new package name.

```
android {
    defaultConfig {
        applicationId "com.example.newpackagename"
        // ... other configurations ...
    }
}
```

iOS (in Info.plist):

1. Open the `ios/Runner/Info.plist` file.
2. Look for the `CFBundleIdentifier` key. Change the value to your new package name.


```
<key>CFBundleIdentifier</key>  
<string>com.example.newpackagename</string>
```

Important notes:

- After changing the package name, you might need to run `flutter clean` in the terminal to remove any cached build files.
- Make sure to update any references to the old package name in your code, such as imports or other configurations that use the package name.
- If you are using Firebase or other services that rely on the package name, you may need to update the configuration on those platforms as well.
- Changing the package name can impact the app's update mechanism on app stores. Ensure you understand the implications before making such changes, especially for apps already published.

After making these changes, you should be able to run your Flutter app with the new package name on both Android and iOS platforms.

How to Change Flutter App Name

1. Open the `pubspec.yaml` file in the root of your Flutter project.
2. Locate the `name` field in the file. It typically looks like this:
`name: your_app_name`
3. Change the value of the `name` field to your desired application name. For example:
`name: new_app_name`
4. Save the changes to the `pubspec.yaml` file.
5. Run the following command in your terminal to get the updated dependencies:
`flutter pub get`
6. After updating the `pubspec.yaml` and running `flutter pub get`, your application name should be updated.

Keep in mind that changing the application name won't automatically update the display name on the device home screen or app launcher. For that, you may need to update the `applicationLabel` in the `AndroidManifest.xml` file for Android, and the `CFBundleName` and `CFBundleDisplayName` in the `Info.plist` file for iOS.

For Android (located in `android/app/src/main/AndroidManifest.xml`), you might find something like:

```
<application
  android:label="Your App Name"
  ...
>
  ...
</application>
```

For iOS (located in `ios/Runner/Info.plist`), you might find something like:

```
<key>CFBundleName</key>
<string>$(PRODUCT_NAME)</string>

<key>CFBundleDisplayName</key>
<string>Your App Name</string>
```

Update the values accordingly and rebuild the app for the changes to take effect. After making these changes, the updated application name should be reflected on both Android and iOS devices.

How to Change Launcher Logo/Icon

Using a Package (`flutter_launcher_icons`):

```
Dev dependencies:
  flutter_launcher_icons: ^0.9.2
```

1. Run `flutter packages get` in the terminal to fetch the package.
2. Open your `pubspec.yaml` file and add the following dependency:

```
flutter_icons:
  android: true
  ios: true
  image_path: "assets/icon/app_icon.png"
```

Make sure to replace `"assets/icon/app_icon.png"` with the path to your custom icon image.

3. Run the following command in the terminal to generate the icons:

```
flutter pub run flutter_launcher_icons:main
```

4. Build and run your app:

```
flutter run
```

Manual Method:

1. Prepare your custom icon image and place it in a suitable location within your project. For example, you can create a folder named `assets` in the root of your project and place the icon there.

2. Open your `pubspec.yaml` file and add the following section to include the assets:

```
Flutter:  
  assets:  
    - assets/icon/app_icon.png
```

Make sure to adjust the path based on your project structure.

3. For Android:

Replace the launcher icon in the `android/app/src/main/res/mipmap` folder. You will typically find folders like `mipmap-hdpi`, `mipmap-mdpi`, `mipmap-xhdpi`, etc.

Replace the launcher icon in each of these folders.

4. For iOS:

Replace the launcher icon in the `ios/Runner/Assets.xcassets` folder. There should be an `AppIcon` set with various icon sizes for different iOS devices.

Replace the icons in each size.

5. Build and run your app:

```
flutter run
```

These steps should help you change the launcher icon in your Flutter app either using a package or manually.

How to Create Project in Firebase

1. Visit the Firebase Console:

- Open your web browser and go to the Firebase Console.

2. Sign in with Google:

- If you don't have a Google account, you'll need to create one. Otherwise, sign in with your existing Google account.

3. Create a New Project:

- Click on the "Add project" button.

4. Enter Project Name:

- Enter a name for your project in the provided field.

5. (Optional) Modify Project ID:

The project ID is automatically generated based on the project name. If needed, you can modify it.

6. Select Google Analytics:

- You can choose to enable Google Analytics for your project. This step is optional.

7. Accept Terms and Click "Create Project":

- Read and accept the terms, then click on the "Create project" button.

8. Wait for Project Creation:

- Firebase will take a moment to create your project. Once it's done, you'll be redirected to the project dashboard.

Now you have successfully created a project in Firebase. From the project dashboard, you can access various Firebase services, configure settings, and integrate Firebase into your applications.

Remember that Firebase provides a variety of services, so you may want to explore and configure specific services based on your project requirements. For example, you might want to set up authentication, a real-time database, cloud functions, or hosting for your web application. Each service has its own configuration steps, which you can find in the Firebase documentation.

Always refer to the official Firebase documentation for the most up-to-date and detailed information.

How to Setup Firebase Project in Flutter

Using Firebase CLI:

1. Install Firebase CLI:

Make sure you have Node.js and npm installed on your machine.

Install the Firebase CLI by running the following command in your terminal or command prompt:

```
npm install -g firebase-tools
```

2. Login to Firebase:

Run the following command to log in to Firebase using your Google account:

```
firebase login
```

3. Create a Firebase Project:

Go to the Firebase Console, and click on "Add project."

Follow the instructions to create a new project.

4. Initialize Firebase in your Flutter project:

Navigate to your Flutter project's root directory in the terminal.

Run the following command to initialize Firebase in your project:

```
firebase init
```

Follow the prompts to select the features you want to use (e.g., Firestore, Authentication) and set up the necessary configurations.

5. Update Firebase Configuration in Flutter:

After initialization, Firebase CLI will create a `firebase.json` file. Use the generated configuration in your Flutter app.

Make sure to copy the configuration settings for each feature you've enabled (e.g., Firestore, Authentication).

Manual Integration:

1. Add Firebase to Flutter:

Open your Flutter project and add the `firebase_core` and other Firebase packages to your `pubspec.yaml` file:

```
dependencies:
  firebase_core: ^latest_version
  firebase_auth: ^latest_version # Example: Add other Firebase
packages
  cloud_firestore: ^latest_version
```

2. Initialize Firebase in Flutter:

In your main Dart file (e.g., `main.dart`), initialize Firebase in the `main` function:

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

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

3. Use Firebase Services:

You can now use Firebase services like Firestore, Authentication, etc., in your Flutter app.

4. Add Firebase Configuration:

Add the Firebase configuration settings (copied from the `firebase.json` file during

Firebase CLI initialization) to your app. For example, add them to your

`android/app/google-services.json` file for Android and

`ios/Runner/GoogleService-Info.plist` for iOS.

5. Run Your App:

Run your Flutter app using the following command:

```
flutter run
```

This process involves both Firebase CLI and manual steps to integrate Firebase services into your Flutter project. Ensure that you have the necessary dependencies in your `pubspec.yaml` file and configure Firebase according to your project's needs.

How to Change Google Map Key in Flutter

Android (AndroidManifest.xml):

1. Open the `AndroidManifest.xml` file in the `android/app/src/main` directory of your Flutter project.
2. Locate the `<meta-data>` element inside the `<application>` element, which contains the Google Maps API key. It should look something like this:

```
<application>
  ...
  <meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY_HERE" />
  ...
</application>
```

3. Replace `"YOUR_API_KEY_HERE"` with your actual Google Maps API key.

iOS (AppDelegate.swift):

1. Open the `AppDelegate.swift` file in the `ios/Runner` directory of your Flutter project.
2. Locate the `application(_:didFinishLaunchingWithOptions:)` method and find the code that sets the Google Maps API key. It should look something like this:

```
GMSServices.provideAPIKey("YOUR_API_KEY_HERE")
```

3. Replace `"YOUR_API_KEY_HERE"` with your actual Google Maps API key.

Important notes:

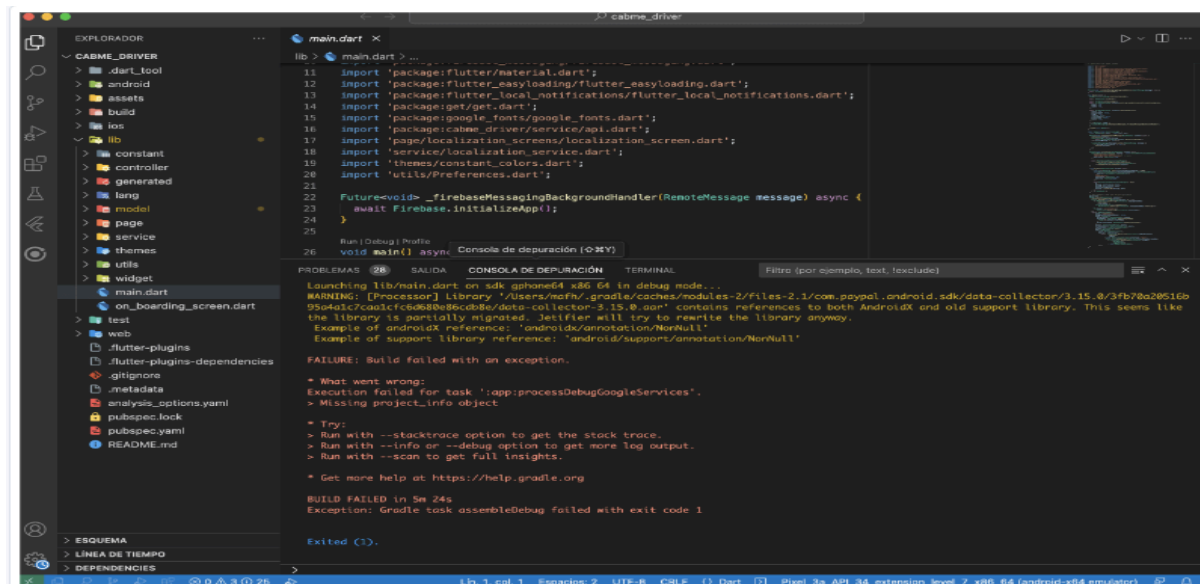
- Make sure that you have the appropriate API key for both Android and iOS platforms from the Google Cloud Console.
- Ensure that the API key has the necessary permissions for the Google Maps SDK.
- Keep your API keys secure and do not expose them publicly, especially in version control repositories.

After making these changes, rebuild your Flutter project for both Android and iOS platforms to apply the updated API keys. You can do this by running `flutter clean` followed by `flutter run` in your project directory.

See video for more: [📺 GORIDE - How to SetUp GoRide Flutter application?](#)

IMPORTANT NOTES

1) If you're facing the following attached error while setting up the project, make sure you have downloaded `google-service.json` file from the firebase and replace the original one. If you do not replace `google-service.json` then you will face the below image issue. Although, if you will face any kind of issue related to this, feel free to raise a support ticket to <https://support.siddhiinfosoft.com>



2) If you're facing the following attached error while setting up the project, make sure you have successfully configured your firebase project otherwise the application won't work.


```
Console
W/Firestore( 6649): This typically indicates that your device does not have a healthy Internet connection at the moment. The client will operate in offline mode until it is able to successfully connect to the backend.
E/flutter ( 6649): [ERROR:flutter/runtime/dart_vm_initializer.cc(41)] Unhandled Exception: [firebase_messaging/unknown] Please set your Application ID. A valid Firebase App ID is required to communicate with Firebase server APIs: It identifies your application with Firebase. Please refer to https://firebase.google.com/support/privacy/init-options.
I/flutter ( 6649): #0 StandardMethodCodec.decodeEnvelope (package:flutter/src/services/message_codecs.dart:651:7)
E/flutter ( 6649): #1 MethodChannel.invokeMethod (package:flutter/src/services/platform_channel.dart:334:18)
E/flutter ( 6649): <asynchronous suspension>
E/flutter ( 6649): #2 MethodChannel.invokeMapMethod (package:flutter/src/services/platform_channel.dart:534:43)
E/flutter ( 6649): <asynchronous suspension>
E/flutter ( 6649): #3 MethodChannelFirebaseMessaging.getToken (package:firebase_messaging_platform_interface/src/method_channel/method_channel_messaging.dart:248:11)
E/flutter ( 6649): <asynchronous suspension>
E/flutter ( 6649): #4 NotificationService.getToken (package:foodie_driver/services/notification_service.dart:64:21)
E/flutter ( 6649): <asynchronous suspension>
E/flutter ( 6649): #5 MyAppState.notificationInit.<anonymous closure> (package:foodie_driver/main.dart:69:22)
E/flutter ( 6649): <asynchronous suspension>
E/flutter ( 6649):
[log] ::::::::::::::Permission authorized::::::::::::::::::
I/FLTFireMsgService( 6649): FlutterFirebaseMessagingBackgroundService started!
I/flutter ( 6649): [Easy Localization] [DEBUG] Build
I/flutter ( 6649): [Easy Localization] [DEBUG] Init Localization Delegate
I/flutter ( 6649): [Easy Localization] [DEBUG] Init provider
I/flutter ( 6649): [Easy Localization] [WARNING] Localization key [Flutter Uber Eats Driver] not found
```

Thank You

© 2023 GoRide. All Rights Reserved.