



## **Eber Driver Documentation**

# 1. Install Xcode On Mac

- Open the App Store on your mac.
- Sign in.
- Search for Xcode.
- Click install or update.

Please check the link to download xcode from the app store:-

<https://apps.apple.com/us/app/xcode/id497799835?mt=12>

Note : To install a specific version or latest version on your mac system your Mac OS must need to be compatible with the version of xcode. For example, to install the latest xcode version 13.1 from the App Store, it requires that your system is updated with Mac OS version 11.3 or later.

For more details you can check this link :

<https://medium.com/@LondonAppBrewery/how-to-download-and-setup-xcode-10-for-ios-development-b63bed1865c>

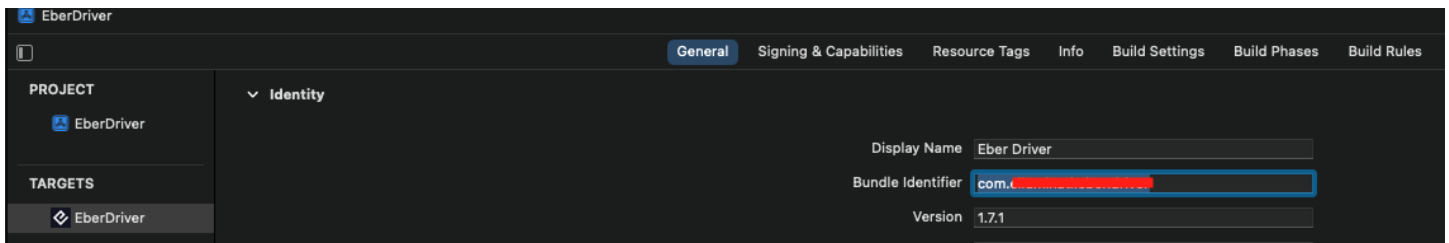
## 2. Changes In Projects (iOS)

### 2.1 Open Project in Xcode

**File Goto** : -> open -> Select .xcworkspace file of your project which is located on your system.

### 2.2 Change bundle identifier

- In the project navigator, select the project and your target to display the project editor.
- Click General Tab
- In the identity section change the Bundle Identifier field. See the below screenshot below for this change:



### 2.3 Change BASE URL from constant file

**File Goto** : -> EberDriver -> HelperClass -> Constants -> Constants.swift

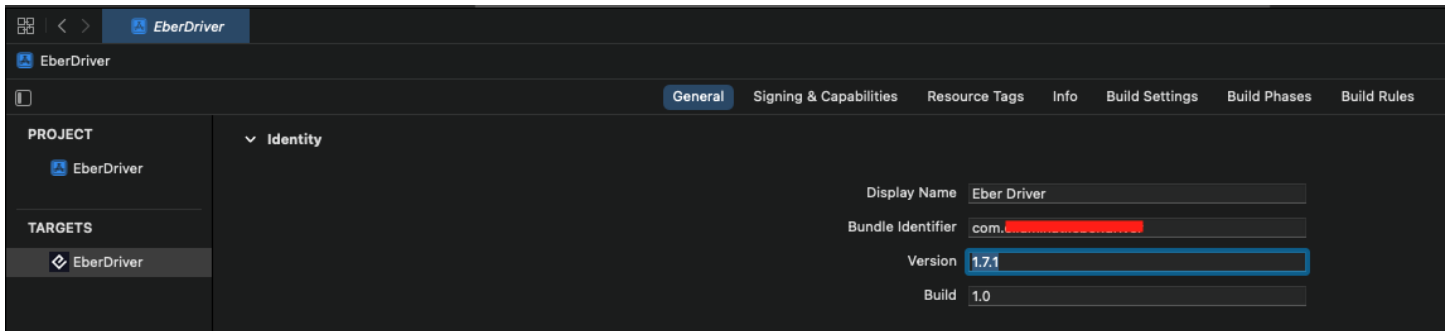
```
struct WebService
{
    //    Test
    static let BASE_URL = "https://*****t/"
    static var IMAGE_BASE_URL = "https://*****t/"

    //    Live
    //    static let BASE_URL = "*****"
    //    static var IMAGE_BASE_URL = "*****"

    //    Local
    //    static let BASE_URL = "*****"
```

## 2.4 Change App version number / Build version number

→ You can change the app version and build version from the identity section of the general tab.



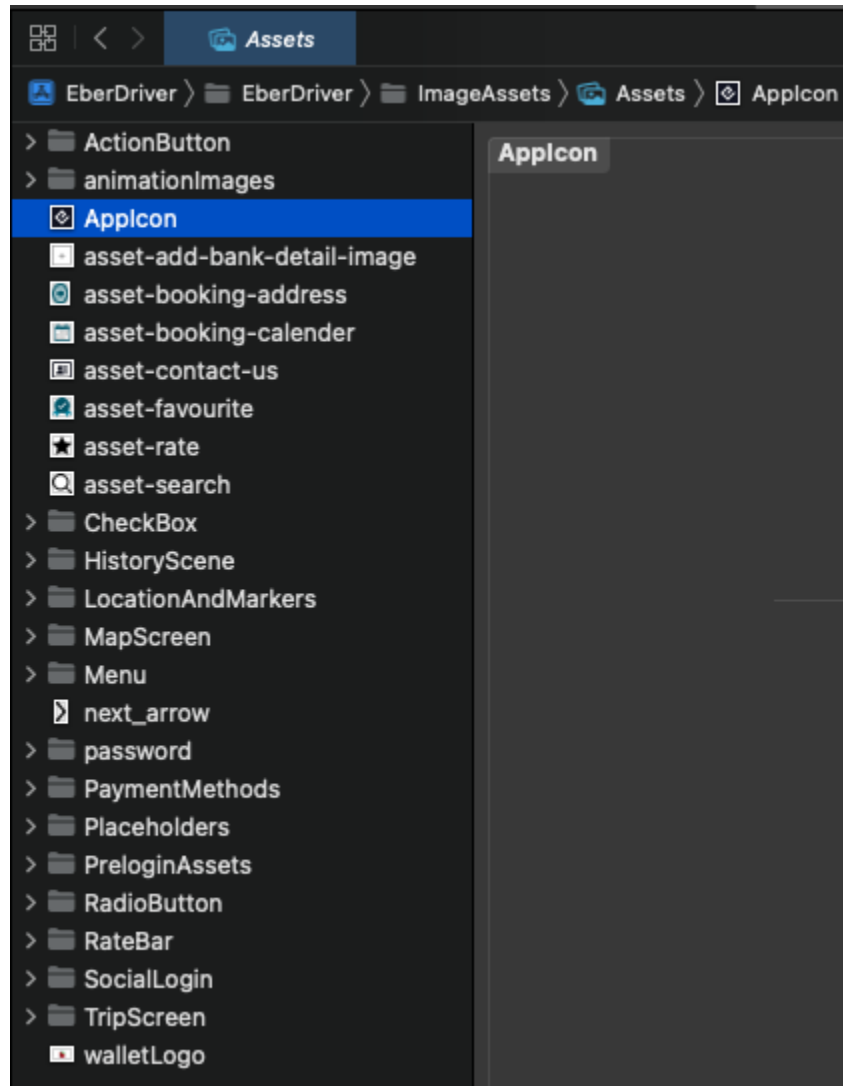
## 2.5 Change your theme color

**File Goto** : EberDriver -> HelperClass -> AppThemeHelper -> myAppTheme.swift, where you can change Section background color, Button background color, Theme color etc..

```
8 import UIKit
9
10 extension UIColor
11 {
12     var imageRepresentation : UIImage {
13         let rect = CGRect(x: 0.0, y: 0.0, width: 320.0, height:64.0)
14         UIGraphicsBeginImageContext(rect.size)
15         let context = UIGraphicsGetCurrentContext()
16
17         context?.setFillColor(self.cgColor)
18         context?.fill(rect)
19
20         let image = UIGraphicsGetImageFromCurrentImageContext()
21         UIGraphicsEndImageContext()
22         return image!
23     }
24
25     static let themeImageColor:UIColor = UIColor(red: 25/255, green: 96/255, blue: 118/255, alpha: 1.0)
26     //Custom Button Colors
27     static let themeButtonBackgroundColor:UIColor = UIColor(red: 25/255, green: 96/255, blue: 118/255, alpha: 1.0)
28     static let themeButtonTitleColor:UIColor = UIColor(red:255/255, green:255/255 ,blue:255/255 , alpha:1.00)
29     static let themeSelectionColor:UIColor = UIColor(red: 0/255, green: 0/255, blue: 0/255, alpha: 1.0)
30
31     static let themeSwitchTintColor:UIColor = UIColor(red: 0/255, green: 0/255, blue: 0/255, alpha: 1.0)
32
33     //Custom TextFieldColors
34     static let themeActiveTextColor:UIColor = UIColor(red: 0/255, green: 0/255, blue: 0/255, alpha: 1.0)
35     static let themeErrorTextColor:UIColor = UIColor(red:220/255, green:74/255 ,blue:48/255 , alpha:1.0)
36     static let themeTextColor:UIColor = UIColor(red: 0/255.0, green: 0/255.0, blue: 0/255.0, alpha: 1.0)
37     static let themeLightTextColor:UIColor = UIColor(red:0/255, green:0/255 ,blue:0/255 , alpha:0.5)
38
39     //NavigationBar Colors
40     static let themeTitleColor:UIColor = UIColor(red:0/255, green:0/255 ,blue:0/255 , alpha:1.00)
41     static let themeNavigationBackgroundColor:UIColor = UIColor(red: 255/255, green: 255/255, blue: 255/255, alpha: 1.0)
42
43     //Theme Dialog Colors
44     static let themeOverlayColor:UIColor = UIColor(red: 80/255.0, green: 80/255.0, blue: 80/255.0, alpha: 0.8)
45     static let themeDialogBackgroundColor:UIColor = UIColor(red: 255/255.0, green: 255/255.0, blue: 255/255.0, alpha: 1.0)
46
47     static let themeViewBackgroundColor:UIColor = UIColor(red: 255/255.0, green: 255/255.0, blue: 255/255.0, alpha: 1.0)
48     static let themeDividerColor:UIColor = UIColor(red: 80/255.0, green: 80/255.0, blue: 80/255.0, alpha: 1.0)
49     static let themeLightDividerColor:UIColor = UIColor(red:136/255, green:139/255 ,blue:136/255 , alpha:0.7)
50 }
```

## 2.6 Change images

**File Goto :** EberDriver -> ImageAssets -> Assets with .xcassets extension



## 2.7 Change font

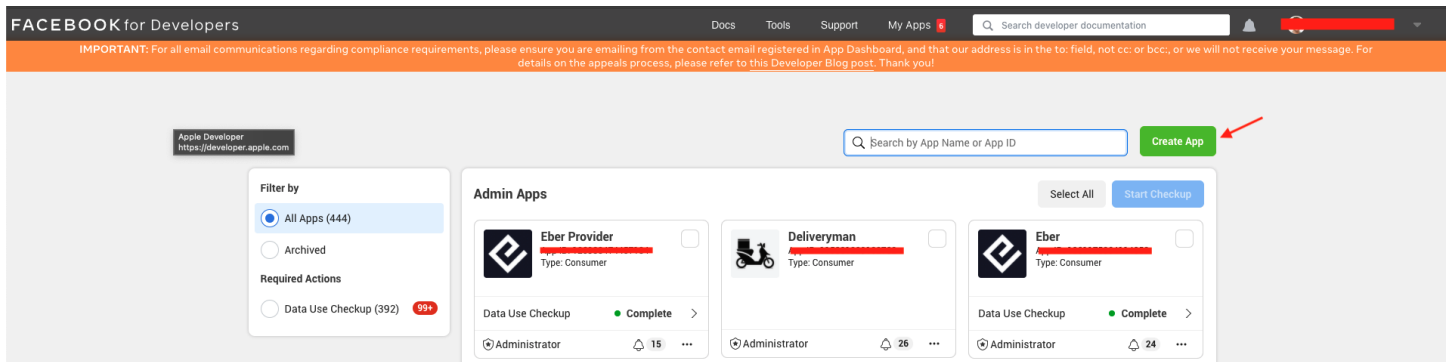
**File Goto** : EberDriver -> HelperClass -> AppThemeHelper -> myAppTheme.swift - where you can change font name, size and types.

```
155 enum FontType {
156     case Bold
157     case Light
158     case Regular
159 }
160
161 class FontHelper:UIFont {
162     class func font(size: CGFloat = FontSize.regular,type:FontType) -> UIFont {
163         switch type {
164             case .Bold:
165                 return UIFont(name: "Roboto-Bold", size: size)!
166             case .Light:
167                 return UIFont(name: "Roboto-Regular", size: size)!
168             case .Regular:
169                 return UIFont(name: "Roboto-Regular", size: size)!
170         }
171     }
172
173     class func assetFont(size: CGFloat = FontSize.regular) -> UIFont {
174         return UIFont(name: "eber", size: size)!
175     }
176 }
```

## 2.8 Set up for facebook Sign In Feature

→ For enabling facebook social login, Create facebook account after open facebook developer site: <https://developers.facebook.com/apps/>

### 2.8.1 Create a New App



### 2.8.2 Open app -> Click on settings -> basic -> + Add Platform -> after select iOS

→ Enter bundle id and other required details and save.

The image shows the 'iOS' configuration screen in the Facebook for Developers app settings. At the top right, there's a 'Quick Start' button and a close icon. The screen is divided into several sections: 'Bundle ID' with a text input field containing 'com. [redacted]'; 'iPhone Store ID' with a text input field containing 'The ID to identify your app in the iOS Store'; 'URL Scheme Suffix - Optional' with an empty text input field; 'iPad Store ID' with a text input field containing 'The ID to identify your app in the iPad Store'; 'Shared Secret' with a text input field containing 'iOS App shared secret'; and a section with three toggle switches: 'Single Sign On' (checked), 'Deep Linking' (unchecked), and 'Log In-App Events Automatically (Recommended)' (unchecked). Below the 'Log In-App Events Automatically' toggle, there's a detailed explanation of its function and a link to 'Learn More'.

## 2.8.3 Now you get one app id.

Dashboard

Settings

Basic

Advanced

Roles

Alerts

App Review

Products

Facebook Login

App Events

Activity Log

Activity Log

We've restricted this app for violating the Facebook Platform Policies. Read more or appeal

App ID

App Secret

Display Name

Namespace

App Domains

Contact Email

Privacy Policy URL

Terms of Service URL

User Data Deletion

Data Deletion Callback URL

App Icon (1024 x 1024)

App Purpose

This app's primary purpose is to access and use data from Facebook's Platform on behalf of:

☒ Yourself or your own business

☐ Clients

Select this option if the primary purpose of this app is to manage data or assets on behalf of an individual client or multiple clients.

Discard Save changes

If you are developing an app that accesses and/or stores sensitive information, you must select the "Yourself or your own business" option.

After you get App ID, you first need to set this id in your project. For more details you can follow this tutorial : <https://www.youtube.com/watch?v=P6uZ0o6xDA4>

## 2.8.4 Configure the Info.plist file with an XML snippet that contains data about your app.

1. Right-click Info.plist, and choose Open As ► Source Code.
2. Copy and paste the following XML snippet into the body of your file ( <dict>...</dict>).

```
<key>CFBundleURLTypes</key>
<array>
<dict>
<key>CFBundleURLSchemes</key>
<array>
<string>fbAPP-ID</string>
</array>
</dict>
</array>
<key>FacebookAppID</key>
```



```
<string>APP-ID</string>
<key>FacebookClientToken</key>
<string>CLIENT-TOKEN</string>
<key>FacebookDisplayName</key>
<string>APP-NAME</string>
```

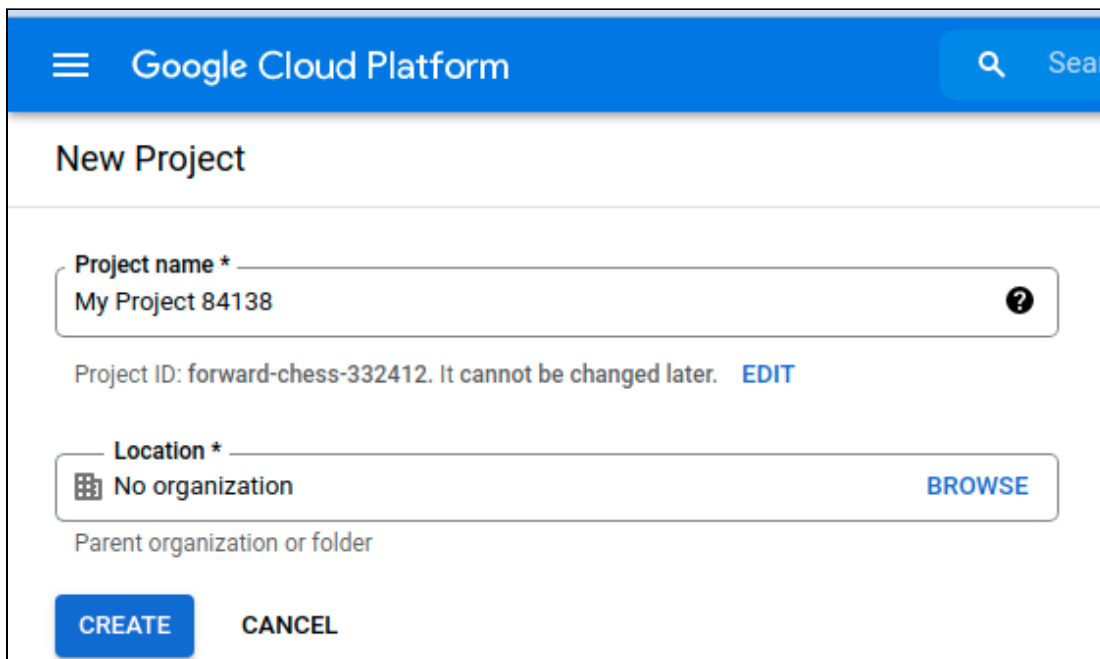
3. In `<array><string>` in the key `[CFBundleURLSchemes]`, replace *APP-ID* with your App ID.
4. In `<string>` in the key `FacebookAppID`, replace *APP-ID* with your App ID.
5. In `<string>` in the key `FacebookClientToken`, replace *CLIENT-TOKEN* with the value found under Settings > Advanced > Client Token in your App Dashboard.
6. In `<string>` in the key `FacebookDisplayName`, replace *APP-NAME* with the name of your app.
7. To use any of the Facebook dialogs (e.g., Login, Share, App Invites, etc.) that can perform an app switch to Facebook apps, your application's Info.plist also needs to include:  
`<dict>...</dict>`).

```
<key>LSApplicationQueriesSchemes</key>
<array>
<string>fbapi</string>
<string>fbapi20130214</string>
<string>fbapi20130410</string>
<string>fbapi20130702</string>
<string>fbapi20131010</string>
<string>fbapi20131219</string>
<string>fbapi20140410</string>
<string>fbapi20140116</string>
<string>fbapi20150313</string>
<string>fbapi20150629</string>
<string>fbapi20160328</string>
<string>fbauth</string>
<string>fb-messenger-share-api</string>
<string>fbauth2</string>
<string>fbshareextension</string>
</array>
```

## 2.9 Google Cloud Console (Google Apis)

- For Using Google Apis (Google Map Api, Geocoding Api, Distance matrix Api etc) In our project we need to create project in google cloud console

1. Open the [Google Cloud Console](#).
2. Next to "Google Cloud Platform," click the Down arrow . A dialog listing current projects appears.
3. Click **New Project**. The New Project screen appears.
4. In the **Project Name** field, enter a descriptive name for your project. If you're executing a quickstart, use "Quickstart."
5. Click **Organization** and select your organization.
6. In the **Location** field, click **Browse** to display potential locations for your project.
7. Click a location and click **Select**.
8. Click **Create**. The console navigates to the Dashboard page and your project is created within a few minutes.

The image shows the 'New Project' form in the Google Cloud Platform console. At the top is a blue header with the Google Cloud Platform logo and a search bar. Below the header, the title 'New Project' is displayed. The form contains two main input fields: 'Project name \*' with the text 'My Project 84138' and a help icon, and 'Location \*' with the text 'No organization' and a 'BROWSE' button. Below the location field, there is a label 'Parent organization or folder'. At the bottom of the form are two buttons: 'CREATE' and 'CANCEL'. A project ID 'forward-chess-332412' is shown with a note that it cannot be changed later and an 'EDIT' link.

For further information on GCP projects, refer to [Creating and managing projects](#).

### 2.9.1 Activate Billing

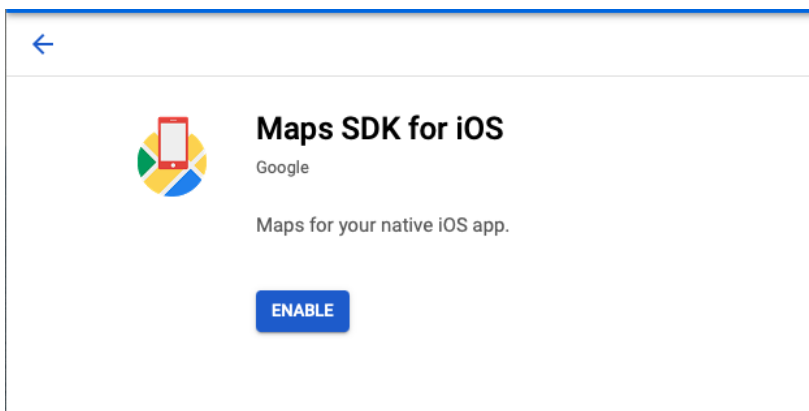
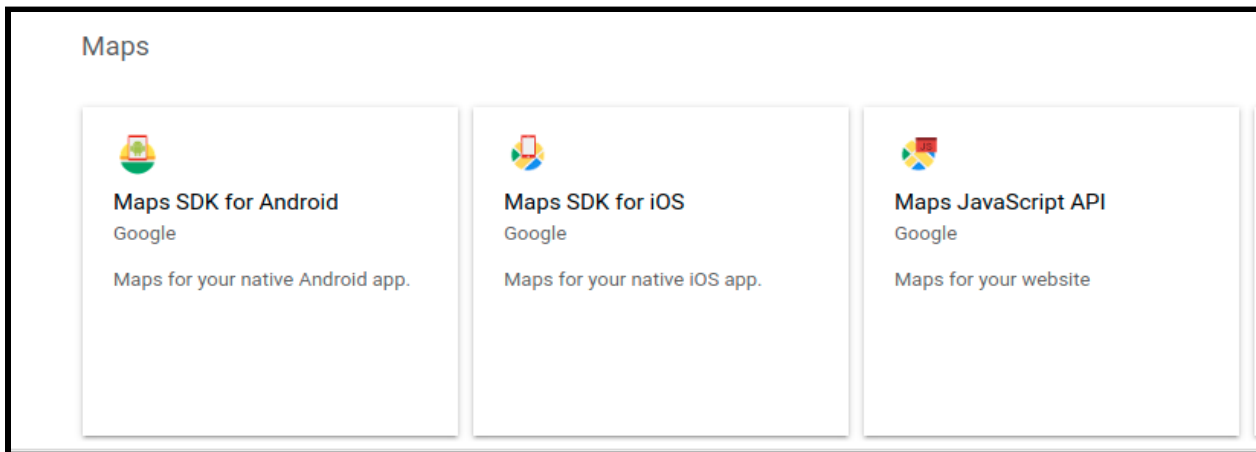
After successfully registering for a trial account you will be entitled to ~\$300 free credits that you can spend within the Google Cloud Platform (GCP). However It would recommend to set up billing by adding a valid credit / debit card.

You can create a Billing Account [here](#) and its worthing remembering that one billing account can be used across multiple GCP projects.

## 2.9.2 Enable a Google Workspace API

1. Open the [Google Cloud Console](#).
2. Next to "Google Cloud Platform," click the Down arrow and select a project.
3. In the top-left corner, click Menu > **APIs & Services**.
4. Click **Enable APIs and Services**. The **Welcome to API Library** page appears.
5. In the search field, enter the name of the API you want to enable.  
For example, type "Map API" to find the Gmail API. If you are enabling an API for a quickstart, refer to the quickstart's Prerequisites section for the API to enable.
6. Click the API to enable. The API page appears.
7. Click **Enable**. The Overview page appears.
8. To enable an additional API, repeat steps 3 - 7.

For Example:



## 2.9.3 Make these libraries enable

### 2.9.3.1 Maps SDK for IOS

- With the Maps SDK for IOS, add maps to your IOS [app](#) including [Wear OS](#) apps using Google Maps data, map displays, and map gesture responses. on web pages and mobile devices. Geolocation API
- For more detail :- <https://developers.google.com/maps/documentation/ios-sdk/overview>

### 2.9.3.2 Geocoding API

- **Geocoding** is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers on a map, or position the map.
- The Geocoding API provides a direct way to access these services via an HTTP request.
- For more detail :-  
<https://developers.google.com/maps/documentation/geocoding/overview>

### 2.9.3.3 Distance Matrix API

- The Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations.
- For more detail :-  
<https://developers.google.com/maps/documentation/distance-matrix/overview>

### 2.9.3.4 Directions API

- Provide directions for multiple transportation modes, featuring real-time traffic information.
- For more detail :- <https://developers.google.com/maps/documentation/directions>

### 2.9.3.5 Places API

- The Places API is a service that returns information about places using HTTP requests. Places are defined within this API as establishments, geographic locations, or prominent points of interest.
- For more detail :-  
<https://developers.google.com/maps/documentation/places/web-service/overview>

For more information on apis you can refer : <https://developers.google.com/maps/documentation>

## 2.9.4 Create Api key

- Go to the Google Maps Platform > Credentials page.  
[Go to the Credentials page](#)
- On the Credentials page, click Create credentials > API key.
- The API key created dialog displays your newly created API key.
- Click Close.
- The new API key is listed on the Credentials page under API keys.  
(Remember to restrict the API key before using it in production.)

- + CREATE CREDENTIALS**
  - API key**

Identifies your project using a simple API key to check quota and access
  - OAuth client ID**

Requests user consent so that your app can access the user's data.
  - Service account**

Enables server-to-server, app-level authentication using robot accounts
  - Help me choose**

Asks a few questions to help you decide which type of credential to use

- After paste this key in project constant file as below:

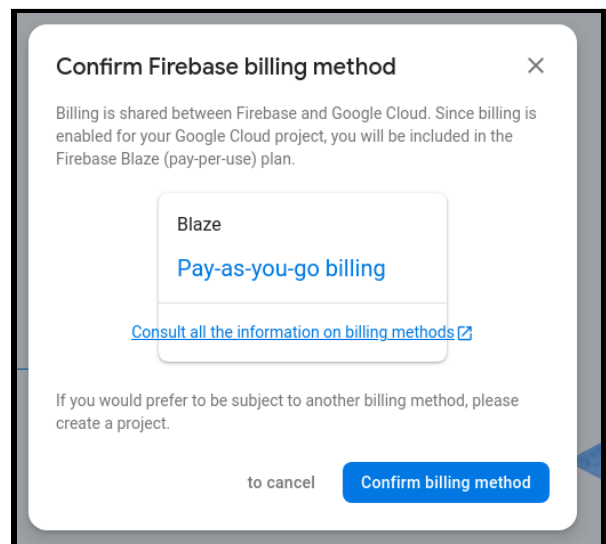
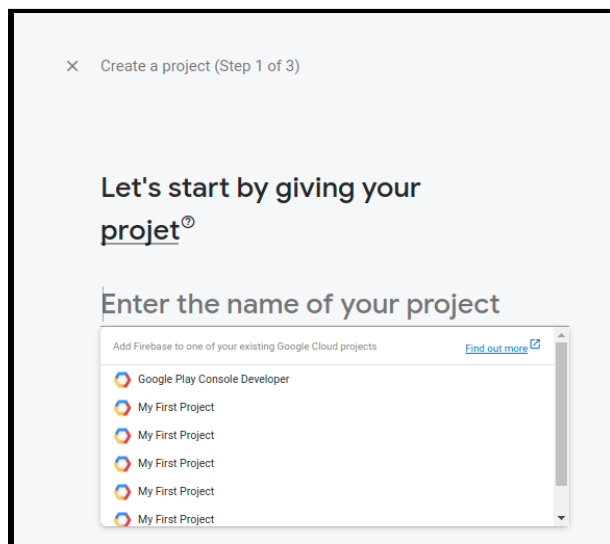
[illegible]

## 3. Firebase Account

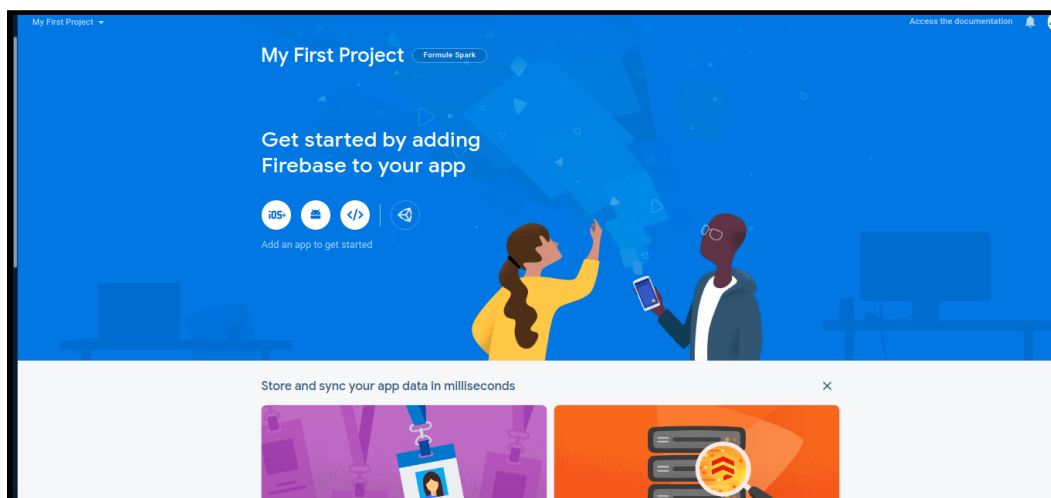
Firebase provides many utilities like cloud messaging, Crashlytics, Analytics, RealTime Databases, In-App Messaging, Dynamic Links etc. You can learn more about firebase products from <https://firebase.google.com/>

### 3.1 Create Project in FirebaseConsole

- Goto Firebase console <https://console.firebase.google.com/u/4/>
- Click Add Project
- You can see Google cloud projects you created in <https://console.cloud.google.com> here, Select Your Project and continue
- Unselect switch. You can set it up later. Continue to create project

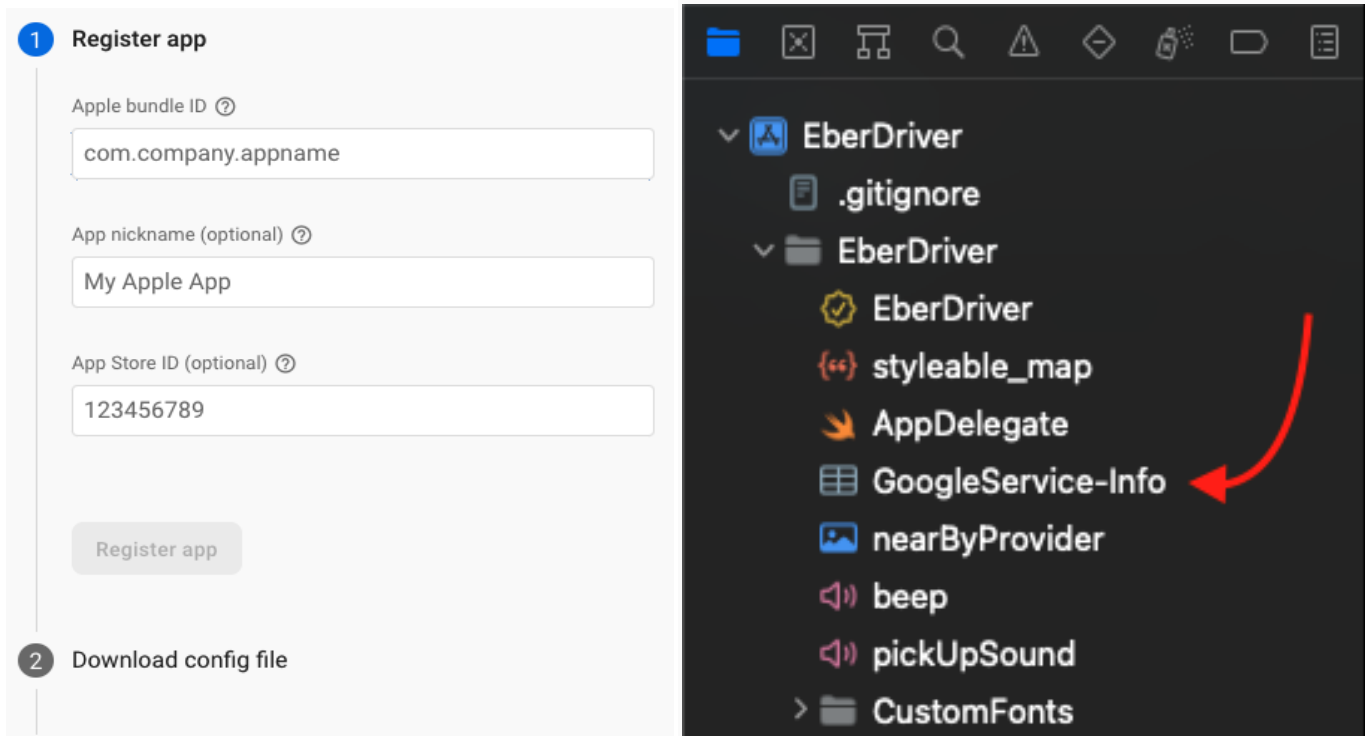


- Confirm Your Billing Method
- In the next steps, you will be asked whether to set up Google Analytics.



## 3.2 Create iOS App

- Now the Project is created. Add iOS app by clicking on iOS icon
- Add your apps package name and App Name and Bundle Identifier.



- Register your app and download **GoogleService-Info.plist** file.
- Add this **GoogleService-Info.plist** file to the root directory of your app. Refer above image.

### 3.3 Create RealTimeDatabase

Store and sync data with our NoSQL cloud database. Data is synced across all clients in real time, and remains available when your app goes offline.

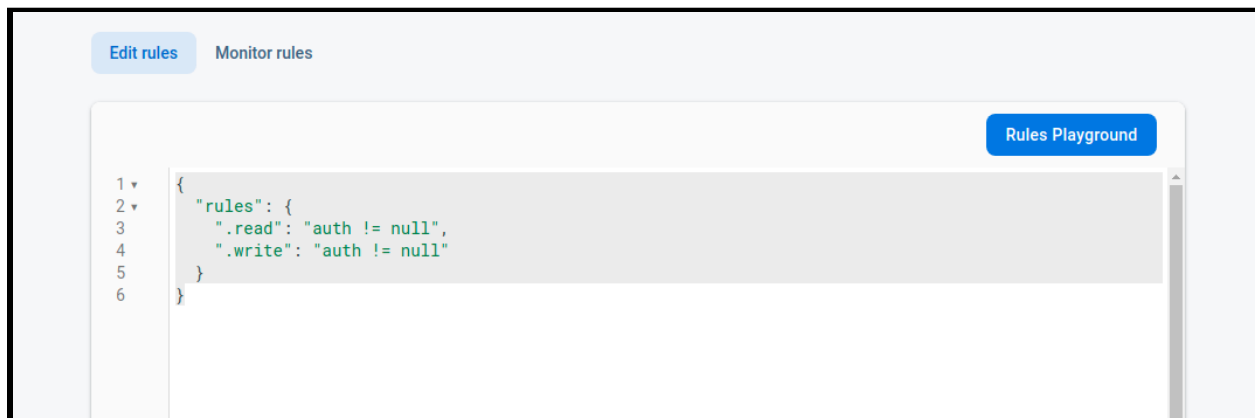
Firebase Realtime Database Security Rules determine who has read and write access to your database, how your data is structured, and what indexes exist.

For more info check this <https://firebase.google.com/docs/database>

We are using Firebase realTimeDatabase for sending, retrieving, and storing chat data.

→ GoTo Firebase console

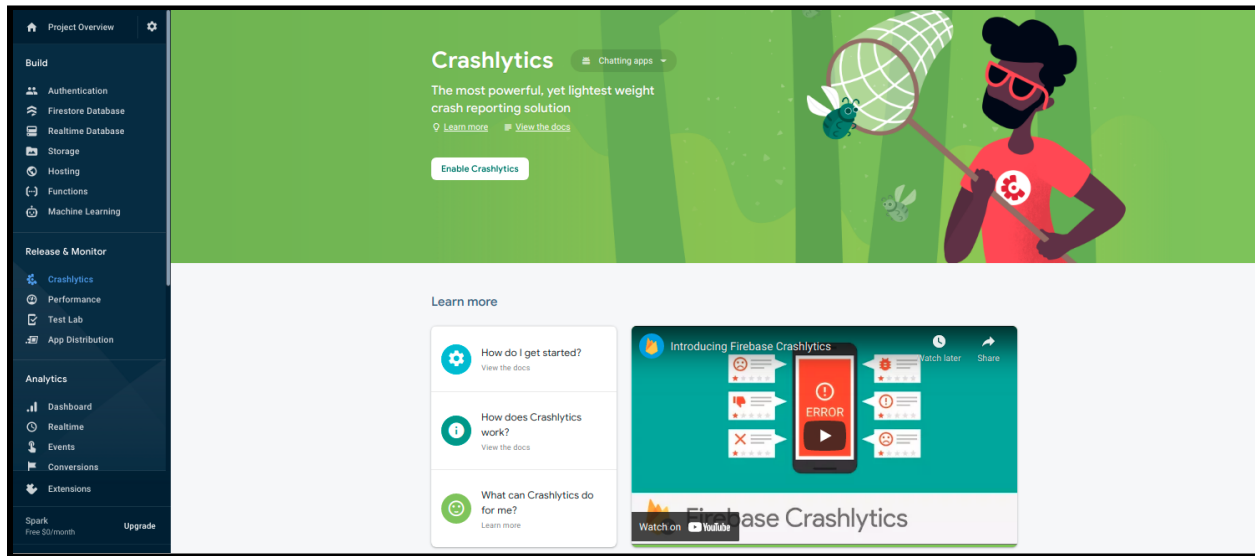
→ Side menu -> Realtime Database -> Create Database ->select locked mode -click rules -> read true and write true ->click on publish





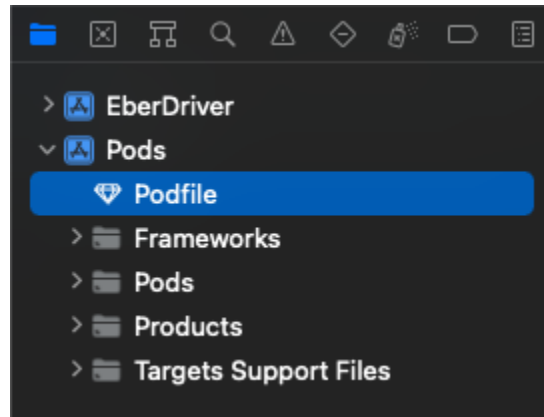
### 3.4 Enable Crashlytics

- GoTo [Firebase console](#)
- Side menu -> Crashlytics
- You can learn how to integrate crashlytics from [here](#)



## 4. Change name in pod file and install pod file (if required then)

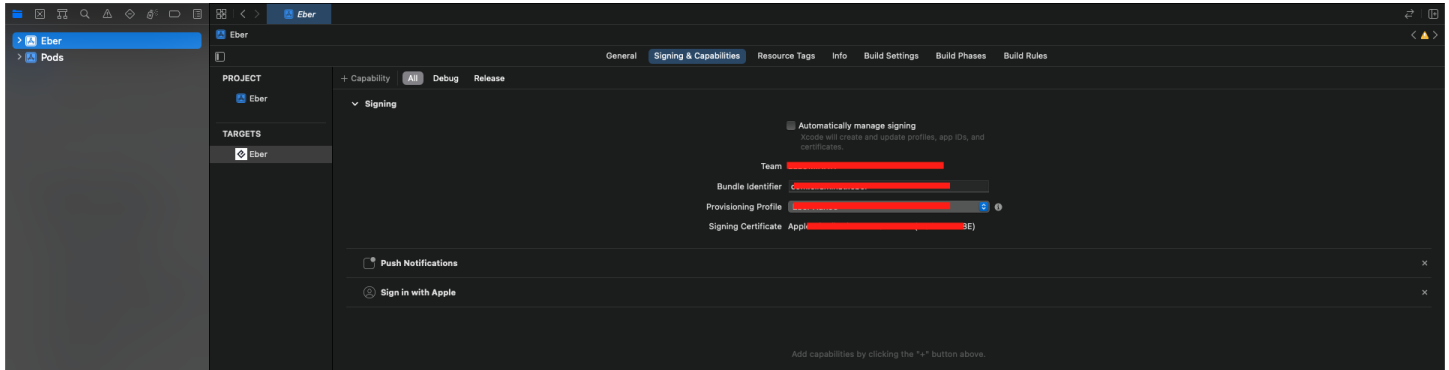
Go to your project location (in finder) where you can see the Podfile and Podfile.lock



- Open Podfile in TextEdit and Edit with your requirements. (like app name changed, add and remove framework, etc..)
- After updating the podfile successfully, save it and open the terminal.
- In the terminal, go to the path where the pod file is located and write command “**pod Install**” into terminal and enter.
- it will create a new workspace of your app name and after you have to open that workspace.
- After installing the pod file and open new workspace, you can see 2 pods file frameworks in libraries so delete the old one.

## 5. Create your app entitlements

→ Select Project → Targets → Capabilities → Push Notifications → Switch ON that. (if already open then switch off and on again)



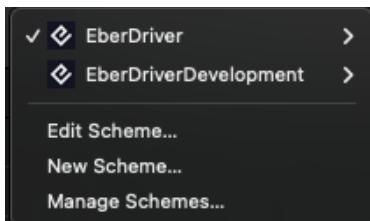
delete old (eber) entitlements from → Select Project → EberDriver → EberDriver.entitlements

It will create your new app name .entitlements file and you can put it in the same place.

## 6. Build project

Check build variants (check which have BASE\_URL)

1. Create New Schema



2. Select target device

3. Run the project