Công ty TNHH Phần Mềm Quản Lý Khách Hàng Việt Nam

Tài liệu hướng dẫn tích hợp Firebase vào app

**Tác giả:**

Lê Hoàng Mạnh

*Ngày phát hành: 24/04/2022*

*Ngày cập nhật: 26/04/202*

MỤC LỤC

[I. Cài đặt các thư viện cần thiết 3](#_Toc280829693)

[1. Dự án sử dụng npm 3](#_Toc1883077792)

[2. Dự án sử dụng yarn 3](#_Toc1421944305)

[II. Cấu hình các thư viện 3](#_Toc1419910319)

[1. Cấu hình Notifee 3](#_Toc1594445969)

[2. Tạo credentials trên Firebase console. 4](#_Toc1552453717)

[3. Cấu hình thư viện firebase và firebase messaging. 11](#_Toc163310569)

[III. Xử lý nhận và đẩy thông báo 15](#_Toc276632317)

[1. Register sự kiện để xử lý thông báo ở trạng thái nền: 15](#_Toc57256064)

[2. Yêu cầu cấp quyền để có push thông báo (chỉ dành cho IOS) 15](#_Toc229993792)

[3. Khởi tạo các hàm cần có để push được thông báo 16](#_Toc35097544)

# Cài đặt các thư viện cần thiết

## Dự án sử dụng npm

|  |
| --- |
| npm install --save add @notifee/react-native @react-native-firebase/app @react-native-firebase/messaging @react-native-firebase/remote-config @react-native-firebase/analytics |

## Dự án sử dụng yarn

|  |
| --- |
| yarn add @notifee/react-native @react-native-firebase/app @react-native-firebase/messaging @react-native-firebase/remote-config @react-native-firebase/analytics |

# Cấu hình các thư viện

## Cấu hình Notifee

\*Thư viện Notifee hiện tại đã được dùng miễn phí cho mục đích thương mại.

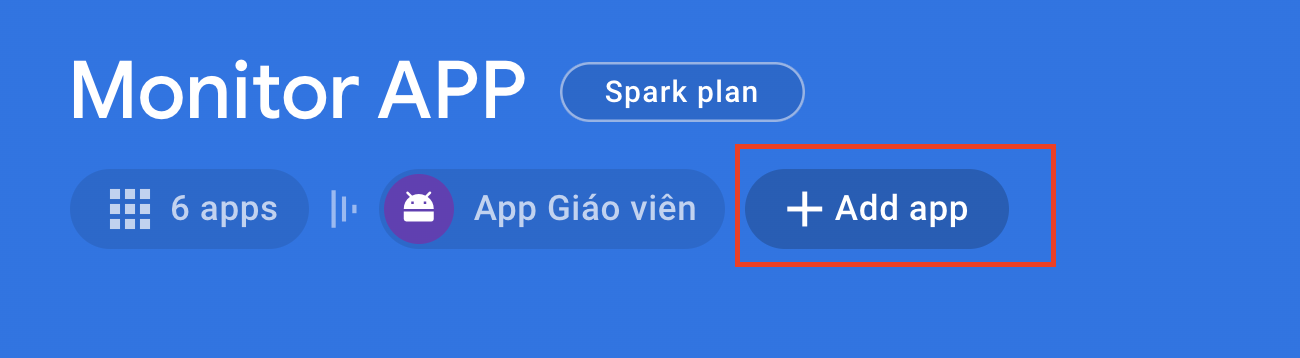
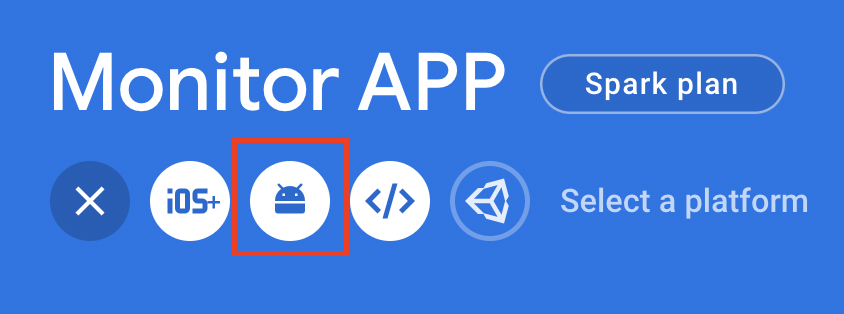
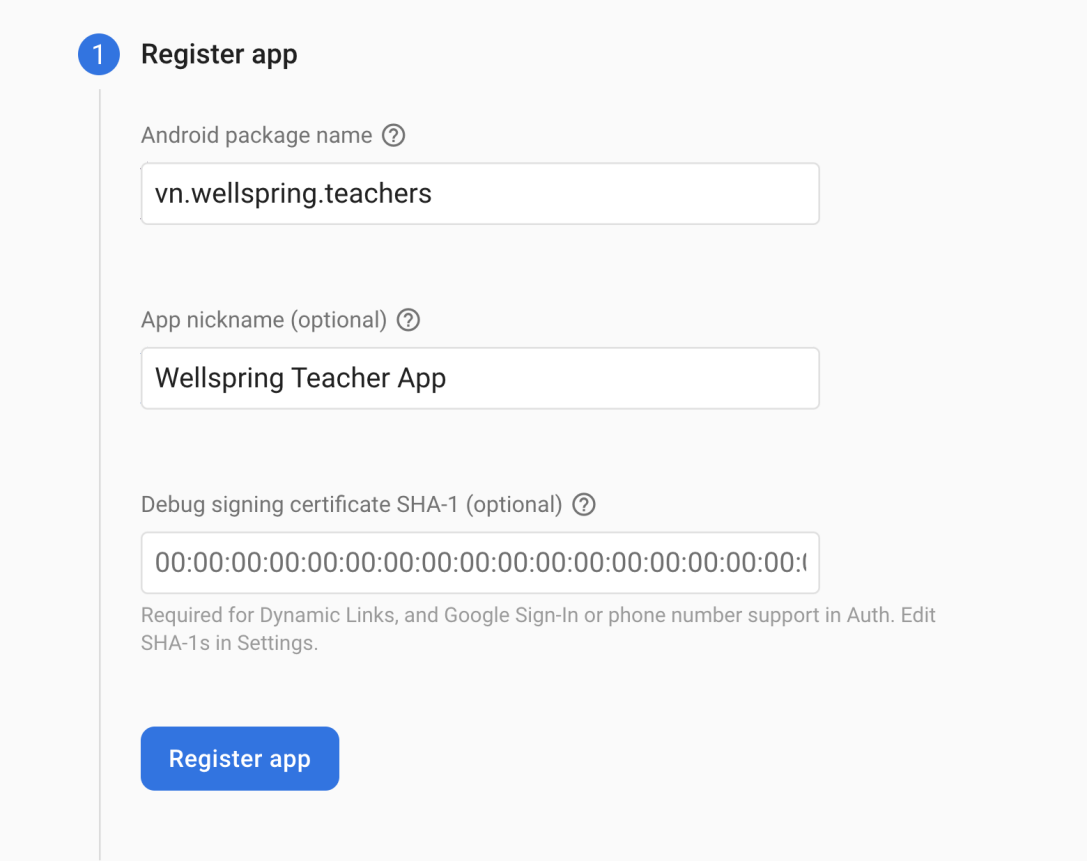
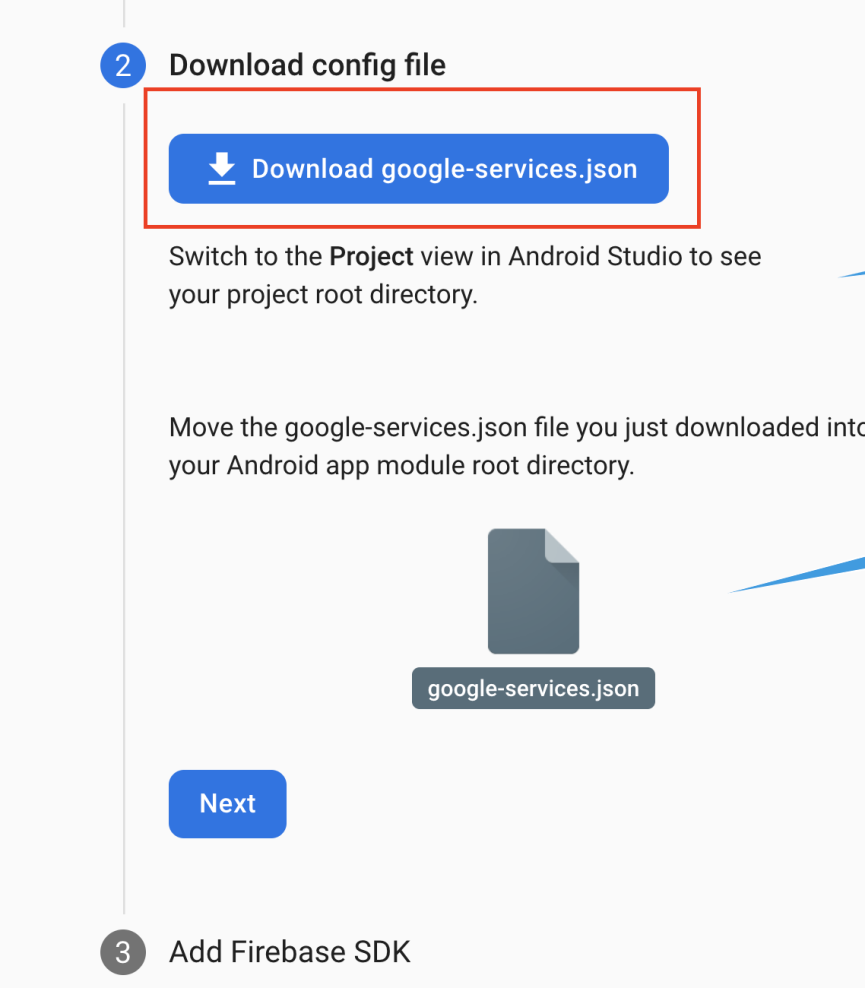
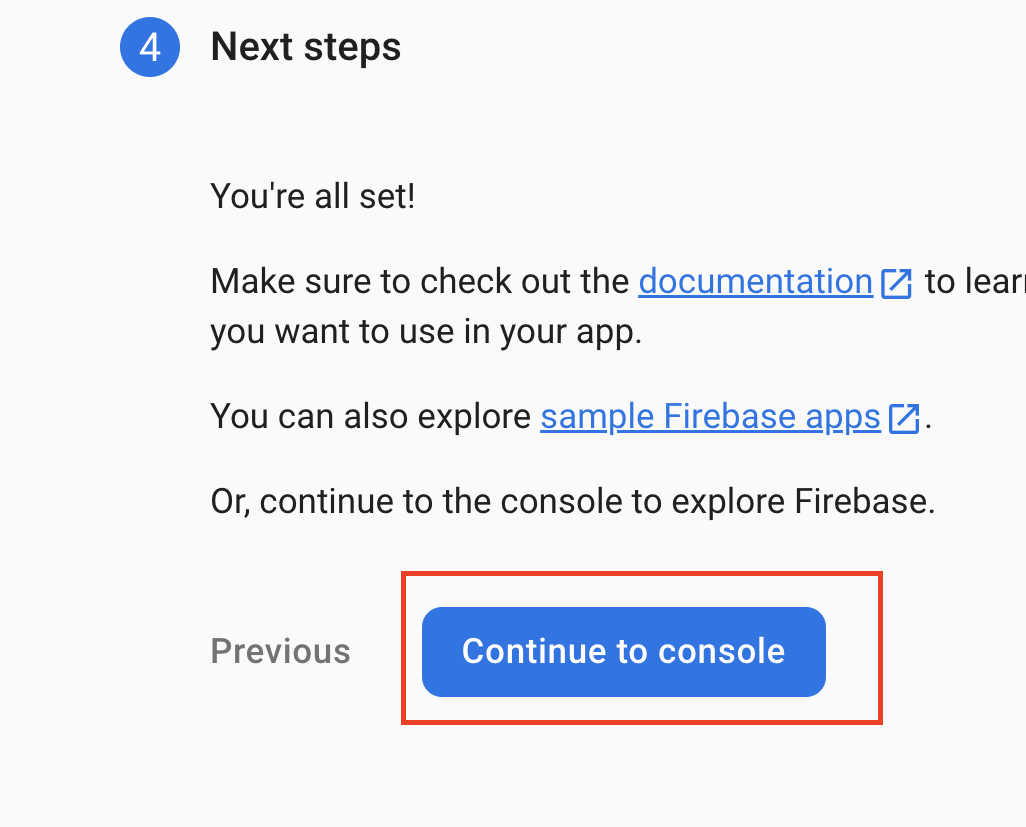
* **Cấu hình trên Android**:
* Bước 1: Add local maven repository
* Thư viện Notifee được đóng gói thành tệp “AAR” của Android và được phân phối trong “local maven repository” để tích hợp các ứng dụng Android.
* Do đó phải thêm repository vào file **android/build.gradle**, mở **android/build.gradle** lên và thêm dòng bên dưới vào **allprojects** -> **repositories**

|  |
| --- |
| // ADD THIS BLOCK - this is how Notifee finds its Android library:  maven {  url "$rootDir/../node\_modules/@notifee/react-native/android/libs"  } |

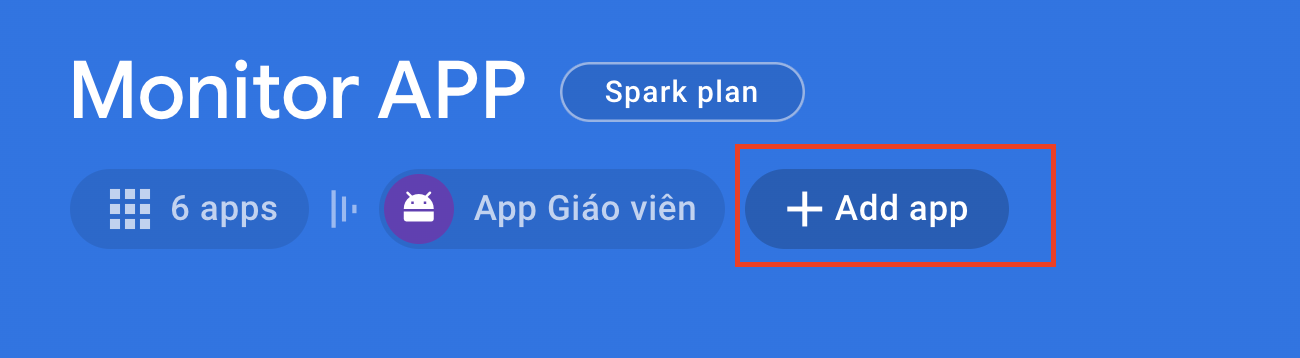
* Bước 2: Thêm đoạn code bên dưới để xử lý trường hợp người dùng nhấn vào thông báo.

|  |
| --- |
| import com.facebook.react.ReactActivityDelegate;// <-- add this line  public class MainActivity extends ReactActivity {  //...  @Override  protected String getMainComponentName() {  return NotifeeApiModule.getMainComponent("ws\_teacher\_app");  }  //....  } |

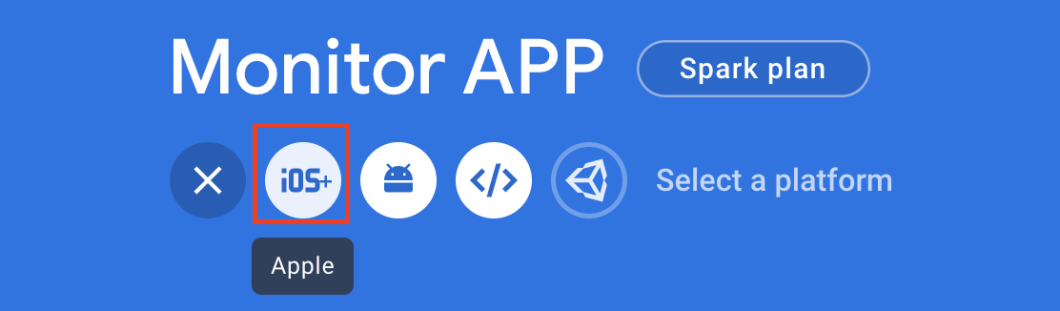
## Tạo credentials trên Firebase console.

* **Bước 1:** Truy cập vào [https://console.firebase.google.com](https://console.firebase.google.com/) và tạo mới một project hoặc chọn vào project đã có sẵn.
* **Bước 2:** Tiến hành cấu hình cho từng plaform
* Cấu hình app chạy Android
* Bước 1: Tại màn hình trang chủ nhấn “**Add app**”.
* 
* Bước 2: Chọn biểu tượng hình Android.
* 
* Bước 3: Điền các thông tin như pakckage name, tên app hiện thị và sau khi nhập xong nhấn “Register App”.
* 
* Bước 4: Tải file **google-services.json** về máy sau đó nhấn “Next”.
* 
* Bước 5: Tiếp tục nhấn “NEXT” đến bước cuối cùng nhấn “” để hoàn tất việc thiết lập thông tin app cần nhận push thông báo.
* 
* Cấu hình app chạy IOS

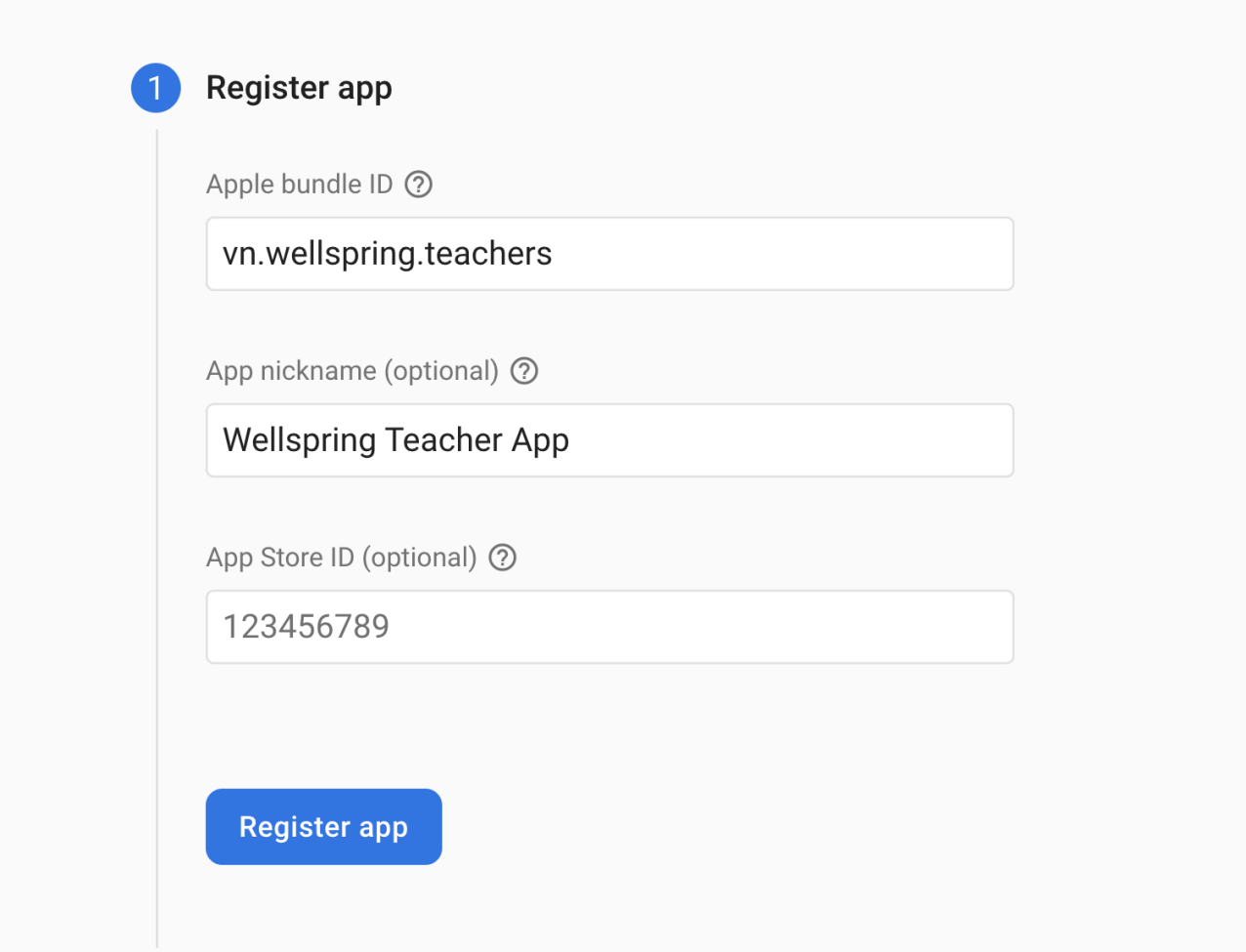
Bước 1: Tại màn hình trang chủ nhấn “**Add app**”.



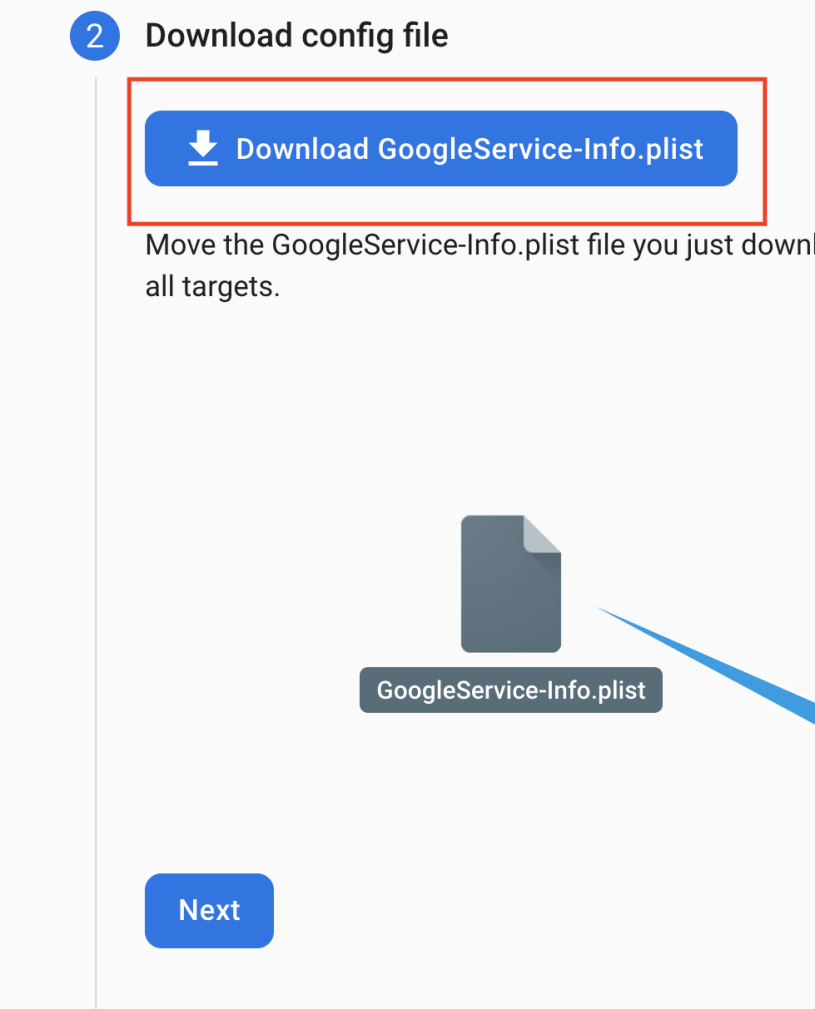
Bước 2: Chọn biểu tượng hình có chữ iOS+.



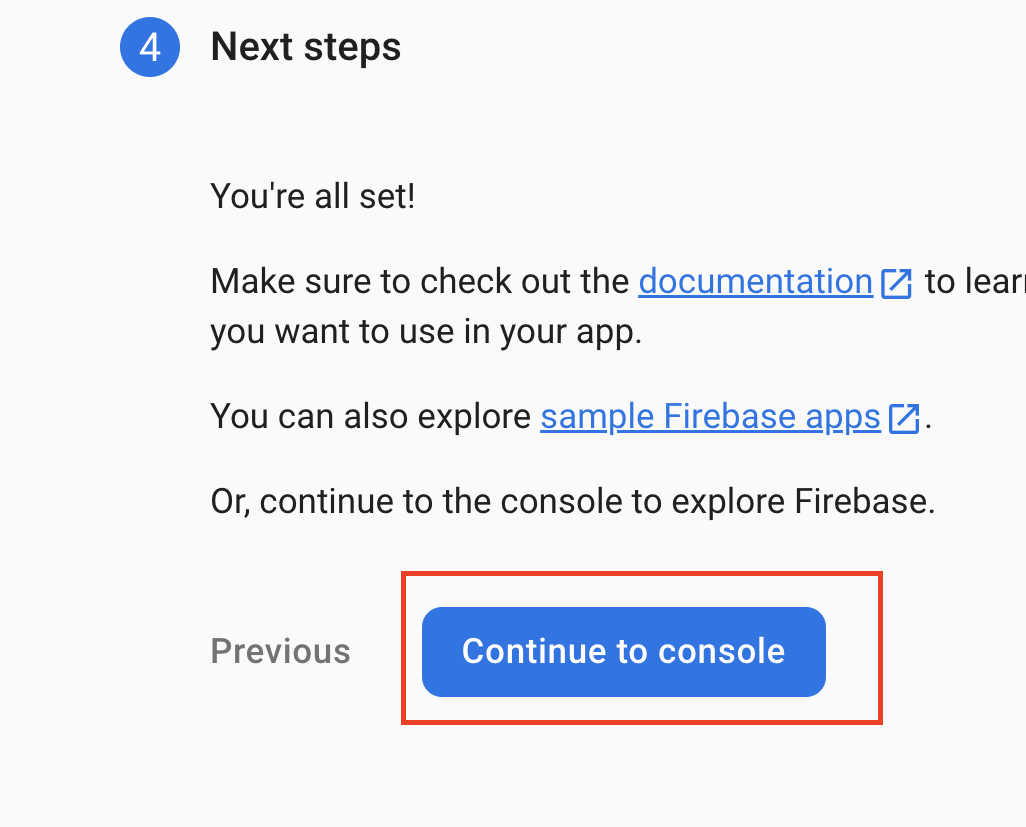
Bước 3: Điền các thông tin như Bundle ID, tên app hiện thị và sau khi nhập xong nhấn “Register App”.



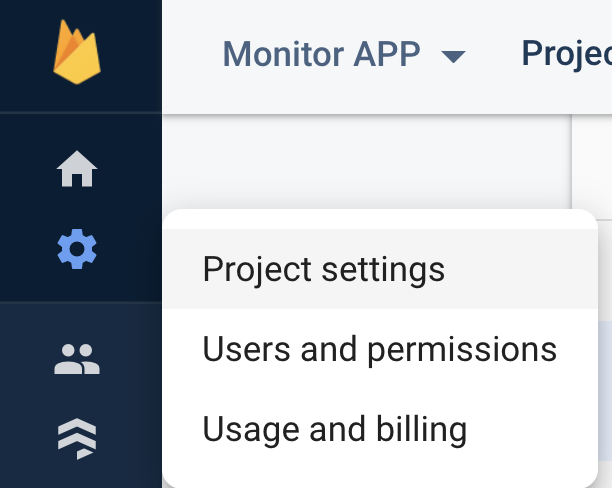
Bước 4: Tải file **GoogleServices-Info.plist** về máy sau đó nhấn “**Next**”.



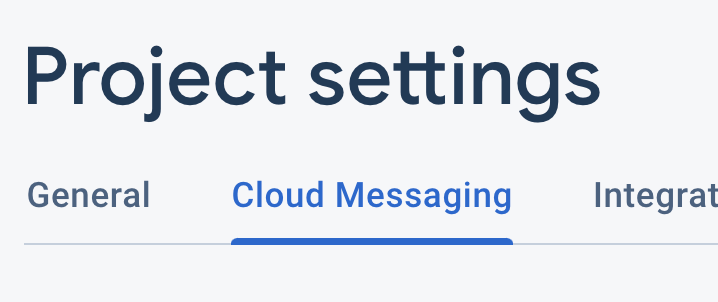
Bước 5: Tiếp tục nhấn “**NEXT**” đến bước cuối cùng nhấn “**Continue to Console**” để hoàn tất việc thiết lập thông tin app cần nhận push thông báo.



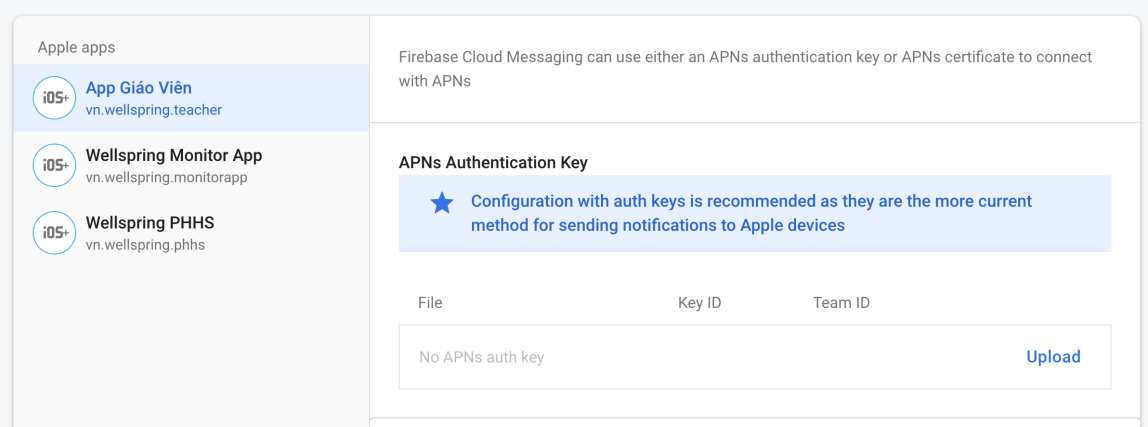
Bước 6: sau khi tạo thành công tiến hành mở cài đặt trên **Firebase Console**, chọn **Project settings**



Bước 7: chọn sang tab “**Cloud Messaging**”

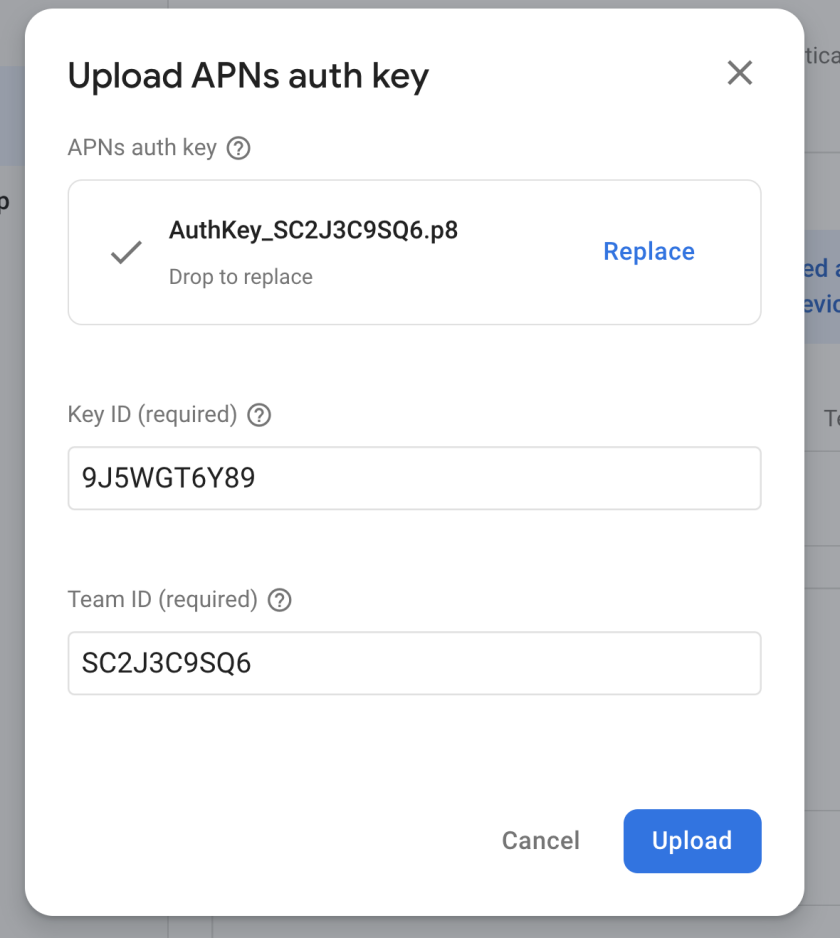


Bước 8: Chọn vào app vừa mới tạo để cập nhật thông tin APNs Authentication Key



Cách lấy thông tin TeamID và APNs Authentication có thể xem [tại đây](https://tailieu.abaha.vn/tinh-nang-khac/huong-dan-dang-ky-tai-khoan-apple-developer/untitled" \l "huong-dan-lay-teamid-va-apns-authentication-key-ios).

Bước 9: sau khi lấy được các thông tin cần thiết tiến hành upload file Authen\_key.



Bước 10: Nhấn “Upload” để cập nhật APNs và kết việc cấu hình thông tin app trên **Firebase Console.**

## Cấu hình thư viện firebase và firebase messaging.

* **Cấu hình trên project Android**.
* Copy file google-services.json theo đường dẫn “/android/app/google-services.json**”**
* Add plugin “google-services” vào bên trong **dependencies** của file /android/build.gradle.

|  |
| --- |
| buildscript {  dependencies {  // ... other dependencies  classpath 'com.google.gms:google-services:4.3.10'  // Add me --- /\  }} |

* Để chạy được plugin phải thêm dòng code sau vào trong file /android/app/build.gradle.

|  |
| --- |
| apply plugin: 'com.android.application'  apply plugin: 'com.google.gms.google-services' // <- Add this line |

* Enabling Multidex: mở file /android/app/build.gradle , dưới android -> defaultConfig thêm **multiDexEnabled true**

|  |
| --- |
| android {  defaultConfig {  // ...  multiDexEnabled true // <-- ADD THIS in the defaultConfig section  }  // ...} |

Kéo xuống phía dưới tìm **dependencies** và thêm dòng code bên dưới

|  |
| --- |
| dependencies {  implementation 'androidx.multidex:multidex:2.0.1' // <-- ADD THIS DEPENDENCY  //....  } |

* Increasing Android build memory: mở file android/gradle.properties và sau đó thêm dòng bên dưới.

|  |
| --- |
| org.gradle.jvmargs=-Xmx2048m -XX:MaxPermSize=512m -XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8 |

Add dependencies: mở file /android/app/build.gradle tìm **dependencies** và thêm các dòng bên dưới:

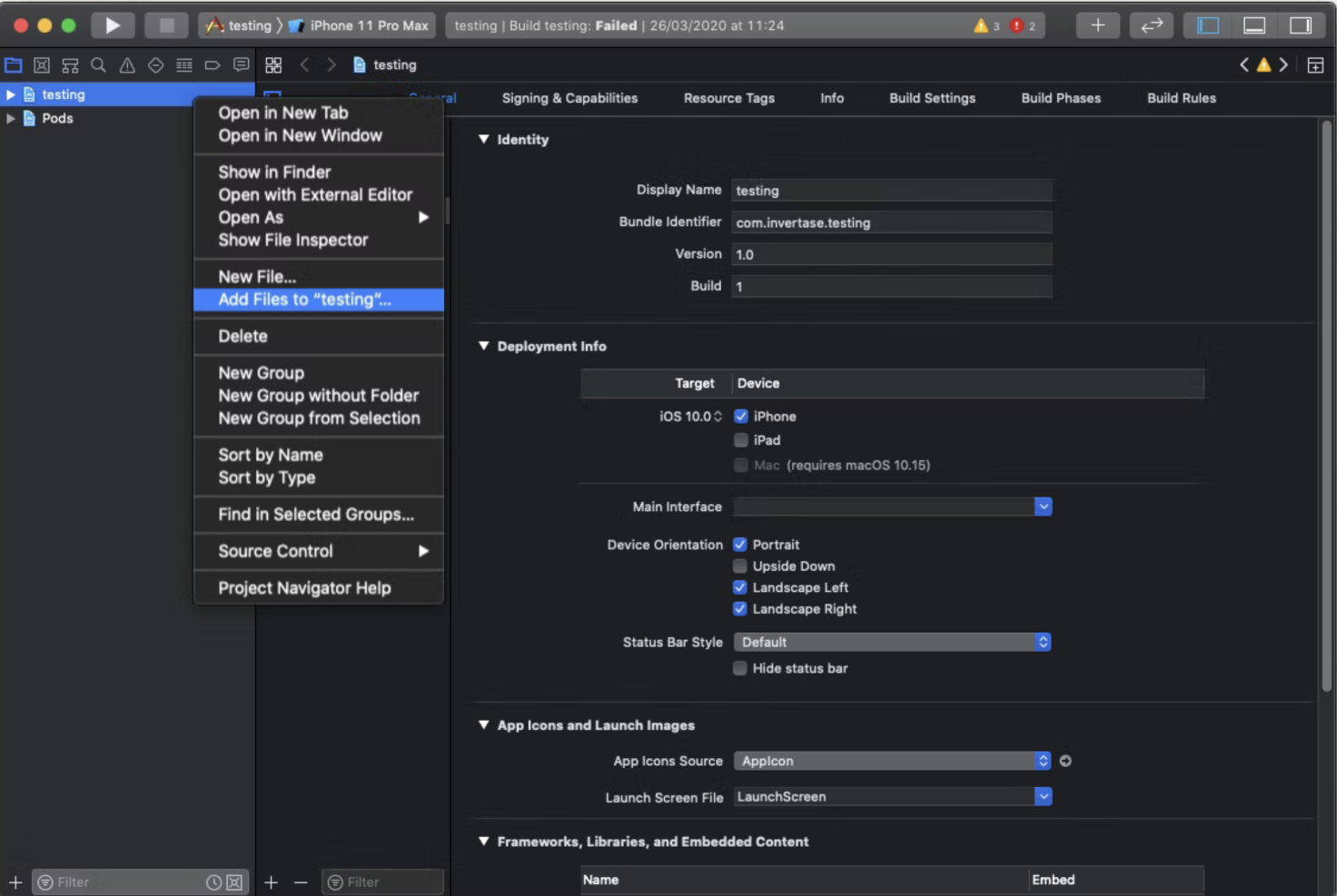
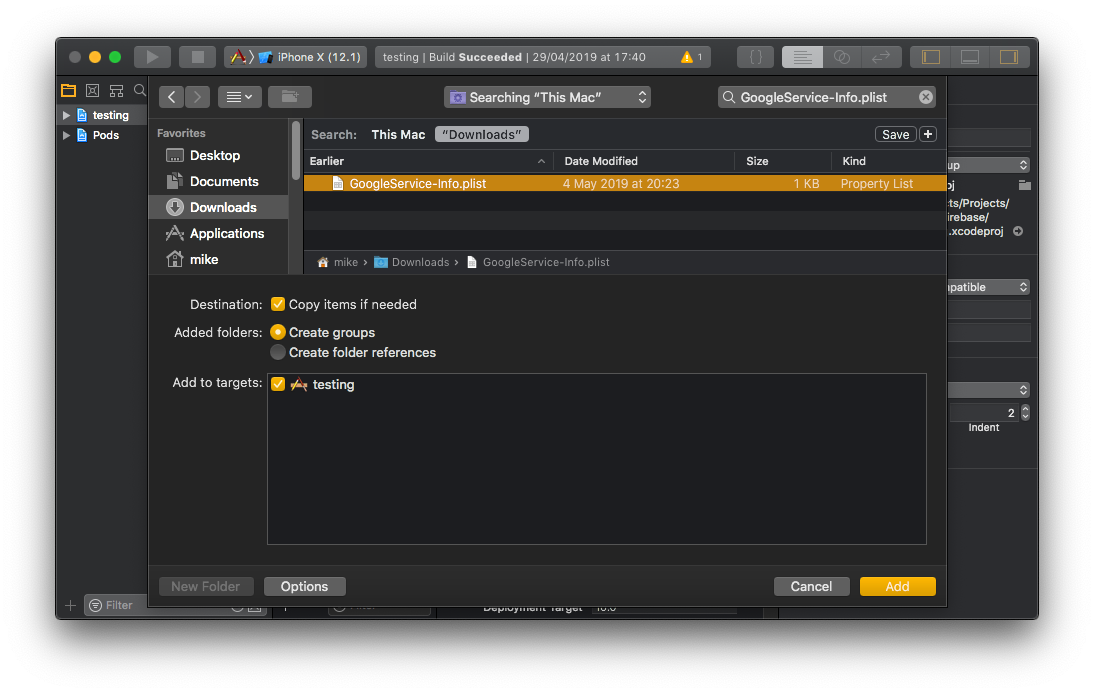
|  |
| --- |
| implementation "com.google.android.gms:play-services-base:17.0.0"  implementation "com.google.firebase:firebase-core:17.0.1"  implementation "com.google.firebase:firebase-messaging:20.1.5" |

* **Cấu hình trên project IOS.**
* Download GoogleService-Info.plist (thực hiện như các bước ở mục 2 dành cho app ios).
* Mở project /ios/Podfile và thêm đoạn code bên dưới ở đầu file.

|  |
| --- |
| # Override Firebase SDK Version  $FirebaseSDKVersion = '8.13.0' |

* Mở terminal lên, di chuyển đến thư mục chứa project và chuyển đến tiếp thư mục ios và chạy đoạn script sau:

|  |
| --- |
| pod install --repo-update |

* Dùng xcode mở project lên theo đường dẫn {projectName}/ios/{projectName}.xcodeproj Nhấp chuột phải vào tên dự án và "Add files" vào dự án, như minh họa bên dưới:
* 
* Chọn tệp GoogleService-Info.plist đã tải xuống và đảm bảo checkbox "Copy items if needed" phải được bật.
* 
* Mở file /ios/{projectName}/AppDelegate.m lên và thêm các dòng code như sau:
* At the top of the file, import the Firebase SDK:

|  |
| --- |
| #import <Firebase.h> |

* Find didFinishLaunchingWithOptions method, add the following to the top of the method:

|  |
| --- |
| - (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {  // Add me --- \/  [FIRApp configure];  // Add me --- /\  // ...  } |

* *\* Các bước cài đặt và cấu hình firebase dành cho React Native có thể xem [tại đây](https://rnfirebase.io/).*

# Xử lý nhận và đẩy thông báo

## Register sự kiện để xử lý thông báo ở trạng thái nền:

Mở fiel index.js và thêm đoạn code bên dưới nằm phía trên **AppRegistry**

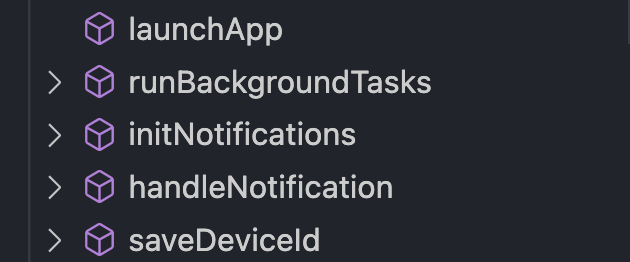
|  |
| --- |
| import messaging from '@react-native-firebase/messaging';  *// Register background handler*  messaging().setBackgroundMessageHandler(async remoteMessage => {  console.log('Message handled in the background!', remoteMessage);  }); |

# Yêu cầu cấp quyền để có push thông báo (chỉ dành cho IOS)

|  |
| --- |
| import messaging from '@react-native-firebase/messaging';  async function requestUserPermission() {  const authStatus = await messaging().requestPermission();  const enabled =  authStatus === messaging.AuthorizationStatus.AUTHORIZED ||  authStatus === messaging.AuthorizationStatus.PROVISIONAL;  return enabled;  } |

# Khởi tạo các hàm cần có để push được thông báo

1. Mở file Global.js các function như sau:



- Function launchApp sẽ được gọi sau khi đăng nhập thành công.

|  |
| --- |
| launchApp: function () {  this.initNotifications();  this.runBackgroundTasks();  } |

- Function initNotifications sẽ xử lý các sự kiện của message:

|  |
| --- |
| initNotifications: function () {  *// Gọi hàm check quyền được push thông báo*  requestUserPermission()  .then((enabled) => {  if (enabled) {  console.log('Authorization status:', enabled);  }  });  *// Just for IOS:*  *// if your app wants to receive remote messages from FCM*  *// need register to remote*  messaging().registerDeviceForRemoteMessages();  *// Lấy thông tin device token*  messaging().getToken().then((token) => {  console.warn('Device token:', token);  *// Gọi hàm lưu thông device token lên server*  Global.saveDeviceId(token);  });  *// function xử lý khi người dùng nhấn vào thông báo (khi ứng dụng đã mở)*  messaging().onNotificationOpenedApp(remoteMessage => {  console.log(  'Notification caused app to open from background state:',  remoteMessage.notification,  );  *// Gọi hàm xử lý để push thông báo*  Global.handleNotification(remoteMessage, 0, undefined);  });  *// function xử lý khi người dùng nhấn vào thông báo (khi ứng dụng đã bị kill)*  messaging()  .getInitialNotification()  .then(remoteMessage => {  if (remoteMessage) {  console.log(  'Notification caused app to open from quit state:',  remoteMessage.notification,  );  *// Gọi hàm xử lý để push thông báo*  Global.handleNotification(remoteMessage, 0, undefined);  }  });  } |

- Function handleNotification sẽ xử lý data thông báo:

\*Lưu ý: đoạn code xử lý chuyển đến màn hình theo module và api MarkNotificationAsRead sẽ phải thay đổi theo từng dự án.

|  |
| --- |
| handleNotification: function (notification, isRead, callback) {  var notify = notification;  *// Kiểm tra xem thông báo có id từ phía server trả về hay không*  *// - Nếu không thì chuyển đến màn hình thông báo*  *// - Nếu có thì xử lý chuyển đến màn hình của record module liên quan*  *// và cập nhật tình trạng là đã đọc*  if (notify?.data?.id) {  *// kiểm tra thông báo thuộc module nào thì chuyển đến màn hình chi tiết*  *// nếu không thuộc module nào thì chuyển đến màn hình thông báo*  switch (notify?.data?.category) {  case 'leaving':  Global.navigationRef?.navigate('LeavingViewScreen');  break;  default:  Global.navigationRef?.navigate('NotificationScreen');  break;  }  *// Set is read for notification clicked*  if (isRead == 0) {  var params = {  action: 'MarkNotificationAsRead',  data: {  id: notify.data.id  }  };  *// Call api*  this.callAPI(null, params, data => {  if (parseInt(data.success) === 1) {  *// call function Update counter notifications*  this.updateCounters();  *// Reset badge counter on Icon app*  setTimeout(() => {  notifee.setBadgeCount(parseInt(this.counters?.all || 0))  .then(() => console.log('Badge count set!'));  }, 2000);  *// return call back handle notification success*  callback?.(true);  }  else {  *// return call back handle notification failure*  callback?.(false);  }  },  error => {  *// return call back handle notification failure*  callback?.(false);  });  }}  else {  Global.navigationRef?.navigate('NotificationScreen');  }  } |

- Function saveDeviceId sẽ xử lý lưu thông tin device token lên server:

\*Lưu ý: api saveDeviceid sẽ có sự khác nhau theo từng dự án, vui lòng kiểm tra kỹ để tránh có lỗi.

|  |
| --- |
| saveDeviceId: function (token) {  *// Store device id on Global*  this.deviceId = token;  var params = {  action: 'AddDeviceId',  data: {  device\_id: token  }  };  *// Call api*  this.callAPI(null, params, data => {  console.log('Response save deviceId: ', data);  if (data.error == 1) {  return;  }  }, error => {  console.log('Save deviceId error: ', error);  });  } |

- Function runBackgroundTasks sẽ xử lý setInterval cứ mỗi 1 phút sẽ call API để cập nhật counter số thông báo chưa đọc và cập nhật lại số lượng ở AppIcon ngoài màn hình home.

|  |
| --- |
| const updateCountersTask = setInterval(() => {  *// Run background task only if user is logged in and network is connected*  if (this.user?.id && AppState.currentState != 'background' && this.isOnline) {  console.log('Update counters triggered');  this.updateCounters((data) => { });  }  }, 1 \* 60000);  // ghi nhận lại event và sẽ remove event khi logout  this.backgroundTasks.push(updateCountersTask); |

|  |
| --- |
| updateCounters: function (callback) {  var params = {  action: 'GetCounters',  };    this.callAPI(null, params, data => {  if (parseInt(data.success) != 1) {  return;  }  console.log('Counter data: ', data);  this.setCounters(data.data);  callback(data.data)  },  error => {  console.log('Get Counter data error: ', error);  });  } |

|  |
| --- |
| setCounters: function (counters) {  this.counters = counters;  notifee.setBadgeCount(parseInt(counters?.all || 0))  .then(() => console.log('Badge count set!'));  // Xử lý emit event thông báo lên là không còn thông báo nào chưa đọc  // để ẩn đi counter số thông báo ở BottomTab bar  if (parseInt(counters?.all || 0) > 0) {  DeviceEventEmitter.emit('Application.markUnreadNotification');  }  } |

1. Mở file xử lý điều hướng và thêm các dòng code sau ở trong hàm componentDidMount hoặc useEffect nếu dùng React Hook:

\* Lưu ý: thêm tại hàm xử lý điều hướng khi đã đăng nhập thành công

|  |
| --- |
| useEffect(() => {  *// Event được trigger khi có push message từ firebase*  const unsubscribe = messaging().onMessage(async remoteMessage => {  console.log('A new FCM message arrived!', JSON.stringify(remoteMessage));  onDisplayNotification(remoteMessage)  });  *// Event của Notifee*  *// Event được trigger xử lý khi app đã mở và user nhấn vào các action trên thông báo (nếu có)*  const unsubscribeForeground = notifee.onForegroundEvent(  ({ type, detail }) => {  switch (type) {  case EventType.DISMISSED:  console.log('User dismissed notification', detail.notification);  break;  case EventType.PRESS:  console.log('User pressed notification', JSON.parse(detail.notification?.data?.data || '{}'));  Global.handleNotification(  detail.notification,  0,  () => {}  );  break;  }  });  return () => {  unsubscribe();  unsubscribeForeground();  };  }, []); |

|  |
| --- |
| *// Hàm xử lý show thông báo*  async function onDisplayNotification(notificationData: any) {  *// Create a channel*  const channelId = await notifee.createChannel({  id: 'wellspringteacher',  name: 'Wellspring Teacher Channel',  });  *// Display a notification by notifee*  await notifee.displayNotification({  title: notificationData?.notification?.title ? notificationData?.notification?.title : 'New notification!',  body: notificationData?.notification?.body || '',  android: {  channelId,  smallIcon: 'name-of-a-small-icon', *// optional, defaults to 'ic\_launcher'.*  sound: 'default'  },  ios: {  sound: 'default'  },  data: {  data: JSON.stringify(notificationData || {})  }  });  Global.updateCounters();  } |