

# Welcome

---

Thank you for your purchase! If you have any questions that you feel should have been in this document, please feel free to contact us at [\*\*support@wicombit.com\*\*](mailto:support@wicombit.com)

- Requirements
- Setting up the Backend
- Setting up the App
- Customization
- Build Standalone App
- Copyright & Credits
- Technical Support

# Requirements

---

## System Requirements

1. Node.js. [Download Here](#)
2. Yarn. [Download Here](#)
3. React Native CLI. Install using **npm install -g react-native-cli**
4. Expo CLI. Install using **npm install -g expo-cli**

## App Requirements

1. Firebase Account (**Required**) [Create Here](#)
2. OneSignal (Optional) [Create Here](#)

## Backend Requirements

1. PHP: **Hosting PHP 8.0.**
2. Extensions: **cURL, OpenSSL, mbstring, MySQLi, imagecreate**
3. Database: **MySQL 5.7.3+ or MariaDB.**
4. Server: **Apache**

# Setting up the Backend

---

## Backend Setup

1. Unzip the downloaded package folder and locate the "Backend" folder.
2. Copy the entire "Backend" folder content and place it into the root directory of your server. This is usually the "public\_html" folder for shared hosting or the "www" folder for a local server.
3. Open your phpMyAdmin control panel and select your current database. If you do not have a database yet, you can create one and name it.
4. Once you have selected your database, click on the "Import" tab located at the top of the page.
5. In the "File to import" section, click on the "Choose File" button and select the "**Database.sql**" file from the unzipped "Backend" folder and click on the "Go" button to import the database.
6. After the import is completed, locate the "**admin**" folder open the "**config.php**" file using a text editor.
7. Change the following details to match your database credentials:

```
$database = array(  
'host' => 'DATABASE_HOST_HERE',  
'db' => 'DATABASE_NAME_HERE',  
'user' => 'DATABASE_USER_HERE',  
'pass' => 'DATABASE_PASSWORD_HERE'  
);
```

8. Use the default login details to access the admin panel:

Username: **admin@admin.com**

Password: **123456**

## Firebase Setup

1. Create a Firebase project  
Before you can use Firebase in your project, you need to create a Firebase project. You can do this by going to the Firebase website and clicking on "Get Started" or "Go to Console". Then, follow the prompts to create a new project.
2. Get your Firebase URL. Once you have created a Firebase project, you will be redirected to the Firebase console. On the project overview page, you will see a "Project settings" button in the top right corner. Click on it.
3. On the project settings page, you will see a section called "Your apps". Find the app you want to add the Firebase URL to and click on the "Web" icon (the one with "</>"). A pop-up will appear with your

Firestore configuration. In this configuration, you will see the Firestore URL under the "databaseURL" key. Copy this URL. In your project, open the **"admin/config.php"**. Replace "YOUR\_FIREBASE\_URL\_HERE" with the URL you copied in the previous step. Make sure to enclose the URL in single quotes ("").

```
define (FIREBASE_URL, 'YOUR_FIREBASE_URL_HERE');
```

# Setting up the App

---

## 1. Firebase Setup

To add Firebase to your app, you'll need a Firebase project and a short snippet of initialization code that has a few details about your project.

1. Sign in to Firebase.
2. Go to <https://console.firebase.google.com> and create a project.
3. Go to "Authentication/Sign-in method" and enable "Email/Password".
4. Go to "Project Settings", add a Web app to your project. Follow the assistant, and copy Firebase configuration code and paste it on **src/config/ConfigFirebase.js**.
5. Open [https://console.firebase.google.com/project/\\_/settings/serviceaccounts/adminsdk](https://console.firebase.google.com/project/_/settings/serviceaccounts/adminsdk) and select the project you want to generate a private key file for.
6. Click Generate New Private Key, then confirm by clicking Generate Key and download the generated **google-service-account.json** file and place it in the **/classes/firebase** folder of your server.

```
apiKey: "",
authDomain: "",
databaseURL: "",
projectId: "",
storageBucket: "",
messagingSenderId: "",
appId: ""
```

## 2. OneSignal Setup

1. First, you need to create an OneSignal account if you haven't already.
2. Create an app: Once you're signed in to your OneSignal account, click on the "Add App" button and select "Create a new mobile & web app".
3. Setup your app platform: Select the platform for your app (iOS, Android, or web) and enter the app name. Select "Next" to continue.
4. Configure your app: Fill in the necessary information for your app. Once you're done, click on "Save" and then "Next".
5. On the next page, you will see the instructions for adding the OneSignal SDK to your app. Follow these instructions carefully as they will vary depending on your platform and development environment. Copy OneSignal API code and paste it on **src/config/ConfigApp.js**.

## Customization

---

### 1. Change **App Color**

Open **src/config/ColorsApp.js** and update this value.

```
const ColorsApp = {
  PRIMARY: "#2472d4",
  SECOND: "#2472d4",
};
```

### 2. Change **Theme Mode**

Open **src/config/ConfigApp.js** and update this value to “light” or “dark”.

```
const ConfigApp = {
  THEMEMODE: "light", // light or dark
}
```

### 3. Change **App Name**

Open **app.json** and replace this value:

```
{
  "expo": {
    "name": "Your App Name"
  }
}
```

### 4. Change **Texts**

Open **src/languages/en.json** and replace these values:

```
{
  ST0: "Home",
  ST1: "Sign In",
  ST2: "Register",
  ST3: "Favorites",
  ...
}
```

### 5. Change **App Icon** and **SplashScreen**

You just have to replace the images located in the **assets** folder.

## 1. Testing App ( EXPO GO )

**If you plan on using Admob, please choose the second option. Admob contains native code that is not compatible with Expo GO, and using it will result in an error.**

1. Navigate to your project directory: Once the project is created, navigate to the project directory by running the following command.

```
cd [project name]
```

2. Start the development server: Run the following command to start the development server.

```
npx expo start
```

This will start the Expo CLI server and a browser window will open with a QR code.

3. Install the Expo app on your device: To run your app on your device, you need to install the Expo app. You can download it from the App Store for IOS or Google Play Store for Android.
4. Scan the QR code: Open the Expo app on your device and scan the QR code from the browser window. This will build and run your app on your device.
5. Test your app: You can now test your app on your device. Any changes you make to your code will automatically be updated on your device.

## 2. Testing App ( Development Client )

### Android

Run the following command to get the Android development build started on EAS:

```
eas build --profile development --platform android
```

After that, Expo will start your build, and in a few minutes, your APK will be ready to install. You can either scan a QR code, go to a URL to download the APK, or manually download and send the APK to your Android device.

### iOS

Run the following command to get the iOS development build started on EAS:

```
eas build --profile development --platform ios
```

Just like with the Android build, this command will present you with a few prompts before the build starts. These prompts will ask for the bundle identifier, followed by a few questions that are needed for code signing. The code signing prompts will ask if you have an existing certificate and provisioning profile that you'd like to use for signing. If not, you can enter your Apple ID credentials, and EAS will create a provisioning profile for you, as well as register your devices to the provisioning profile.

Once you've entered this information, the build will start. When it's finished, you'll receive instructions for installing the build on your device.

### **Native app testing**

Now that Expo has built our native apps for us, we can begin testing our React Native app. We'll do this by running a local development server that our native apps can connect to. That command is:

```
expo start --dev-client
```

When you open either the Android or iOS app, you'll be presented with a screen to connect to the local development server. Your phone may find the local development server automatically

## Build Standalone App

---

Please follow this document for detail clarify how to build the standalone app and submit to App Store and Google Play: <https://docs.expo.dev/archive/classic-updates/building-standalone-apps/>

Open the **App.json** and add config for **package id**:

```
"ios": {  
  "supportsTablet": true,  
  "bundleIdentifier": "com.yourcompany.appname"  
},  
"android": {  
  "package": "com.yourcompany.appname"  
}
```

# Copyright & Credits

---

## Resources Used

- [React Native](#)
- [Expo.io](#)
- [Firebase](#)
- [React Native Paper](#)
- [OneSignal](#)

## Technical Support

---

Thank you again for purchasing this product. If you have any questions that are beyond the scope of this help file, please feel free to send an email to [support@wicombit.com](mailto:support@wicombit.com).

*Wicombit Team.*

Email: [support@wicombit.com](mailto:support@wicombit.com)