

Push Notifications for Web,PWA,Native Mobile

Introduction

This document explains the configuration process for enabling **Push Notifications** in a Mendix application across **Web, PWA, and Native Mobile** platforms.

The goal of this setup is to deliver **real-time alerts** and enhance **user engagement** through instant notifications. The implementation uses **Firebase Cloud Messaging (FCM)** to send and receive messages efficiently.

This guide covers:

- Firebase setup and VAPID key integration
- Push notification registration in Mendix
- Testing notifications across Web, Mobile, and PWA
- Common errors and their resolutions

Part 1: Add Module Dependencies

This section outlines the steps to add the required dependencies for the **Push Notifications Connector** in the Mendix app.^[1]

Only essential modules were installed — others were skipped since they already existed in the project.

1. Encryption Module

- Navigate to Marketplace → Search “Encryption” → Download.

Follow the setup instructions provided in the Marketplace documentation.

2. Community Commons

- Navigate to Marketplace → Search “Community Commons” → Download.
- No additional configuration is required.

3. Nanoflow Commons

- Navigate to Marketplace → Search “Nanoflow Commons” → Download.
- Ready to use after installation.

4. Native Mobile Resources

- Navigate to Marketplace → Search “Native Mobile Resources” → Download.
- Assign the NativeMobileResources.User role to users who will access notifications.

5. Atlas Core

- Navigate to Marketplace → Search “Atlas Core” → Download.
- No further setup needed.

6. Data Widgets

- Navigate to Marketplace → Search “Data Widgets” → Download.

7. Pop-Up Menu Widget

- Navigate to Marketplace → Search “Pop-Up Menu” → Download.

8. Combo Box Widget

- Navigate to Marketplace → Search “Combo Box” → Download.

9. Switch Widget

- Navigate to Marketplace → Search “Switch” → Download.

Part 2: Implement the Push Notifications Module

After installing all prerequisites, proceed with implementing the **Push Notifications Connector** to enable push functionality in the Mendix app.

Download the Module

1. Open **Mendix Marketplace** from **Studio Pro**.
2. Search for “**Push Notifications Connector**.”
3. Open the module and click **Download**.
4. Once downloaded, the module will appear under **App Explorer** → **Modules**.

Part 3: Set Up Google Firebase Cloud Messaging (FCM)

Server

Introduction

Firebase Cloud Messaging (FCM) is a cloud-based service provided by Google that enables you to send push notifications across multiple platforms including Web, Android, and iOS in a reliable and scalable way.

In our Mendix application, we integrate FCM to handle all communication between the Push Notification Connector (server) and the user’s device (Web, PWA, or Mobile app).

Why We Use Firebase (FCM)

1. Cross-Platform Support:

FCM allows us to send notifications to **both web browsers and mobile devices** using a single setup.

2. Real-Time Message Delivery

It ensures instant delivery of messages and alerts, even when the app is closed or running in the background.

3. **Secure Communication:**

Firebase uses unique authentication tokens (like the **VAPID key** and **Server Key**) to securely identify your app and devices.

4. **Scalability and Reliability:**

It can handle thousands of devices at once, ensuring stable delivery without needing your own notification server.

5. **Integration with Mendix:**

Mendix’s **Push Notification Connector** uses Firebase as the backend service to register devices, generate tokens, and send notifications automatically.

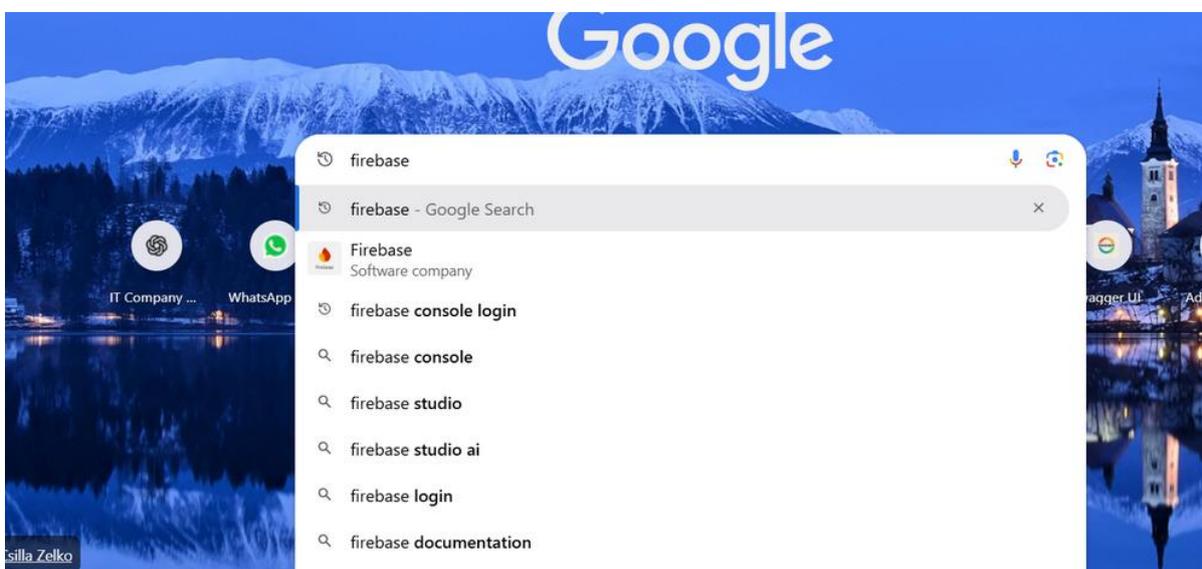
What We Do in This Step

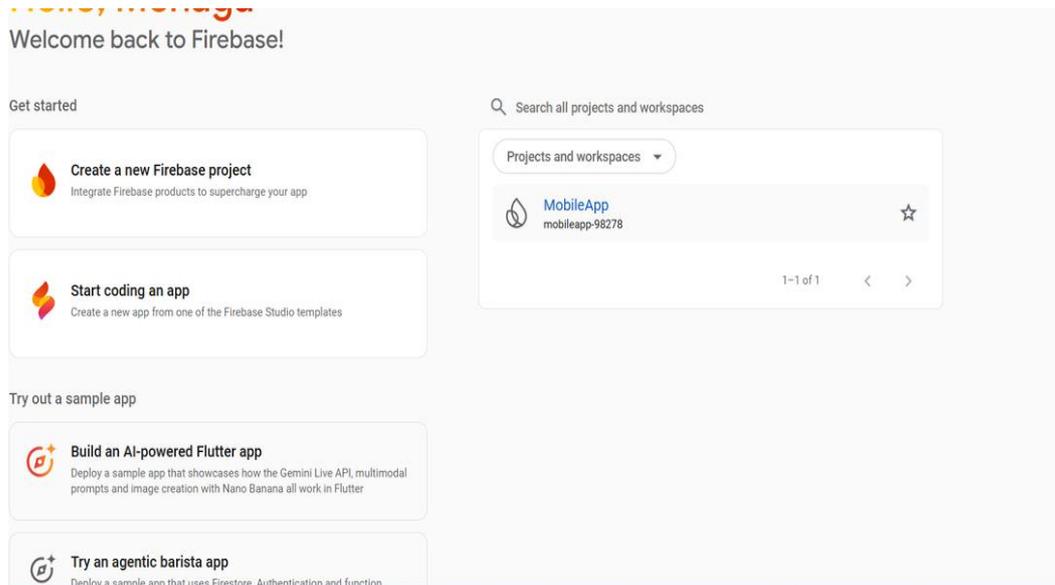
Step 1: Access Firebase Console

Open your web browser and Search Firebase and click First Link then go to the **Firebase Console** by searching for “Firebase” on Google and selecting the official Firebase website.

Step 2: Create a New Firebase Project In the Firebase Console, click “**Add Project**” or “**Create a Project.**”

Enter a suitable project name (for example, **PushNotificationApp**) and click **Continue** to proceed.



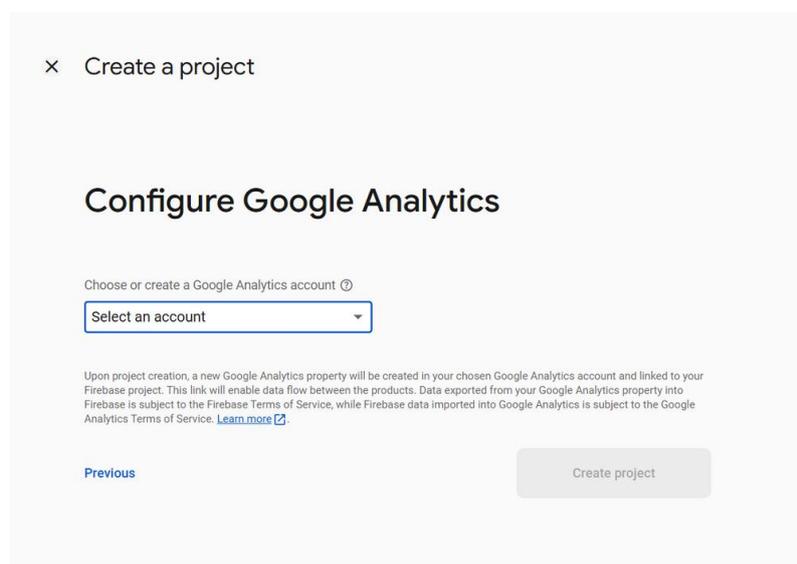


Step 3: Configure Google Analytics (Optional)

In the next screen, you will be prompted to configure **Google Analytics** for your Firebase project.

- If you want to track app usage and engagement, select an existing Google Analytics account or create a new one from the dropdown menu.
- If analytics is not required, you can disable it by toggling **“Enable Google Analytics for this project”** off.

Once configured, click **Create Project** to complete the Firebase project setup.



Step 4: Firebase Project Dashboard

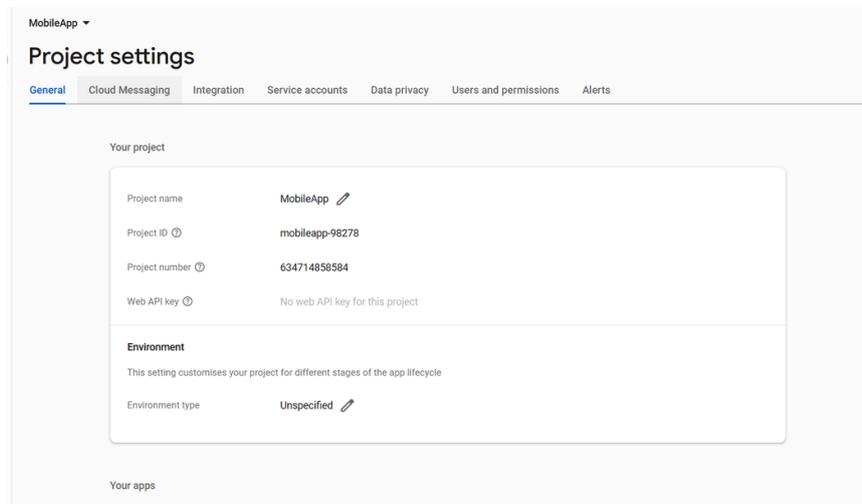
After creating your project, you'll reach the **Firestore Project Overview** page

Here, you can manage services like Authentication, FCM, and Analytics.

Step 5: Open Project Settings

In the **Firestore Dashboard**, locate the **gear (⚙️) icon** next to *Project Overview* on the left sidebar.

Click the icon and select **“Project settings”** from the dropdown menu.



Step 6: View Project Details

In the **Project Settings** page, under the **General** tab, you can view key project details such as:

- **Project Name**
- **Project ID**
- **Project Number**
- **Web API Key**

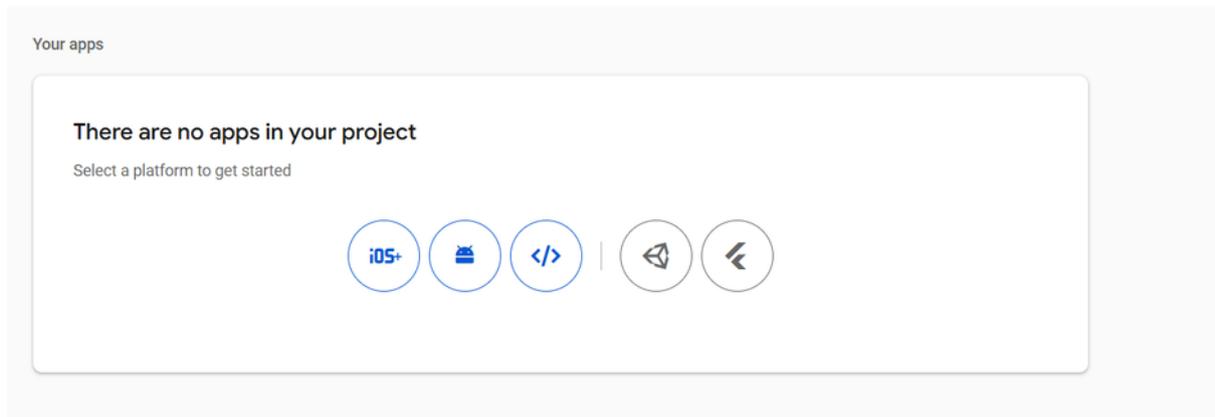
These values will be required later for integrating Firebase with your Mendix app for push notifications.

Step 7: Add Your App to Firebase

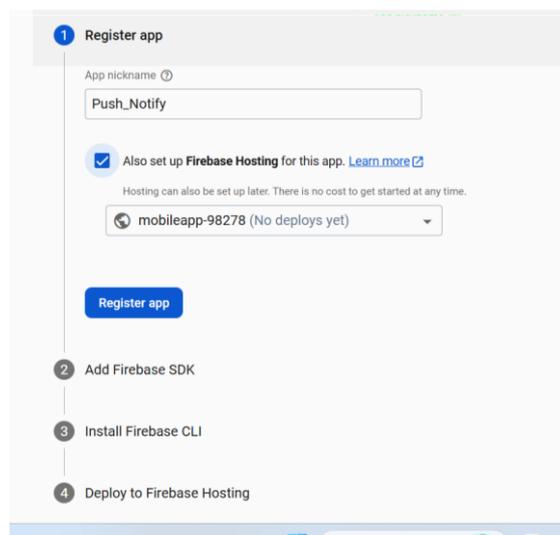
Scroll down to the **Your apps** section.

Click on your target platform — **iOS**, **Android**, or **Web** — to start configuring Firebase for that specific app.

This step links your Mendix application with Firebase, enabling push notification functionality.



- Click the **Web icon** (</>) under “Your apps.”
- In the “App nickname” field, enter: **Push_Notify**
- Then click **Register app** to continue.



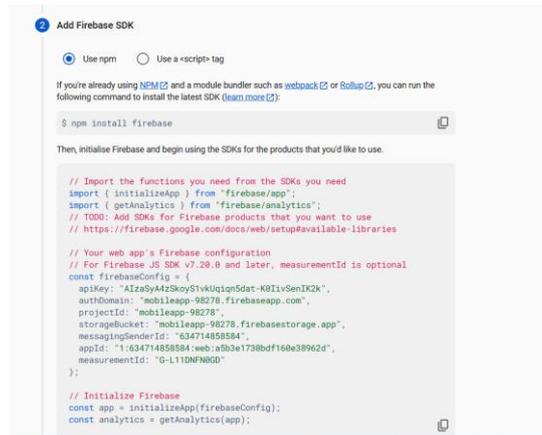
Step 9: Add Firebase SDK

This step is essential as it provides the configuration code required to connect your Firebase project with your web application.

You will use this configuration while creating your index.html and JavaScript (app.js) files.

- Copy the provided Firebase SDK configuration and initialization code.
- This code ensures that your web app can communicate with Firebase services like Authentication, Cloud Messaging, and Analytics.

Note: Keep your Firebase configuration safe — it links your project to the Firebase backend.



```
2 Add Firebase SDK
 Use npm  Use a <script>-tag

If you're already using NPM and a module bundler such as webpack or Rollup, you can run the
following command to install the latest SDK (learn more)

$ npm install firebase

Then, initialise Firebase and begin using the SDKs for the products that you'd like to use.

// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyA4zSkoyStvkiqig5dat-K8IivSenIK2k",
  authDomain: "mobileapp-98278.firebaseio.com",
  projectId: "mobileapp-98278",
  storageBucket: "mobileapp-98278.firebaseio.com",
  messagingSenderId: "634714858584",
  appId: "1:634714858584:web:1303e1738bdf160e38962d",
  measurementId: "G-L11DNFN6GD"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Step 10: Install Firebase CLI

This step is important for **deploying your web app** to Firebase Hosting.

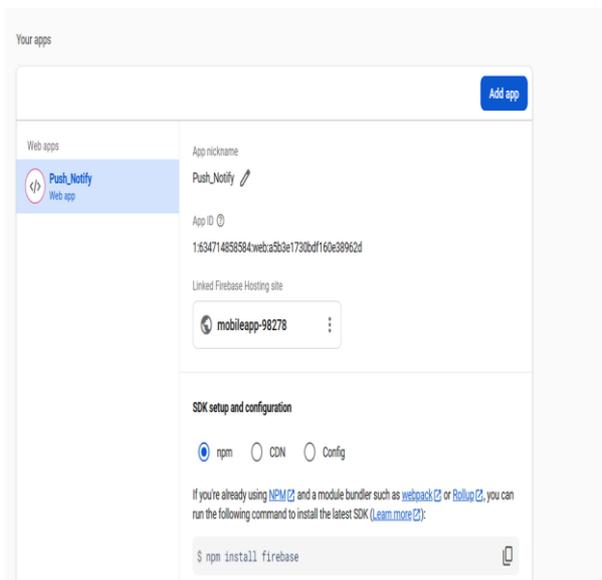
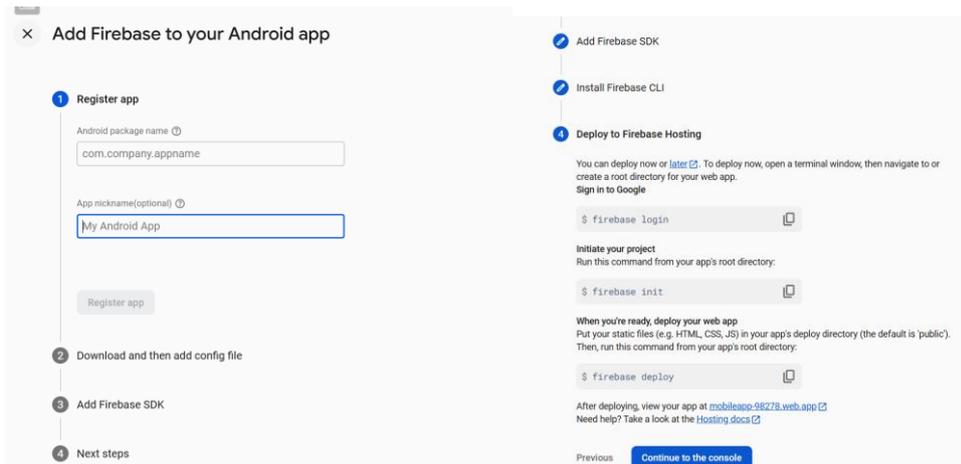
The **Firebase CLI (Command Line Interface)** allows you to connect your local project files to Firebase and perform hosting commands.

Run the following command in your terminal to install it globally:

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

Note: The Firebase CLI helps in initializing, configuring, and deploying your web app directly to Firebase Hosting.

Then Click next and Continue To The Console.



Step-by-Step: Add Android App

Step 1: Register the Android App

In the Firebase Console, click **Add App** → **Android**.

Enter the following details:

Package Name: com.YourCompany Name

App Nickname: PushNotify

Click **Register App** and then **Download google-services.json**.

Keep the file safe — it will be used later in Mendix for push notification configuration.

- You'll upload this file later — keep it safe for now.
- Click Next and Continue to console

Step 6: Register the iOS App

1. Click **Add App** → **iOS** in the Firebase Console.
2. Enter the following details:
 - **iOS Bundle ID:** (Use your app's bundle identifier)
 - **App Nickname:** PushNotify_iOS
1. Click **Register App** and download the **GoogleService-Info.plist** file.
2. Keep the file — it will be added to your iOS project later for Firebase integration.

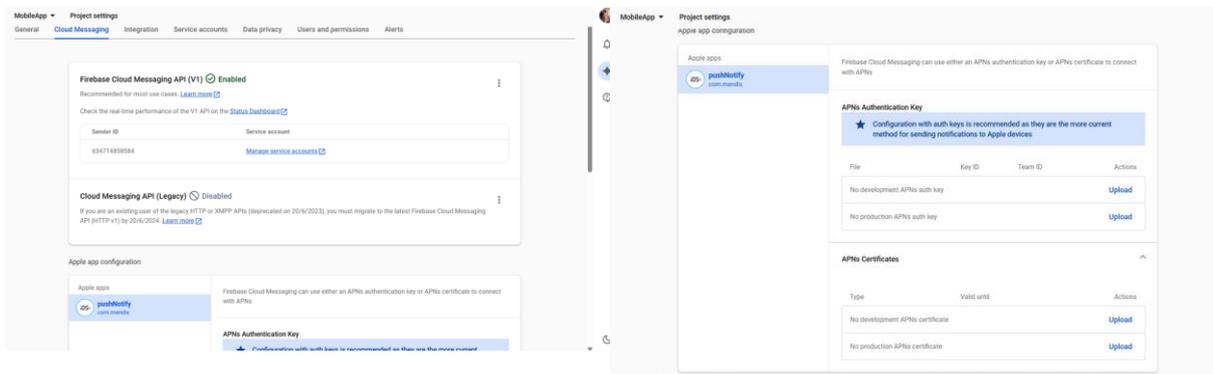
After setting up Firebase for Android, iOS, and Web apps, proceed to the **Cloud Messaging** tab in **Project Settings**.

CloudMessaging Configuration

1. **Firestore Cloud Messaging API (V1)**
 - The **V1 API** is **enabled** by default and is the recommended version for sending push notifications.
 - It provides enhanced security and reliability compared to the legacy HTTP API.
1. **Legacy API (Deprecated)**
 - The **HTTP/XMPP Legacy API** is now **deprecated**.
 - All new configurations must use the **V1 API** for better performance and compatibility.

1. App Configurations (iOS & Web)

- **iOS Configuration:** Use APNs authentication keys or certificates to enable notification delivery on Apple devices.
- **Web Configuration:** Generate a **Web Push Certificate (VAPID key pair)** for browser-based push notifications.



4. Click on “Generate key pair” under Web Push certificates.

What happens:

When you click it, **Firestore automatically creates a VAPID key pair** (a public and private key).

You’ll then see a long string (starting with something like `BDiP_F1Fdb...`) appear — that’s your **public VAPID key**.

How to Use It:

After generating the key:

1. **Copy the public key** (the long alphanumeric code displayed after generating the key pair).
2. Go to your **Mendix application** → **Push Notification Module** → navigate to *useme* → *web* → *WebPushVapidKey*.
3. **Paste the copied public key** into the **WebPushVapidKey** field.
4. Save the changes.

This configuration links your Firebase Cloud Messaging (FCM) with your Mendix application, allowing it to send secure web push notifications to users' browsers.

Setting Up a Service Account

1. In the **Firebase Console**, click the **⌵ (Settings)** icon in the upper-left corner and select **Project settings**.
2. Navigate to the **Service accounts** tab.
3. Click **Generate new private key** and **download the JSON file**.
 - Store this file in a **secure location**, as it contains credentials required for backend access.
1. You'll use this **private key file** later when configuring **Firebase Cloud Messaging (FCM)** in your Mendix application backend.

Note:

The generated key provides full API access to all Firebase services.

For security, if you only need access to **Cloud Messaging**, click **Manage all service accounts** (top-right corner) and create a **restricted service account** limited to FCM usage.

Part 4: Configure Push Notifications

1. Sync Entities using Model Reflection

- Open your **Mendix Studio Pro** project.
- Navigate to **App Explorer** → **Modules** → **Model Reflection**.
- Click **Sync Entities** to ensure all the **Push Notification** module entities and their required dependencies are correctly reflected in your domain model.

2. Configure Encryption Key

- Go to **App Explorer** → **Settings** → **Configurations** → **Constants**.
- Locate the constant named **Encryption.EncryptionKey** and provide a unique key value (for example: PushNotify@2025Key!).

Why Encryption Key is Used

The **Encryption Key** is essential for securely storing sensitive data such as:

- Device tokens
- Notification credentials
- Firebase service account details

It ensures that confidential data is **encrypted and decrypted safely** when communicating between Mendix and Firebase. Without this key, your push notification setup will fail to authenticate or send messages securely.

If you have completed Add Module Dependencies and Implement the Push Notifications Module per your use case, do the following to configure your push notifications:

3. Configure Security & User Roles

- Open **App** → **Security** in the **App Explorer**.
- Go to **User Roles** and assign the following:
 - **Administrator** → For users managing configuration and administrative tasks.
 - **Anonymous** → Optional, for public/guest users.
 - **User** → For regular users who receive notifications.
- For **Native Mobile Apps**, ensure that **NativeMobileResources.User** role is added to any user role that interacts with notifications.
- Click **Save** to update security settings.

Configure Navigation

- Navigate to **App** → **Navigation**.
- Under the **Responsive navigation profile**, add a new **Open Page** item.

- Select the **Administration page** from the **PushNotifications** module (located in _USE ME/Admin).

Configure After Startup Microflow

- Open or create your **AfterStartup** microflow.
- Add a **Submicroflow call** to **AfterStartup_PushNotifications**.
- This ensures that push notification services initialize automatically when your app starts.

If you have completed

1. Open your app in Mendix Studio Pro.
2. Log in as the Administrator user
3. Navigate to the **Administration** page.

Part 5: Setting Up Firebase Push Platforms in Mendix

If you have completed the above step then follow this :

1. Open your app in Mendix Studio Pro.
2. Log in as the Administrator user
3. Navigate to the **Administration** page.
4. When the configuration window appears, you'll see options to select which platforms you want to support for push notifications:
 - **Push notifications for iOS**
 - **Push notifications for Android**
 - **Push notifications for Web**

Select the platforms you want to support

Push notifications for iOS

Push notifications for Android

Push notifications for Web

Set the project ID generated by Firebase for your project

Upload the service account private key you retrieved from Firebase

... [Browse...](#)

[Complete initial setup](#) [Skip](#)

Select all the platforms you want to enable (for example, iOS, Android, and Web).

In the field “Set the project ID generated by Firebase for your project”,

- Go to your Firebase Console → Project Settings → General tab.
- Copy the Project ID shown there.
- Paste it into this field in Mendix.

In the section “Upload the service account private key you retrieved from Firebase”,

Go to Firebase Console → Project Settings → Service Accounts tab.

Click Generate new private key to download the JSON file.

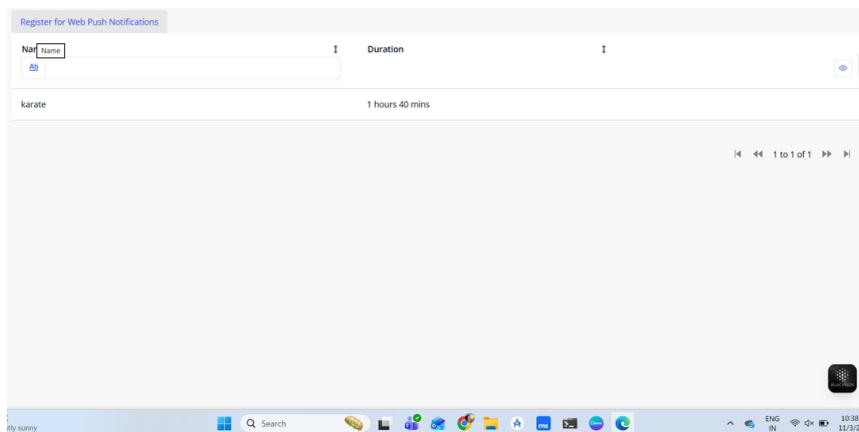
In Mendix, click Browse and upload that same file here, Once the details are filled, click Complete Initial Setup.

WEB Push Notification

Adding Web Registration Snippet

Steps:

1. Open your Home_Web page (or any main page where users land after login).
2. Drag and drop the [WebRegistration_Snippet] from the PushNotifications → USE_ME → Snippets section into your page layout.
3. Place it at the top of the page (as shown in the screenshot).



Purpose:

- This snippet is responsible for registering your web users for push notifications.
- When a user opens the page, this snippet runs the logic to request browser permission and register the device with Firebase for web push notifications.

Where to Use:

- Add this snippet to any web page where you want users to start receiving notifications automatically (commonly the Home page or Dashboard page).
- This ensures that once users visit the app, their browser subscribes to the notification service in the background.

Allow Browser Notifications

Step:

After adding the WebRegistration_Snippet and running your app, make sure that (User)browser notifications are allowed for your Mendix app domain.

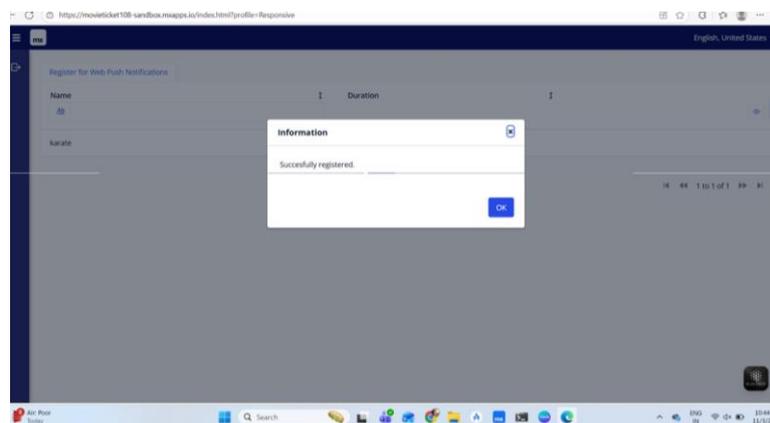
How to Check:

1. Click the (lock icon) next to your app URL in the browser.
2. Under Permissions for this site, ensure that Notifications → Allowed is selected (as shown in the screenshot).
3. If it's blocked, change it to Allow and refresh the page.

Result:

Once added and deployed, when a user visits the page:

- The browser will ask permission to show notifications.
- If accepted, the user's device will be registered and start receiving push notifications from your Mendix app.



What happens here:

Once a user clicks “**Register for Web Push Notifications**”, the app runs a microflow that registers the browser and then displays this popup —

“Successfully registered.”

This message indicates that:

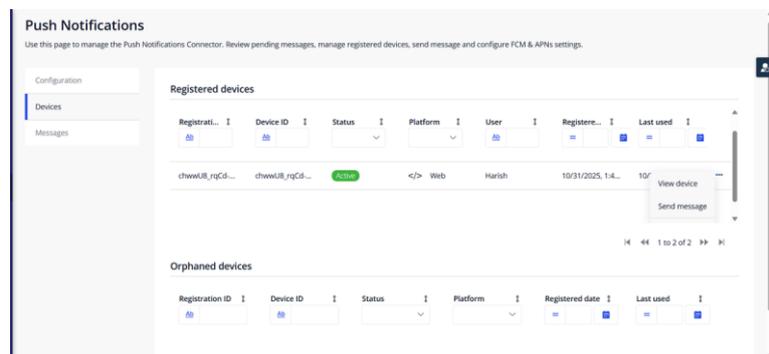
- The **FCM (Firebase Cloud Messaging)** connection is active.
- The user's browser has been **successfully subscribed** for push notifications.
- The app can now **send notifications** directly to this browser session.

Sending a Test Push Notification

After successfully registering your device for web or mobile push notifications, follow the steps below to send your first test message.

Steps to Send a Push Notification

1. **Login** to your web application.
2. Navigate to the **Push Notifications Administration Page** that you added earlier in your app's navigation.
3. Go to the **Devices** tab.



Here, you will see a list of all registered devices categorized by **Platform** (Web / Android / iOS).

- **Select your test device** from the list, You can either click **Send Message** from the options menu or **double-click** the device entry.

Message details
Recipient(s)

Title

Body

Action name

Identifier (GUID)

The action to be performed when the user opens the notification. This field is optional. If provided, this should match one of the configured actions in the Notifications widget.

Scheduled delivery

⌵

Badge

Google configuration

Time to live

Apple configuration

Subtitle

Play sound

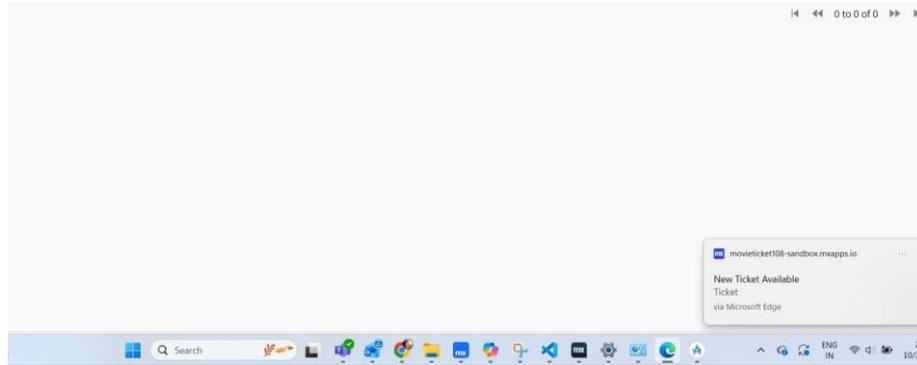
Send message
Cancel

In the **Message Details** form:

- **Title:** Enter any title for your notification.
- **Body:** Enter the message text you want to send.
- **Action name:** Set this to **Example**.
- Leave all other fields with their **default values**.
- Click **Send Message**.

Expected Result

Once sent, the registered device (Web, Android, or iOS) will receive the notification instantly as shown below.



Push Notifications in Progressive Web Apps (PWA)

Overview

Push Notifications in a **Progressive Web App (PWA)** work **similarly to Web Push Notifications**, but they have a few important limitations and setup differences.

Key Points

1. Uses the Push Notifications Module

- PWA push notifications are configured using the Push Notifications module.
- The setup process (Firebase project, VAPID key, service account key) is the same as for web.

1. Do NOT use the Notifications Widget

- The Notifications widget provided in the Push Notifications Connector is not supported in PWAs.
- This widget only works in hybrid mobile apps (built with native shell support).
- Receiving or handling “on receive” or “on click” events inside the app is not available for PWAs.

1. Device Registration

- The PWA registers as a web device in the Push Notifications Admin page.
- Once registered, you can send notifications to it from the Devices tab just like Android/iOS/Web.

Initialize Firebase for PWA Push Notifications

Steps to Configure

Create a Custom index.html

- Go to your project folder → theme\web
- Create a new file named index.html
- Follow the Mendix guide for creating a custom index page.

Initialize Firebase

- Open the new index.html in a text editor.
- Add the below Firebase setup before the line:

```
<script src="mxclientsystem/mxui/mxui.js?{{cachebust}}"></script>
```

For This There Is an Example

Mendix recommends copying *index-example.html* from the **deployment/web** folder to the **theme/web**, rename it to *index.html*, and then use it as a starting point. copy this and paste in theme\web

```
<script src="https://www.gstatic.com/firebasejs/10.11.0/firebase-app-compat.js"></script>
<script src="https://www.gstatic.com/firebasejs/10.11.0/firebase-messaging-compat.js"></script>
<script>
const firebaseConfig = {
  apiKey: "...",
  authDomain: "...",
  projectId: "...",
  storageBucket: "...",
  messagingSenderId: "...",
  appId: "..."
};
firebase.initializeApp(firebaseConfig);
</script>
```

This is an Sample Index.html code

```
<!doctype html>
```

```

<html>
  <head>
    <meta charset="utf-8"><title>Mendix</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    {{themecss}}
    {{appicons}}
    {{manifest}}
    {{startupimages}}
  </head>
  <body dir="ltr">
    <noscript>To use this application, please enable JavaScript.</noscript>
    <div id="content"></div>
    <script>
      dojoConfig = {
        isDebug: false,
        useCustomLogger: true,
        async: true,
        baseUrl: "mxclientsystem/dojo/",
        cacheBust: "{{cachebust}}",
        rtlRedirect: "index-rtl.html"
      };
    </script>
    <script>
      if (!document.cookie || !document.cookie.match(/(^|;) *originURI=/gi)) {
        const url = new URL(window.location.href);
        const subPath = url.pathname.substring(0, url.pathname.lastIndexOf("/"));
        document.cookie = `originURI=${subPath}/login.html${window.location.protocol === "https:" ? ";SameSite=None;Secure": ""}`;
      }
    </script>
    <!-- Add Firebase SDKs BEFORE Mendix runtime -->
    <script src="https://www.gstatic.com/firebasejs/10.11.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/10.11.0/firebase-messaging-compat.js"></script>
    <script>
      const firebaseConfig = {
        apiKey: "AIzaSyAVQ-aPQ3FtcWzryfELoF9gez812foLkRY",
        authDomain: "movie-d0023.firebaseio.com",
        projectId: "movie-d0023",
        storageBucket: "movie-d0023.firebaseio.com",
        messagingSenderId: "527349464471",
        appId: "1:527349464471:web:160a654178339578f958d2",
        measurementId: "G-ERCGZSB7GE"
      };
      firebase.initializeApp(firebaseConfig);
      const messaging = firebase.messaging();
    </script>

```

```
<!-- Mendix Runtime must load last -->
<script src="mxclientsystem/mxui/mxui.js?{{cachebust}}"></script>
</body>
</html>
```

Create the file `theme\web\firebase-messaging-sw.js` with the following content (replace `firebaseConfig` with your configuration from :

```
importScripts('https://www.gstatic.com/firebasejs/10.11.0/firebase-app-compat.js');
importScripts('https://www.gstatic.com/firebasejs/10.11.0/firebase-messaging-compat.js');

const firebaseConfig = {

  apiKey: "...",

  authDomain: "...",

  projectId: "...",

  storageBucket: "...",

  messagingSenderId: "...",

  appId: "..."

};

firebase.initializeApp(firebaseConfig);

const messaging = firebase.messaging();
```

This is An Sample `Index.html` code

```
// Import Firebase scripts
importScripts('https://www.gstatic.com/firebasejs/10.11.0/firebase-app-compat.js');
importScripts('https://www.gstatic.com/firebasejs/10.11.0/firebase-messaging-compat.js');
// Initialize Firebase (same config as in index.html)
```

```
firebase.initializeApp({
  apiKey: "AIzaSyAVQ-aPQ3FtcWzryfELoF9gez812foLkRY",
  authDomain: "movie-d0023.firebaseio.com",
  projectId: "movie-d0023",
  storageBucket: "movie-d0023.firebaseio.com",
  messagingSenderId: "527349464471",
  appId: "1:527349464471:web:160a654178339578f958d2",
  measurementId: "G-ERCGZSB7GE"
});
// Retrieve Firebase Messaging instance
const messaging = firebase.messaging();
// Handle background messages
messaging.onBackgroundMessage(function(payload) {
  console.log('Received background message: ', payload);
  const notificationTitle = payload.notification.title;
  const notificationOptions = {
    body: payload.notification.body,
    icon: '/push-icon.png' // optional icon
  };
  self.registration.showNotification(notificationTitle, notificationOptions);
});
```

Add the snippet `WebRegistration_Snippet` found in `_USE ME/Web` in the Push Notifications module to your home page. It contains a button that your users must click to register for push notifications.

Stop the Mendix Runtime in Studio Pro if it is running and start it afterwards. Do not use **Rerun**, as that will not pick up the changes in your theme folder.

Sending the Test Notification

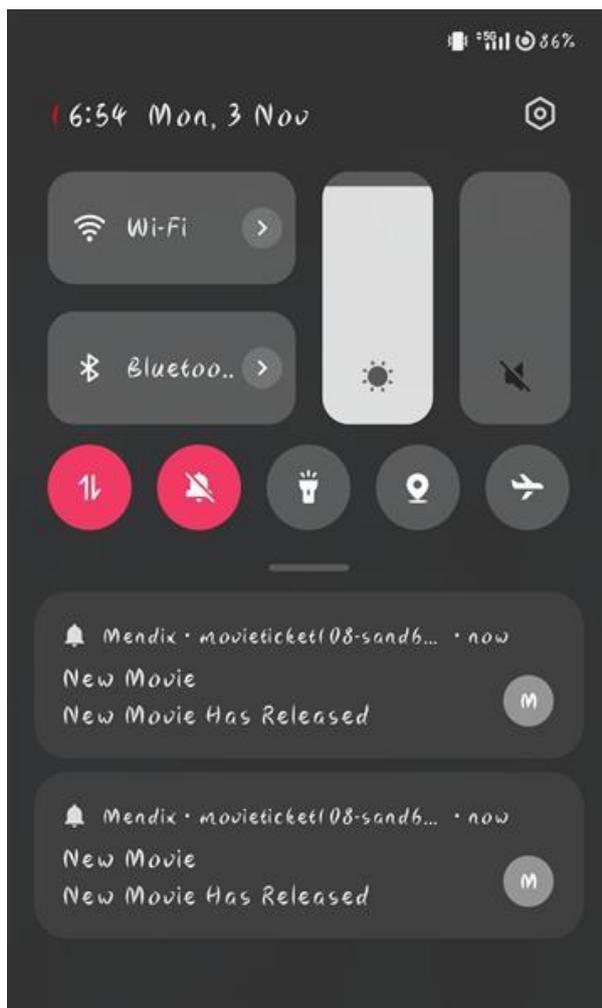
To send your first test notification, do the following:

1. Login to your web application.
2. Go to the push notification administration page you added earlier to your navigation.
3. Go to the **Devices** tab.
4. Select the your test device.
5. Click **New Message** (or double-click your test device).

6. Fill in any **Title/Body** that you want for your notification.
7. Set the **Action name** to *Example*.
8. Leave the remaining fields to the defaults.
9. Click **Send**

Expected Result

Once sent, the registered device (PWA) will receive the notification instantly as shown below.



Push Notifications in Native

This section explains how push notifications work in **Mendix Native Mobile Apps** and how device registration, Firebase integration, and message delivery are handled.

Push Notifications for native apps rely on **Firestore Cloud Messaging (FCM)** and the **Mendix Native Mobile Runtime**.

Unlike Web/PWA, **native apps register automatically** for push notifications—no snippet or button is required.

1. How Device Registration Works (Automatic)

Unlike Web/PWA, **Native mobile apps do not use any registration snippet**.

When the mobile app **starts**, Mendix automatically runs the native push registration flow:

What happens during app launch:

1. The **AfterStartup_PushNotifications** microflow is executed.
2. The module calls the **Native JavaScript Action**:
3. RegisterForPushNotifications
4. The React Native shell calls the Firebase Mobile SDK.
5. Firebase generates a **device push token**.
6. Mendix saves this token into:
7. PushNotifications.Device
8. The device appears automatically in:

Administration → **Devices Tab**

Native device registration is **fully automatic**.

2. Requirements Before Native Registration Works

Ensure these prerequisites:

- 1. Firebase Android Setup

- Add **google-services.json** inside your Mendix native folder
(/android/app/ once built)

□ 2. Firebase iOS Setup

- Add **GoogleService-Info.plist**
(inside /ios/ folder created after native build)

□ 3. Push Notification Configuration in Mendix

- Administration Page → Configuration Tab:
 - Enable:
 - Push notifications for Android
 - Push notifications for iOS
 - Paste **Firestore Project ID**
 - Upload **Firestore Service Account JSON file**
 - Complete Initial Setup

□ 4. User Must Have Correct Module Roles

Add module role:

NativeMobileResources.User

PushNotifications.User

3. Handling Notification Click Actions

Native apps support notification actions through the **Notifications widget**.

The widget triggers a nanoflow with:

- Title
- Body
- ActionName
- GUID (optional)

Example:

When ActionName = “OpenOrder”, you can open a specific page or object in the app.

To configure custom actions:

1. Open your native layout page.
2. Locate **Notifications widget** in the page.
3. Set custom Action Names.
4. Create nanoflows for each action.

4. Sending Push Notifications

To send to a native device:

1. Open the app in browser (as Admin).
2. Go to:

Administration → Devices

3. You will see devices:
 - Platform = Android / iOS
 - Status = Active

1. Select the device

2. Click:

Send Message

3. Fill details:
 - Title
 - Body
 - Action Name

Click **Send Message**.

Your native app will receive the notification instantly.

Expected Result

Once sent, the registered device (Andriod) will receive the notification instantly as shown below.

