Expo Push Notification Integration

Overview

This document outlines the complete end-to-end implementation of **Expo Push Notifications** in an **Expo-managed React Native app** with notifications triggered from a **custom backend server**. It includes setup for both Android and iOS, development and production considerations, and integration of the `expo-server-sdk` on the backend.

Step:1FCM Setup for Android

1.1Generate Service Account Key (Required by Expo)

- Go to Firebase Console.
- Open your project or create a new one.
- Navigate to **Project Settings > Service Accounts**. Click **Generate new private key** and download the `serviceAccountKey.json`

1.2Upload Service Account Key to Expo

- Go to Expo Developer Dashboard.
- Open your project.
- Navigate to **Push Notifications > Android Credentials**.
- Upload the previously downloaded `serviceAccountKey.json` file.

1.3Generate google-services.json File:

- In Firebase Console, go to **Project Settings > General > Your Apps (Android)**.
- Register your app's **Android package name**.
- Add your **SHA-1 key** (required for proper functioning with FCM) which you get where you have upload serviceaccountKey.json file in the expo.dev dashboard
- Download the google-services.json file.
- Place this file inside your Expo project directory.
- Add it to `.gitignore` for security reasons.
- However, ensure it's **not ignored by EAS** by managing `.easignore` accordingly.

Step 2: Frontend Notification Setup (Expo App)

2.1 Install Required Packages

- npx expo install expo-notifications expo-device expo-constants

2.2 Register for Push Notifications

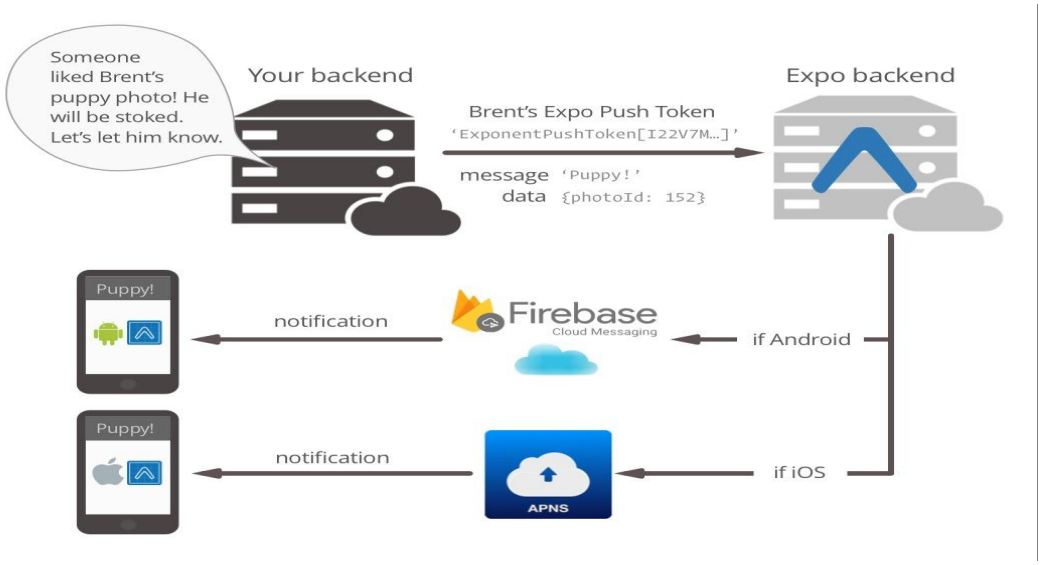
- In your component, Use expo-notifications to handle notification permissions and token generation. Use expo-device to check if the app is running on a physical device. **(we can refer its documentation to implement this)**
- **No separate setup is needed** for Android and iOS when using Expo.
- The same logic and codebase handles push notifications across both platforms seamlessly.

Step 3: Generate a Development Build

- Expo Go **does not support push notifications**. So we need dev build to implement expo notification.
- Using **expo build –profile development** command we can generate development build.
- Once the build is complete, install the APK on your device and run the app to test notifications.
- Get Push Token and Use Expo Push Notification Tool to test the token.

Step 4: Backend Integration

- Install Expo Server SDK in your backend and using that we can send notification to expo server



🔄 backend to expo server Flow Overview:

1. ***Trigger Event:***
    - ***A specific event occurs in your application (e.g RFQ creation).***
2. ***Send Push Token & Message:***
    - ***Your backend receives this event and prepares a payload that includes:***
        - ***Expo Push Token***
        - ***Notification message.***
        - ***Optional data(action url etc).***
3. ***Send to Expo Backend:***
    - ***The backend sends this payload to the Expo Push Notification API.***
4. ***Expo Forwards to Device Platforms:***
    - ***For Android, it forwards the notification to Firebase Cloud Messaging (FCM).***
    - ***For iOS, it sends the notification to Apple Push Notification Service (APNS).***

Resolvers with Integrated Notification Logic:

Scenario: RFQ Created by Store Manager – Notify to PM

- Resolver: createRFQRequirement()
- Notification Function: sendRFQNotification() triggered to notify PMs about a new RFQ

Scenario: On creating RFQ and selecting vendors - Notify to ALL selected vendors

- Resolver: createRFQConfiguration()
- Notification function: sendVendorRFQNotification() triggered to notify Vendors(sellers) about new rfq

Scenario: Notify to PMs on receiving Bid from seller

- Resolver: createRfqResponse()
- Notification function: sendRFQBidReceivedNotification() triggered to notify PMs

Scenario: Notification to Supplier when po is awarded

- Resolver: generateManualPurchaseOrder() and generateRFQPurchaseOrder()
- Notification function: sendPOCreationPushNotification() triggered to notify supplier

Scenario: Notification for Cancel PO to supplier

- Resolver: cancelPO()
- Notification function: sendPOCancelledNotification() triggered to notify supplier on Cancel PO.

Final Notes:

- Keep your push tokens in a database mapped to user IDs for easy access.
- Never commit `google-services.json` or service account keys to Git.

Code Path:-

Frontend :- app\react-native-app\src\hooks\usepushNotification.js

Backend:- src\common\fcm-push-notification –

├── fcm-push-notification/

│ └── fcm.graphql.js

│ └── fcm.resolvers.js - to store and remove expo token

│ └── fcm.typedefs.js

│ └── notifications.js – functions which actually send notification to expo server from our server. These functions we have integrated in different resolvers.