Push Notification Design

Push Notification Design

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Architecture Overview

```
sequenceDiagram
    participant Backend
    participant NotifService as Notification Service\n(OneSignal / Firebase)
    participant APNs as APNs (iOS)
    participant FCM as FCM (Android)
    participant Device

    Backend->>NotifService: POST /notifications\n{userId, type, data}
    NotifService->>APNs: Push payload (iOS users)
    NotifService->>FCM: Push payload (Android users)
    APNs-->>Device: Deliver notification (iOS)
    FCM-->>Device: Deliver notification (Android)
    Device->>Backend: Delivery receipt (optional)
    Device->>Backend: Open/click event (analytics)
```

Push service: `{{OneSignal | Firebase Cloud Messaging (unified) | AWS SNS | Custom}}`

2. Provider Setup

2.1 APNs (iOS)

Property	Value
Auth method	`{{APNs Auth Key (.p8) — preferred over cert}}`
Key ID	`{{KEY_ID — from Apple Developer Portal}}`
Team ID	`{{TEAM_ID}}`
Bundle ID	`{{com.company.app}}`
Environment	Dev: Sandbox
Key location	`{{Vault reference — never commit}}`

Capabilities required in Xcode:

☑ Push Notifications

☑ Background Modes → Remote notifications

- `{{[x] Background Modes → Background fetch}}` (if using background sync)

2.2 FCM (Android)

Property	Value
Project ID	`{{firebase-project-id}}`
Server key	`{{Vault reference}}`
Sender ID	`{{SENDER_ID}}`
google-services.json	`android/app/google-services.json` (gitignored — CI injected)

Android Manifest additions:

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<!-- For Android 13+ — request at runtime -->
```

2.3 Unified Service Configuration

SDK: `{{expo-notifications | react-native-firebase | onesignal-react-native}}`

```
// src/services/notifications.ts — abstraction layer
export async function registerForPushNotifications(): Promise<string | null> {
  const { status: existingStatus } = await Notifications.getPermissionsAsync();
  let finalStatus = existingStatus;

  if (existingStatus !== 'granted') {
    const { status } = await Notifications.requestPermissionsAsync();
    finalStatus = status;
  }

  if (finalStatus !== 'granted') return null;

  const token = await Notifications.getExpoPushTokenAsync({
    projectId: Constants.expoConfig?.extra?.eas?.projectId,
  });

  // Register token with backend
  await api.post('/users/push-token', { token: token.data });
  return token.data;
}
```

---

3. Notification Types & Channels

3.1 Transactional Notifications

Type	Trigger	Priority	Sound	Badge
`order.confirmed`	Order placed	High	Default	+1
`payment.received`	Payment processed	High	Default	+1
`message.received`	New chat message	High	Custom	+1
`account.security`	Password changed, new login	Critical	Default	+1
`{{TYPE}}`	`{{TRIGGER}}`	`{{High/Normal/Low}}`	`{{Default/Custom/None}}`	`{{+1/None}}`

3.2 Marketing / Engagement Notifications

Type	Description	Frequency Cap	Opt-out Channel
`promo.offer`	Discount or limited-time offer	Max 2/week	Marketing channel
`feature.announce`	New feature announcement	Max 1/month	Marketing channel
`re_engagement`	Win-back inactive users	Max 1/week	Marketing channel

3.3 Android Notification Channels

```
// Create channels on app startup (Android 8+ / API 26+)
await Notifications.setNotificationChannelAsync('transactional', {
  name: 'Orders & Payments',
  importance: Notifications.AndroidImportance.HIGH,
  vibrationPattern: [0, 250, 250, 250],
  lightColor: '#FF231F7C',
  sound: 'default',
});

await Notifications.setNotificationChannelAsync('messages', {
  name: 'Messages',
  importance: Notifications.AndroidImportance.HIGH,
  sound: 'message.wav',
});

await Notifications.setNotificationChannelAsync('marketing', {
  name: 'Promotions & Updates',
  importance: Notifications.AndroidImportance.DEFAULT,
  sound: null,
});
```

---

4. Payload Format & Schema

4.1 Data Payload vs Notification Payload

Type	Shown by OS?	Customizable	Use when
Notification payload	Yes (auto)	Limited	Simple alerts
Data payload	No	Full control	App handles display

Type	Shown by OS?	Customizable	Use when
Combined	Yes + data	Full	Standard approach

Recommended approach: Send both — notification payload for guaranteed delivery, data payload for app logic.

4.2 Standard Payload Schema

```
{
  "notification": {
    "title": "{{Notification title}}",
    "body": "{{Notification body text}}",
    "image": "{{optional image URL}}"
  },
  "data": {
    "notificationType": "{{order.confirmed | message.received | etc.}}",
    "entityType": "{{order | message | user | etc.}}",
    "entityId": "{{UUID}}",
    "deepLinkPath": "{{/orders/123}}",
    "timestamp": "{{ISO_8601}}",
    "version": "1"
  },
  "apns": {
    "payload": {
      "aps": {
        "sound": "default",
        "badge": 1,
        "content-available": 1
      }
    }
  },
  "android": {
    "channelId": "{{transactional | messages | marketing}}",
    "priority": "high"
  }
}
```

4.3 Localization

- All notification text must be localized
- Approach: `{{Backend sends localized text based on user's locale preference | App localizes using data payload keys}}`
- Fallback locale: `en`

---

5. Deep Linking Strategy

5.1 URL Scheme Configuration

URL scheme: `{{appname://}}` **Universal links (iOS):** `{{https://app.domain.com}}` **App links (Android):** `{{https://app.domain.com}}`

5.2 Deep Link Routing

Notification Type	Deep Link	Screen
`order.confirmed`	`appname://orders/{{orderId}}`	Order Detail
`message.received`	`appname://chats/{{chatId}}`	Chat Screen
`payment.received`	`appname://wallet`	Wallet Screen
`promo.offer`	`appname://offers/{{offerId}}`	Offer Detail

5.3 Navigation on Notification Tap

```
// App foreground — notification tap handler
Notifications.addNotificationResponseReceivedListener((response) => {
  const { notificationType, deepLinkPath } = response.notification.request.content.data;

  if (deepLinkPath) {
    // Use router.push for Expo Router, or navigation.navigate for React Navigation
    router.push(deepLinkPath);
  }
});

// App killed — check initial notification
useEffect(() => {
  Notifications.getLastNotificationResponseAsync().then((response) => {
    if (response?.notification.request.content.data?.deepLinkPath) {
      router.replace(response.notification.request.content.data.deepLinkPath);
```

```
    }
  });
}, []);
```

---

6. Opt-In / Opt-Out Flow

Permission request timing: `{{After onboarding, on first meaningful action — NOT on app launch}}`

Permission flow:

```
App Launch
    ↓
Onboarding Complete
    ↓
First Relevant Action (e.g., order placed)
    ↓
Pre-prompt Modal (custom UI — explain value)
    "Get notified when your order is ready"
    [Allow]  [Not now]
    ↓ (Allow)
OS Permission Dialog
    ↓
Register token with backend
```

Soft prompt: Always show custom pre-prompt before OS dialog. Users who dismiss OS dialog permanently cannot be re-asked — use pre-prompt to qualify intent.

---

7. Notification Preferences Per User

```
interface NotificationPreferences {
  pushEnabled: boolean;       // Master toggle
  channels: {
    transactional: boolean;   // Orders, payments — default: true
    messages: boolean;        // Chat messages — default: true
    marketing: boolean;       // Promotions — default: false
    security: boolean;        // Security alerts — always true, non-toggleable
```

```
  };
  quietHours: {
    enabled: boolean;
    start: string;            // "22:00" HH:mm
    end: string;              // "08:00" HH:mm
    timezone: string;         // "Europe/Oslo"
  };
}
```

API endpoint: `PUT /users/notification-preferences`

Sync: Preferences stored on server — synced on login and app foreground.

---

8. Rate Limiting & Throttling

Category	Limit	Window	Behavior at limit
Transactional	Unlimited	—	Always delivered
Messages	100	1 hour	Batch: "X new messages"
Marketing	2	7 days	Drop excess, no queue
Security	Unlimited	—	Always delivered

Backend enforcement: Rate limit checked before sending to push service. Excess notifications are logged but not sent.

---

9. Analytics & Tracking

Event	Tracked How	Data Points
Sent	Backend log	notificationType, userId, timestamp
Delivered	Push service receipt	notificationType, userId, deliveredAt
Opened	App handler	notificationType, userId, openedAt, timeToOpen
Dismissed	`{{Platform support}}`	`{{Android only – iOS limited}}`
Action tap	App handler	notificationType, actionId, userId

Funnel metrics:

- Delivery rate = delivered / sent
- Open rate = opened / delivered
- CTR (click-through) = action taps / opened

Privacy: User IDs are hashed in analytics. No PII in event data.

---

10. Testing Strategy

Test Type	Method	Environment
Payload validation	Unit test push service	Dev
Delivery smoke test	Send test notification via dashboard	Staging
Deep link routing	E2E test: tap notification → verify screen	Staging
Opt-in flow	E2E test: full permission flow	Staging / device
Rate limiting	Integration test: exceed limit → verify drop	Staging

Test push tool: `{{OneSignal Dashboard | Expo push notification tool | Firebase console}}`

TODO: Create Maestro test flows for notification tap → deep link scenarios.

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			
Backend Lead			
Product Owner			

Revision #8
Created 2026-02-23 12:05:21 UTC by John
Updated 2026-06-28 20:03:09 UTC by John