Courier

www.courier.com

Notifications for Product Managers

Contents

Introduction

Notifications form the basis for almost all communication with your customers — whether you're looking to inform them about new product features, alert them to a potential risk, or simply check in with them to entice them to engage with your product. But everyone has different devices, preferred communication channels, and expectations, making it hard to get the notification experience right.

If you're a PM, you might feel overwhelmed by the sheer number of options for sending out notifications. How do you engage your users without annoying them? How can you build a customizable experience without overburdening your developers? And how can you learn from previous user engagement with your notifications?

> This ebook serves as a guide for product managers looking to improve their product's notification experience.

We'll talk about why a great notification experience is so central to modern products, how you can give your users agency through preference management, and how you can use data analytics to improve your product's notification experience. Plus, we'll look at the best setup for your notification codebase, and how automation tools can help your dev team craft a notification experience that will keep your users active and engaged.

CHAPTER 1

Building a great notification experience

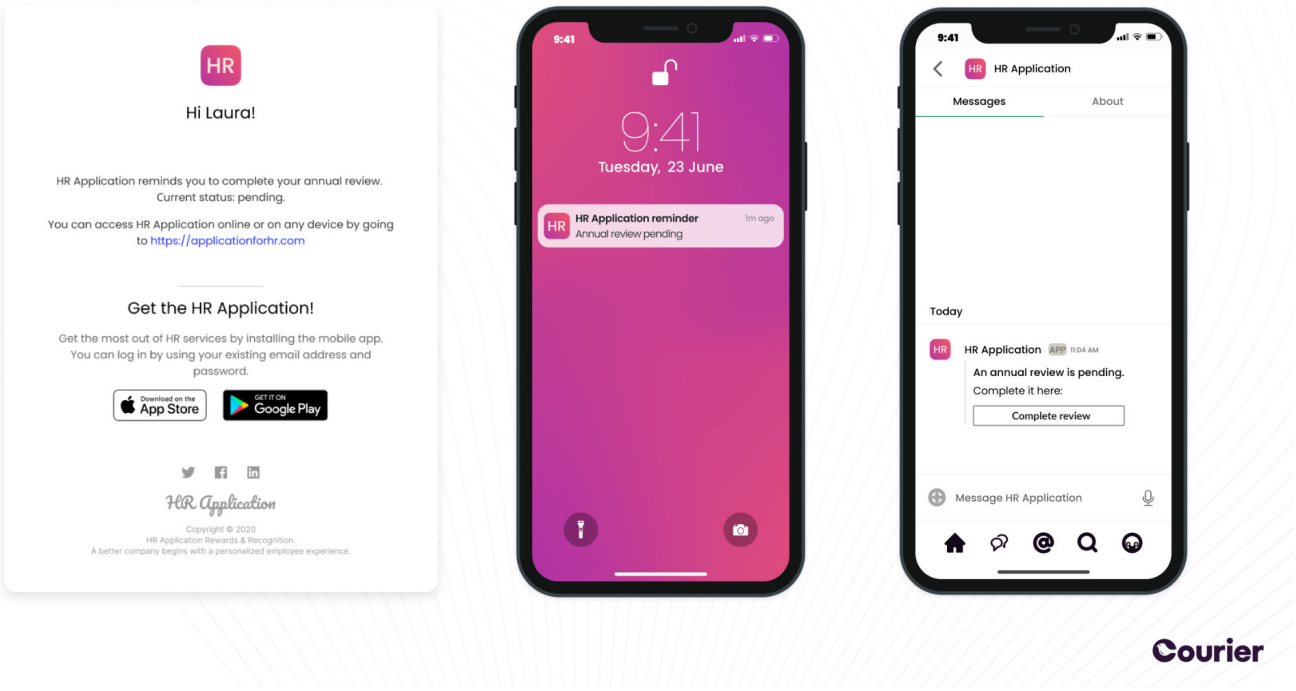
Your product's notification experience can make or break how your users perceive your application. That's why getting the notification experience right should be at the forefront of product managers' minds.

In this chapter, we'll do a deep dive into what contributes to a great notification experience. We'll talk about various notification types, which notification channel is best suited for what kind of message, and how to make sure your users don't leave your product due to wrongly timed notifications.

What is a notification experience?

The term “notification experience” includes everything your users see, hear, do, and feel when they interact with notifications from your app or software service — from emails to push notifications to Slack messages and beyond.

Email, push, and Slack notifications for an HR application.



Notification channels are a central component of a notification experience. Each channel is a way to reach the user through a particular technology, platform, or service that is able to deliver notifications. Here are some examples:

- Mobile push (including banners, badges, sound alerts)
- Email
- Chat (like Slack or WhatsApp)
- SMS
- Phone call
- In-app notifications

It’s rarely sufficient to only use one notification channel, which is why a successful notification experience includes **notification logic**. This logic decides which channels and sending times to use under which circumstances. It also processes user preferences and picks the right language based on the user’s profile.

Transactional versus lifecycle notifications

There are two broad types of notifications used by software products and services: transactional notifications and lifecycle notifications.

Transactional notifications are directly relevant to a specific action that a user takes within your app, like creating a post or placing an order. Examples include a delivery app that sends users an immediate order confirmation, or a marketplace app that notifies users of recent price changes for a product they're looking to buy.

Lifecycle notifications are meant to encourage users to return to and interact with the product itself, such as reminders to users who have been inactive for a longer period of time, or weekly content digests.

While you could use any notification channel for both transactional and lifecycle purposes, some are better suited to one or the other. The following table details several channels and their advantages and disadvantages for different types of notification.

A transactional push notification and a lifecycle push notification: similar format but different purposes.

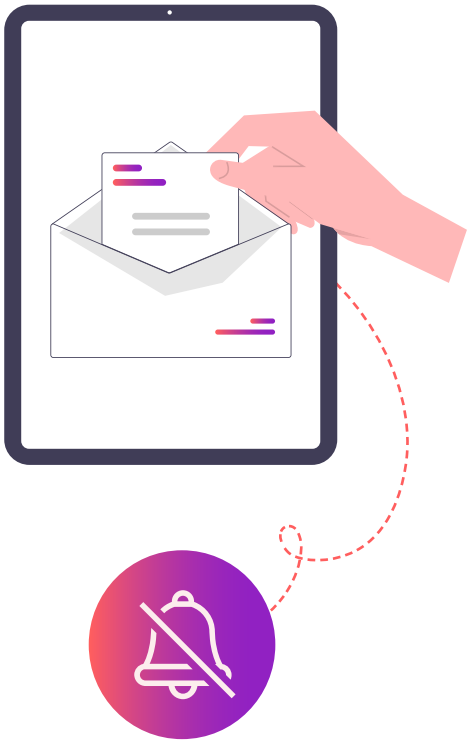
Channel	Advantages	Disadvantages	Best suited for	Not suitable for
Email	Wide adoption.	Can take days for the recipient to read an email. Possibility of hitting deliverability issues. Possibility of getting lost in the inbox.	Important information that the user shouldn't miss, like pricing changes, account updates, security alerts; marketing messages that are not time-sensitive.	Time-sensitive alerts — as email is not designed to be read instantly.
Mobile push	Received quickly by the user; can take the user directly to the app to perform an action.	Alert fatigue; limited settings on the user side.	Timely alerts.	(Arguably) marketing messages.
SMS	Wide adoption.	Doesn't work without a mobile connection. Length limitations.	Timely alerts about appointments, events, etc.	Most app notifications.
Chat (Slack, Discord)	Reaching chat app users with less interruption; giving users an opportunity to see messages in the app when browsing other activities.	Can get lost in the chat. Possibility of not reaching the recipient if they are not checking the chat often.	Context-specific notifications — for example, work conversations or gaming conversations.	Time-sensitive notifications outside of chat hours. Security alerts.

As the table shows, there is no "one size fits all" channel that is perfect for all uses. A good notification experience should therefore embrace flexibility and take advantage of each channel's strengths.

A bad notification experience can frustrate users

Product managers understand that notifications matter — but sometimes they don't *see just how much* they matter. A bad notification experience can be highly detrimental to the success of your application. For example:

- **Using notification channels incorrectly can cause users to churn.** Repeatedly abusing push notifications for time-insensitive alerts can feel unprofessional and spammy, causing users to lose trust in your product or service.
- **Wrongly timed notifications can interrupt and annoy users.** A user receiving an avalanche of notifications in a short time span or in the middle of the night is likely to completely block your app from displaying notifications, denying you a valuable engagement channel.



The hallmarks of a good notification experience

In contrast to the potentially disastrous consequences of a bad notification experience, a good experience can elevate your product in the minds of users. Here are the key aspects of a successful notification experience that you should consider when designing your own.

Right channels. Choosing suitable channels for your notifications makes it easier to tweak and improve your notification experience later on. For example, in a banking app, mobile push is a good fit for transaction notifications due to its timeliness, while email wouldn't be so suitable. However, if there is a suspicious transaction, it might be better to send an SMS, or even escalate to a phone call to make sure the customer doesn't miss the notification.

Right timing. It's better to avoid sending notifications through channels like mobile push at night, as that might wake your customer up unnecessarily. A local marketplace app, for instance, should avoid sending mobile push and other intrusive notifications after a certain time in the user's time zone. Instead, it could send a summary email in the morning.

Right customizations and preferences. In addition to allowing notifications to be completely enabled or disabled, applications should offer ways for users to customize notification settings in a more granular manner, to suit their individual needs and preferences. We'll explore notification preferences in the next chapter.

Following platform best practices. Whether using iOS push, in-app notifications, Slack, or another channel, tailoring your notifications to the channel they are delivered by will make them more effective. For example, an app sending notifications via Slack could make use of Slack's various built-in "blocks" to create seamless alerts that combine text, images, and buttons.

In order to build a great notification experience for their users, PMs should consider notifications a core part of the product and treat them accordingly. In the following chapters, we'll look at different methods for designing the kind of tailored, thoughtful, and seamless notification experience that your users are looking for.

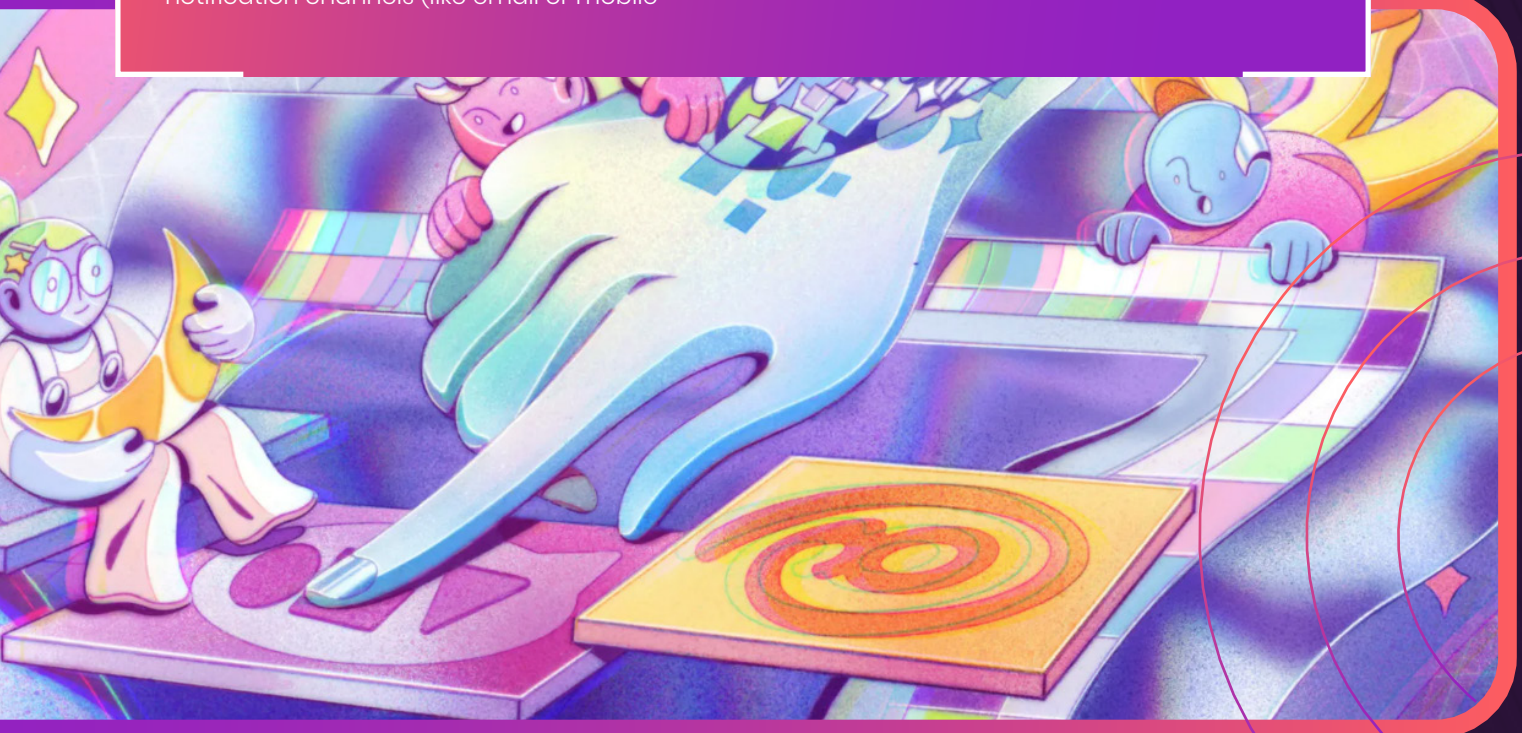
CHAPTER 2

Preference management for notifications

Implementing notifications is a balancing act. Users want to see new information like messages, time-sensitive tasks, or attractive product updates as soon as it's available — without becoming annoyed when they receive notifications through the wrong channel, or at an inconvenient time.

Productivity apps like Slack or Microsoft Teams, whose core functionality involves notifications, offer several options for users to configure their notification experience. From choosing notification channels (like email or mobile push) to configuring out-of-office hours, those apps try to make their users' lives less distracted and more productive by letting users control which notifications they want to get when and where.

In this chapter, we'll look at notification preferences — what they are and which preference options you should consider including in your app — and share some tips on which more advanced options you could build to delight your customers.

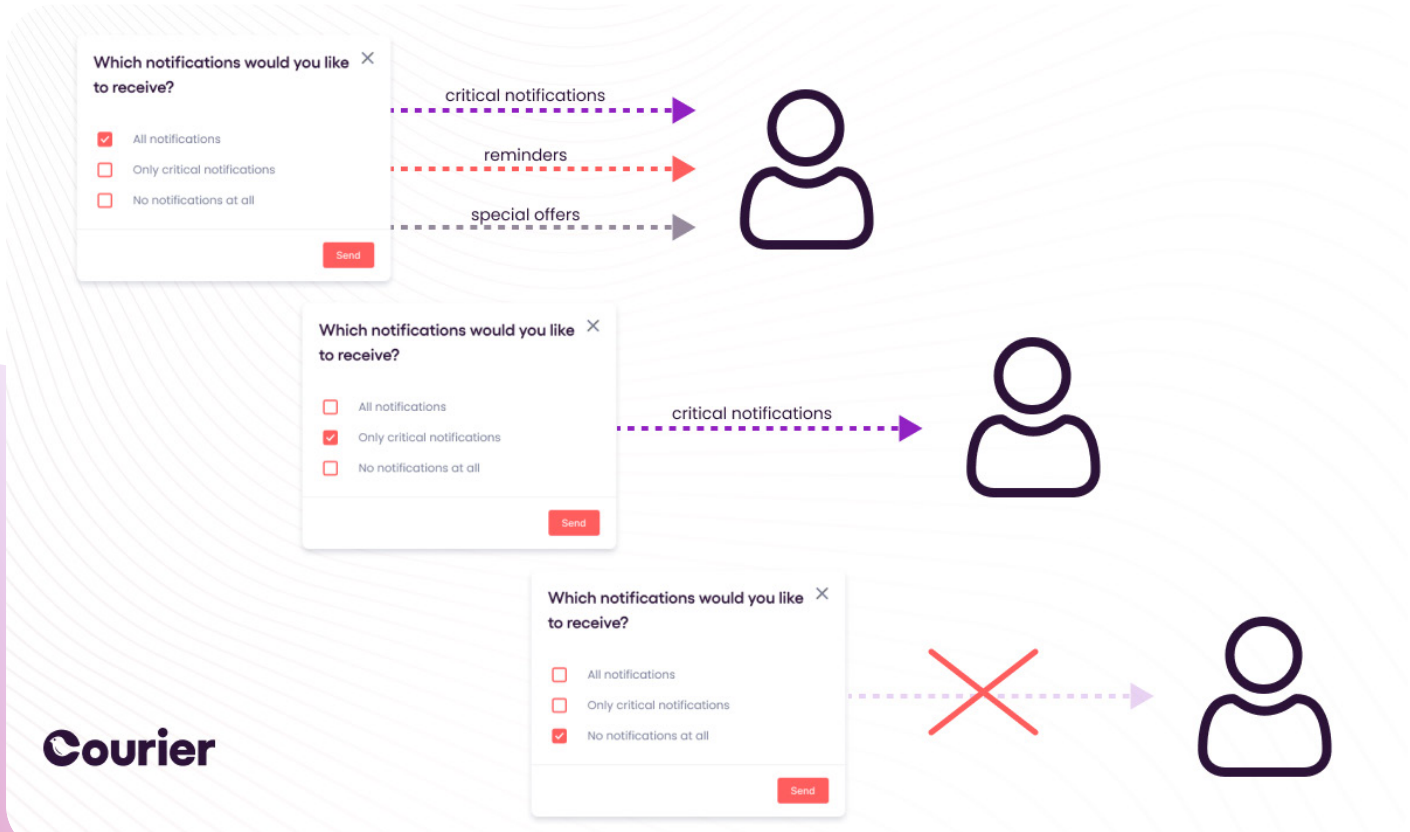


What are notification preferences?

Notification preferences are a way for users to configure how and when they would like to be notified of anything important happening within the product or service. For example, an event app may offer its users to configure whether they want to periodically see all upcoming concerts in their area, or only want to be notified when their favorite artists are playing.

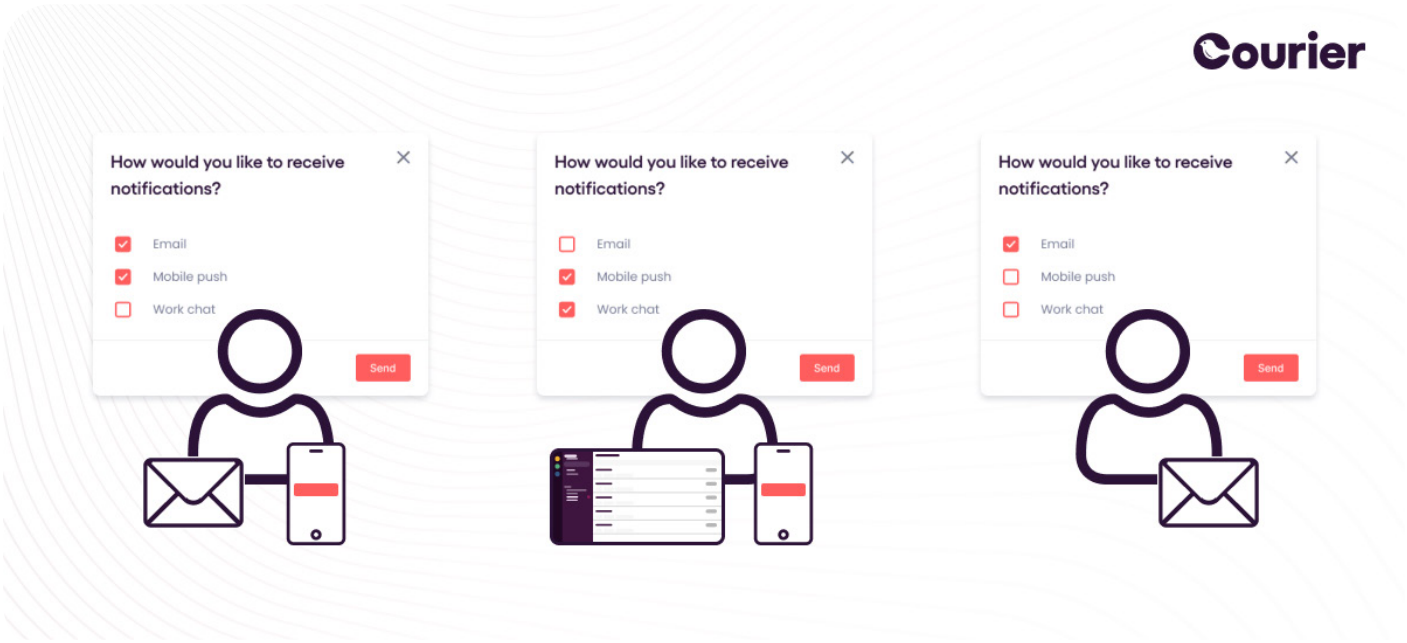
Notification preferences can be broken down into two categories: notification logic and notification channels.

Preferences related to notification logic influence when the app notifies its user. For example, a user of a local marketplace app might want to get notified about each new offer in their area, or just receive a summary of all offers at the end of the day — or not get any notifications at all about offers, perhaps because they are only selling and not buying at the moment.



Channel preferences influence how the notifications are sent — via email, through Slack or Discord, via an SMS message, or otherwise. Using the marketplace app example from above, a frequent seller might want to get instant push notifications for any new messages, while an occasional buyer might prefer to be notified via email only.

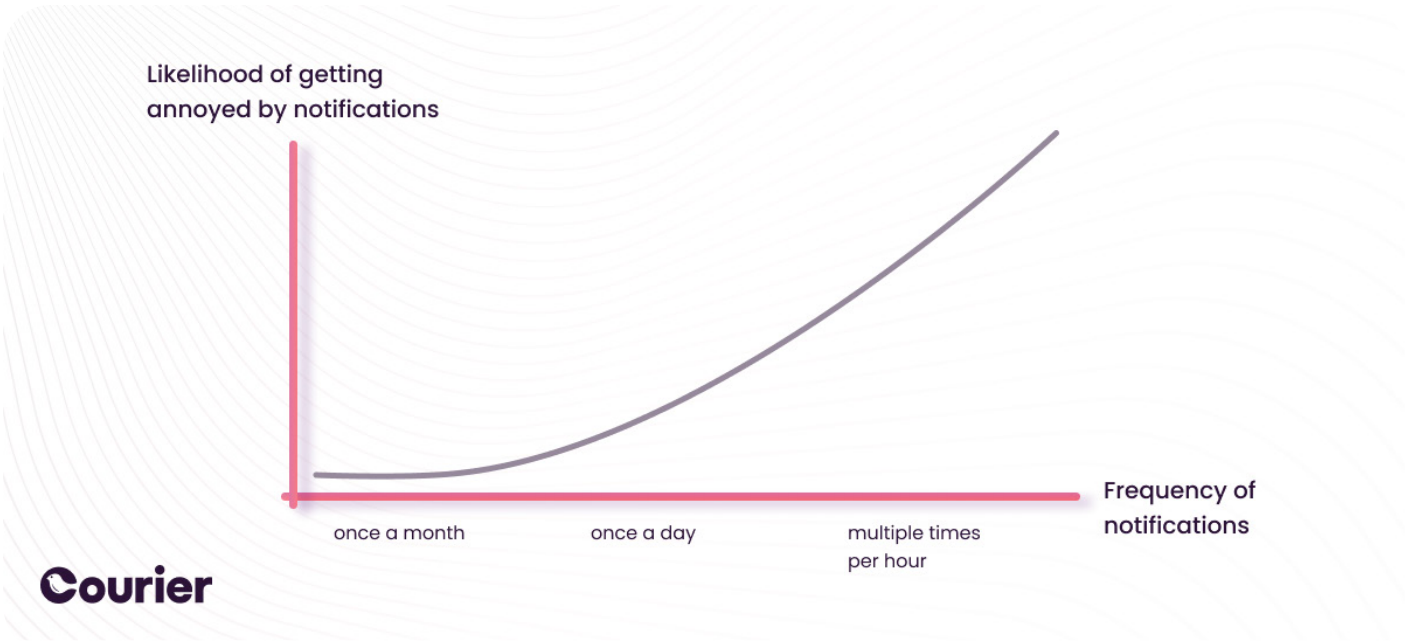
It's important for users to be in control of both the notification logic and the channels used. So, as the provider of a product or service, you need to make sure that your users can configure both if you want to offer them an optimal user experience.



Why adding notification preferences is a worthwhile investment

As the amount of notifications sent increases throughout the software industry, users are rarely given control over what they want to be notified about and when. Customers therefore increasingly resort to extreme measures — disabling all notifications for an app or service, or even deleting their accounts.

Product managers tasked with improving notifications sometimes believe that it's possible to build a standard notification system that suits most users. And while this might be true for occasional notifications, the one-size-fits-all approach is no longer suitable when notifying users frequently. Additionally, the power users of your product or service will likely be the ones most affected by an inflexible notification experience — and those are the users that you most need to keep.



When you give users control over which notifications they want to receive, they are much more likely to want to continue receiving your notifications and be engaged with your app.

Best practices for notification preference management

Before adding notification preference management to your product or service, you'll need to identify the preferences that would have the most impact for your users. The main challenge when determining which preferences to include is the **balance between complexity and flexibility:** you want the preferences to be powerful, but also easy to understand and use.

Below are some of the legal and practical factors you must consider when implementing your notification preferences, and some of the choices your users will expect to be able to make about their notification experience.

GDPR and CCPA compliance regarding consent

Compliance with regulations like the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) is a requirement for the vast majority of B2B software companies today, and notifications need to be compliant too. As part of notification preference management, you will have to collect and store the information on whether your users have provided consent to getting notifications, of what kind, and how that consent was provided.

In order to stay compliant with laws like GDPR and CCPA, you will need to make it possible for users to view their consent settings and give them the ability to withdraw consent in the future.

Storage and retrieval of notification preferences

Depending on the number of users you have and the number of notification settings you'll be implementing, you'll need to consider how the user preferences will be stored. Because the settings need to be taken into account for every event that's potentially relevant to a user, simply adding a database table for notification preferences might not be sufficient.

Work with your engineering team to understand how and when notification preferences will be accessed within your app and choose a data store for the preferences that is suitable for the task. You can learn more about the technical side of implementing notification preferences in The Developer's Guide to Building Notification Systems: Routing and Preferences.

User opt-in for different notification types

One of the largest sources of annoyance for users is getting both lifecycle (marketing) notifications and transactional notifications within the same channels and with the same settings. Ideally, applications should allow users to opt out of notifications that they're not interested in, while staying opted in to the ones they do want.

Configurable notification channels

Allow users to set their preferred channels for getting notifications. Some users will want to only get emails from your service; others will want more timely notifications through push.

It's ideal if you allow them to set preferences on channels for various notification groups — so that, for example, in a collaboration tool a user can get push notifications for mentions of their username, but get an email summary for all notifications that don't involve them directly.

User choice regarding notification frequency, including digests

A common mistake when implementing preferences is to offer only two choices for a particular topic: all notifications or none at all. Frequently, users need a middle ground. For example, a member of a Discord community might not want to get notified of every new message, but they potentially wouldn't mind a summary of messages.

A common solution in such situations is to offer periodic digests — often via email. Users can configure the time period they would like to get a summary for, such as daily or weekly. The service then collects all notifications generated during that period and sends them as a single overview.

User-configurable topics — all notifications versus critical notifications

There is more nuance than the basic distinction between lifecycle and transactional notifications. Each reasonably complex application or service will likely have various kinds of lifecycle and transactional notifications, and your users might be interested in only some of them.

Let your users configure topics they are interested in getting notified about via a given channel. These topics can be as broad as "time-sensitive notifications" or "urgent account-related messages," or more specific like "timely special offers."

Additional notification preferences that will delight customers

With the basics of notification preference management covered, let's have a look at a few more advanced options. These additional preferences, while not vital to core functionality, provide ways to make your application more convenient for users, and even surprise them with thoughtful functionality that shows you have truly tailored your product to its audience.

Allow users to specify multiple email addresses

Most services operate under the simplistic assumption that users have only one email address, but frequently that's not the case. For developers, for example, the email address associated with their GitHub account might be their personal email, while they also have a company-issued email address. Because of this, GitHub allows users to choose which email address to use for notifications.

If your application can be used in both work and non-work contexts, consider letting your users pick which email address, device, or phone number they want to use for notifications — it will make their experience more enjoyable.

Redirect notifications to another user

While you are on vacation, you might want someone on your team to periodically check for any critical notifications in your area of responsibility and escalate to someone else if needed. That's easier said than done, however: while it's possible to redirect emails with rules and filters, notifications that get sent to other channels like mobile push often can't be redirected to another user.

Does your product involve collaboration in a business context? If so, a feature for redirecting notifications might be a good fit for your user base.

Silence notifications at night

Many business communication apps try not to disturb users at night. If you decide to implement this feature in your notification system, don't forget about time zone support!

And some folks' biological clocks might work differently, so giving them an option to configure their own hours without notifications can be handy. Slack, for example, achieves this using its configurable Do Not Disturb mode.

Regardless of the preferences you want to add for your users, try to create a solution that balances complexity and functionality and, ideally, keeps notifications flexible while also being easy to use for all users. Specifically, this should be done in a way that doesn't overburden your dev team. In the next chapter, we'll look at how decoupling your notification code from the general codebase can save both developers and PMs time and nerves.

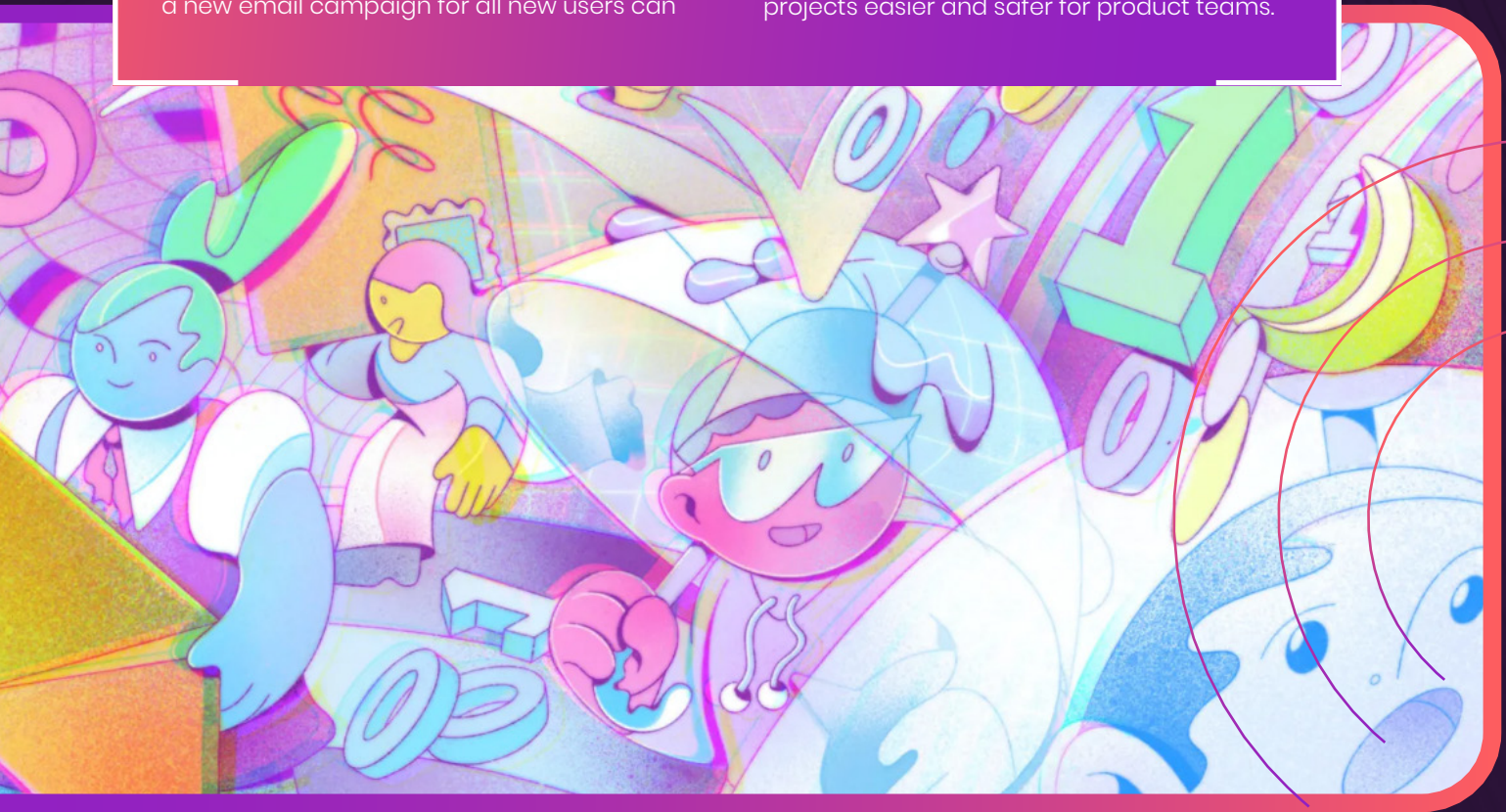
CHAPTER 3

How decoupling notifications from application code can empower PMs

As a product manager, have you ever had to say no to improving your app's notifications because of how complex the project seemed? If yes, you're not alone.

The reason is that changing notifications, in the experience of most PMs, requires a lot of work between engineering and product due to how closely the notifications are tied to the rest of the application. While tasks like adding a new email campaign for all new users can usually be done outside of the core app, sending more customized notifications can be complex, requiring changes to an application's source code.

Luckily, there's a trick to making changing notifications less difficult, and it's called decoupling. In this chapter, you'll learn how decoupling notifications from your application's codebase can make notification projects easier and safer for product teams.



Why PMs believe notification projects are hard

Usually, the task of improving an app's notification experience — like adding or changing notifications, or connecting additional notification channels — falls to the engineering team that's responsible for the main product. While notification projects tend to be high-value for product managers because of their high impact on the overall customer experience, they may not be a priority for engineering teams tasked primarily with implementing new features or ensuring application reliability. Notification projects also score high on complexity and risk at most companies, as each additional notification channel and customization requires additional engineering resources.

Project scorecards

	value	complexity	risk	decision
new feature	high	medium	low	implement next
app perfomance improvements	medium	medium	low	implement next
notification improvements	high	high	high	postpone

Courier

With so much complexity involved, product managers have to make a tough call: is it really worth taking on notification work when there are easier projects to work on with more tangible benefits? Because task backlogs frequently have other items in them that don't carry as much complexity or risk, it's logical for product managers to postpone complex notification projects — meaning notification projects get moved down the priority list, again and again.

The limits of common engineering solutions to notification problems

To try and make notification changes less difficult to implement, engineering teams often turn to open-source tools for their framework or programming language of choice. For example, those working with Ruby on Rails applications often use Rails' built-in email sending module, Action Mailer, which includes functionality for sending plain-text and HTML emails.

While tools like Action Mailer are well-maintained and convenient to use, they can't handle more complex tasks like expanding beyond the standard notification channels. If you want to add Slack or SMS notifications, for example, choices like Action Mailer can become a limiting factor, rather than an enabler of new product functionality.

The next logical option for a software team running into tooling limitations is to go with a fully customized notification system — but such a system brings its own challenges.

Even a custom notification system is a bottleneck for notification improvements

It may seem that the solution to the limitations of open-source tools is to simply build your own custom notification system. A custom notification system, however, is complex code that needs to juggle several components like the notification logic, supported channels, and templates for specific notifications.

The development process is slightly different for notifications than for the rest of the codebase, as it typically consists of frequent but small changes. But because it is embedded in the general codebase, any changes have to follow the steps of the software development process, which can quickly become cumbersome and could interfere with the ongoing development of your application.

For example, to update the wording or the design of a notification across all channels, the engineering team would have to:

- Change the HTML email template.
- Change the Slack notification template.
- Change the Microsoft Teams notification template.
- Test all three changes.
- Wait for code review on the changes.
- Deploy all three changes to production.
- Hand off to the PM for a final confirmation.

Every time a PM wants more changes implemented for notifications, the process has to start over again, even for trivial content adaptations. This adds additional overhead and takes your engineers' time away from working on more valuable application features and upgrades.

The result of home-grown notifications: they rarely get updated

Because of the additional delays introduced by the development process, the friction for implementing notification changes ends up being so high that PMs often choose to not implement notification changes at all. Instead of focusing on building the best possible notification experience, they settle for the default notifications that are built into their application during initial development. Changes to the notification experience are avoided as much as possible, causing it to become outdated.

There is a better way: how PMs can work with notifications without involving the engineering team

As discussed above, the way notifications are handled in most SaaS apps today — templates stored in code, notification functionality changes requiring code deployments — is ineffective and outdated.

Instead of having to go through the development process for each notification change, PMs and other non-technical stakeholders need to have the right tools to work with notifications directly. Below are the core practices that act as enablers for non-technical audiences.

Make templates accessible to non-engineers. Notification templates in the form of text, HTML, or programming-language-specific template files need to be located outside of the application code, and PMs as well as other non-technical teams need to have access and be able to make changes.

Make notification logic configurable without needing to re-deploy the app. The logic involved in deciding which notifications to send when should not be located in the application code, but rather sit next to it and be configurable without needing to deploy the application. The configuration can live in a text file, a web interface, or another config format. PMs should ideally be able to change the configuration themselves.

Build guardrails to prevent notification issues from escaping to production. Ideally, all notification functionality should be covered with automated and, possibly, manual tests so that the organization can be confident in the product's notification experience at all times. The areas to check include: notification text fitting within the display area; possible internationalization glitches; technical issues like app crashes resulting from changes in notification text length; and handling of unexpected characters if notifications include user input.

A key tool for allowing PMs to stay on top of their users' notification experiences is analytics. Analytics can help PMs find out what worked and what didn't, and make sure the rest of the organization can see the benefits for larger changes to notifications. In the next chapter, we'll look at how analytics can improve the notification experience.

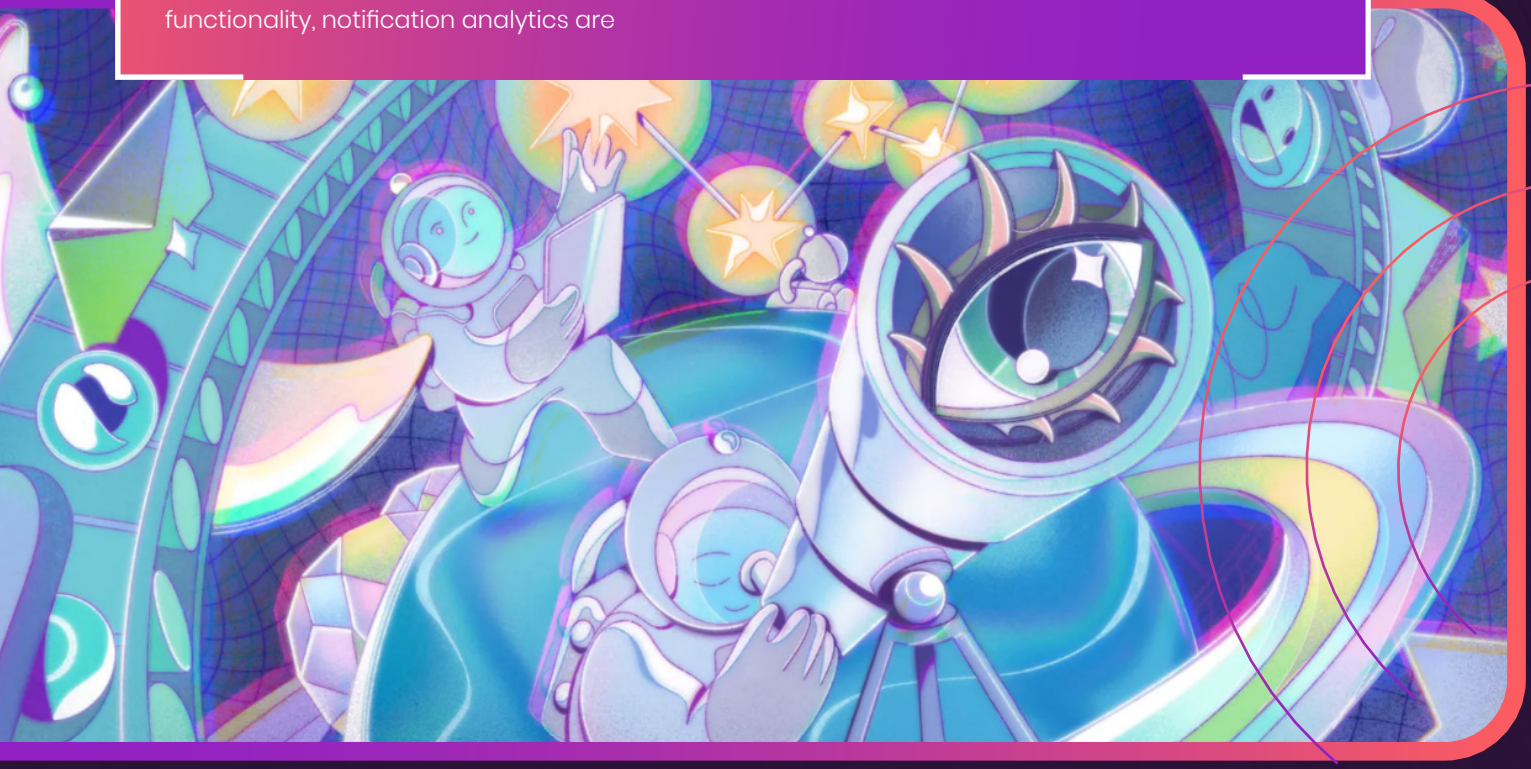
CHAPTER 4

Omnichannel analytics — the key to building better notification experiences

Web analytics are a powerful tool for businesses, and product managers rely heavily on web analytics data when making product decisions. Stats like website traffic, conversions, in-app events, or unique users are frequently used as indicators of business health, because they help identify whether the product is moving in the right direction.

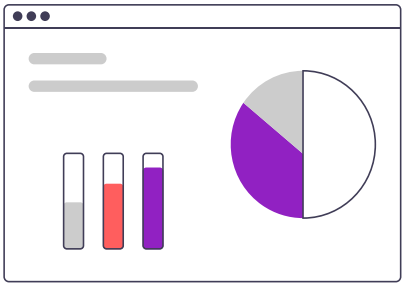
With notifications, however, analytics are a different story. While the notification experience is a key part of application functionality, notification analytics are rarely available, or limited at best. Without notification analytics, you risk making decisions about your notifications that don't fit with the real-world behaviors of your users.

This chapter explains why implementing analytics for notifications is difficult and how omnichannel analytics empowers PMs. It also provides some tips on making the best of available analytics data to design a more informed notification experience for your users.



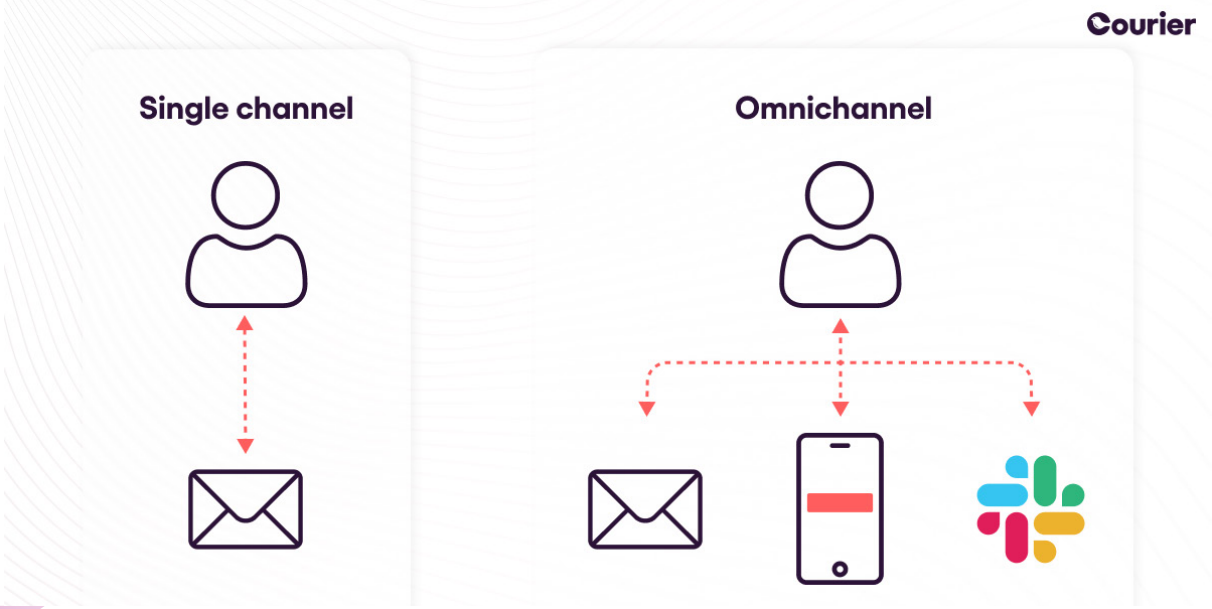
What is omnichannel analytics?

Single data points on their own are not very informative, but in aggregate, they can tell a powerful story. Analytics is the art of assembling raw data from your product in a meaningful way to gain insight into your users' real-world actions. For instance, you could aggregate event data collected in an app, and infer from it which of the app's features are most popular. You can then use those insights to propel your product further by expanding the features that your users like, and retiring those they don't. **Analytics helps you forecast and shape the future by understanding the past and present.**



Businesses communicate with their customers through multiple channels. Email, push notifications, SMS, and messages on social media complement more traditional means of communication like phone calls and letters. With that many options for reaching out to your customers, it's insufficient to look at only one channel if you want to gain insights about how a specific communication strategy is working.

If you use a multi-channel strategy to communicate with your customers, it's typically not helpful to look at those channels in isolation. A customer who ignores your newsletters in their email inbox may be doing so because they have already seen and liked your announcements on a social media platform. With omnichannel analytics, you look at your users' interactions with your product holistically, across channels, so that important interactions are not overlooked or misinterpreted.



Analytics for your notifications

Notification data means specifically how your users interact with your notifications. Do they click on an email as soon as they receive it? Do they swipe your push notifications from their home screens? By doing analytics on your notification data, you can understand which communication strategies work well, and which don't.

A central metric in notification analytics is the open rate: it quantifies how many users opened a notification within a certain time span of sending it. A low open rate means that you'll have to think about changing your strategy — perhaps by sending fewer notifications, using different channels, or updating your content. An analytics strategy that takes multiple channels into account can ensure that your open-rate data is accurate. But first, let's have a look at the specifics of notification data.

The difficulties of obtaining notification data

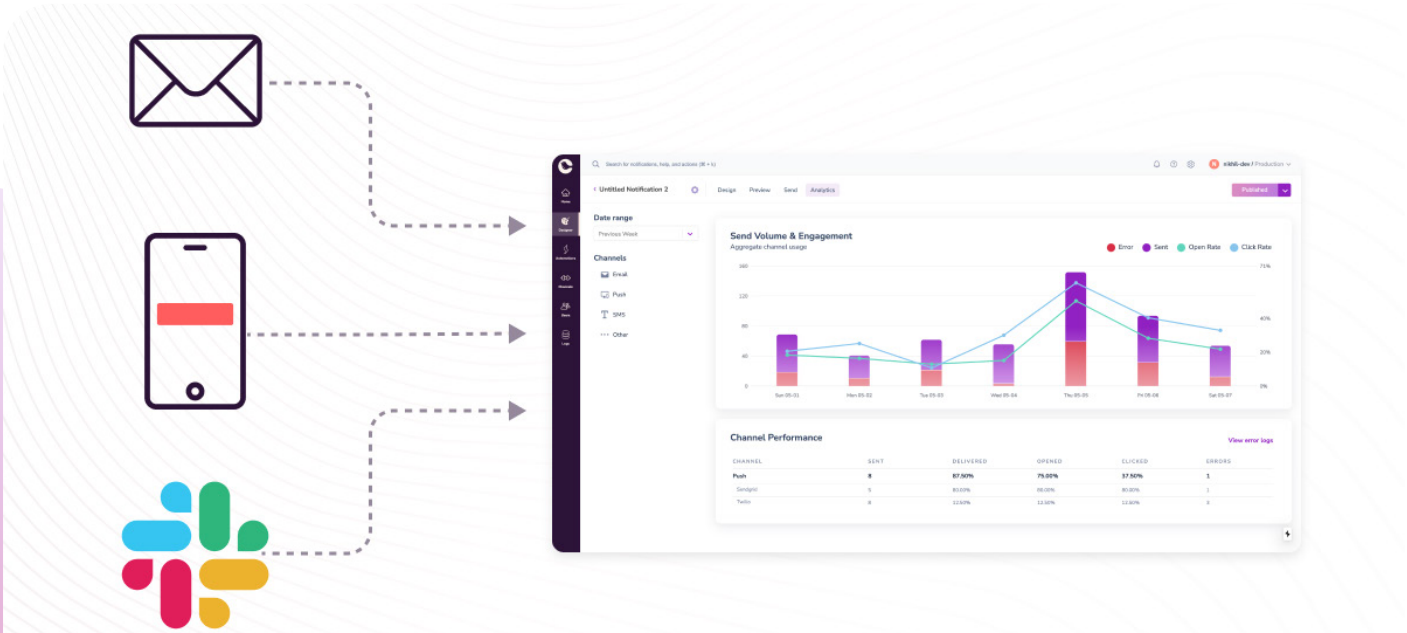
Data is the basis for all analytics, but it can be hard to get your hands on the kind of data needed to perform analytics on notifications. Apple's iOS, for example, doesn't allow developers to precisely track impressions of push notifications on their users' devices. With SMS messages, there is no analytics at all: you can know what messages you sent, but you don't have any guarantee that users received them.

Empowering PMs with omnichannel analytics

As everyone's notification experience becomes more personal, PMs will need to move to analytics that are grouped by notification content, rather than by channel. For example, all notifications about new mentions in a document app should be presented together in an analytics dashboard, regardless of whether the notifications were sent via email, Slack, or mobile push, depending on each user's preferences.

How to get omnichannel analytics for your application

In order to enable omnichannel analytics for notifications, PMs need to overcome a few hurdles. You will need to use notification tools that support data collection, move that data to a central location where it can serve as a basis for analytics, and be able to pull reports in an actionable format.



Implement analytics for each channel

The first step is to collect whatever data you reasonably can around notification usage, while respecting relevant privacy regulations and creating as little inconvenience to the user as possible. For example, when sending emails via a third-party email platform like SendGrid or AWS SES, the following data points can be collected:

- For each email notification sent: recipient, date, subject, attachments
- Open and click metrics
- Statistics on unsubscribes
- Statistics on complaints

For some channels, it will not be possible to collect direct data for notification interactions. In such cases, you will need to come up with proxy data points: that is, data that may not be directly what you need but that can be used to infer the metrics that you care about. For example, in the case of SMS messages, you can assume a specific open rate based on industry benchmarks, or insert a link which you can track clicks on. You can then calculate an approximate open rate throughout your customer base.

Some channels have well-established workarounds for tracking. In emails, for example, it's common to use web beacons, also known as tracking pixels. These are usually implemented as tiny images hosted on a third-party web server, and when a user opens the email, the email client loads that resource from the third-party server, which indicates to the sender that the recipient has opened the email.

Such solutions don't always work, however: users can disable images in emails, and iOS's built-in mail application has a setting that prevents such tracking from working. There are other tracking techniques, like adding user-specific information to all links in the email, so if a link is clicked, it is straightforward to know which user the interaction came from.

Ensure data quality

To generate trustworthy reports about notifications, you will need to be confident in the data the reports rely on. Implement periodic checks for data quality, and make sure that your proxy metrics remain reliable.

In addition, you should understand the limitations of your data — maybe there is a delay in the stats that some systems give you, or maybe there are sometimes empty rows in the data. You need to account for that ahead of time so you don't run into issues later.

Identify how you'll cross-reference stats between channels

The technical implementations of notification channels like email and SMS do not provide a mechanism for uniquely identifying their contents or monitoring user interaction.

To be able to cross-reference the same kinds of notifications across channels you'll need to create an identifier — for example, a unique ID number — for each notification. Such IDs need to be the same across multiple channels so that you can group them together later when generating reports.

Transform all data to a common format

In order to look at multiple channels together, you must have all the data available in a single report, dashboard, or spreadsheet. This is not always straightforward — different providers present data differently in their statistics — so you will need to transform all data into a standardized format, for example, through an ETL pipeline.

Once the data is transformed, you can use any reporting or business analytics system that you already use — for example, Looker, Tableau, or Excel — to look at all the data points together.

Group by notification identifier, topics, or notification types

Once all the data is in a single place and in a unified format, and you have implemented cross-references to identify notifications across channels, you can group data points by their notification ID. Going further, you can then group this data by topics or types of notifications. For example, you might visualize all lifecycle notifications on one dashboard, and all transactional notifications on another dashboard.

Using analytics to make better notifications

Having access to analytics for notifications is an achievement in itself, but remember that you're doing it for a purpose: to make your notifications better.

Now that you have data on notifications across channels, here are a few ways you can use the omnichannel data to identify opportunities to fix or improve your notification experience.

Identify opportunities for improvement

Consider **which existing notifications have a lot of engagement**. What can you learn from this? Is there any way you can replicate the success of such notifications in other parts of your notification experience?

In contrast, **which notifications currently have little engagement from users?** Maybe that's a sign that these notifications need to be reworked or removed, or that you're sending too many notifications overall.

Another question that can help identify opportunities is **which notifications do your best customers receive the most?** The "best" part here is up for interpretation according to your business goals — maybe it's top customers by revenue, or by their level of engagement, or by another metric. Is there a connection between receiving these notifications and the value they get from using your application? If yes, how does that translate to the rest of your user base?

What trends can you monitor?

In addition to looking for specific answers in your notification data, sometimes it may be helpful to monitor trends so that you proactively see if things start to change. A few trends that you can look out for are:

- **The relationship between notification interactions and user growth.** If you add more users but they interact less with notifications, it might suggest that the users are less engaged, or that the notification experience is no longer a good fit for your newer customer base.
- **Cohort analysis.** Are newer users responding to notifications better or worse than users that have been around for a while? By looking at notification stats in the context of user cohorts, you can quantify the effect that notification changes have on new user onboarding.

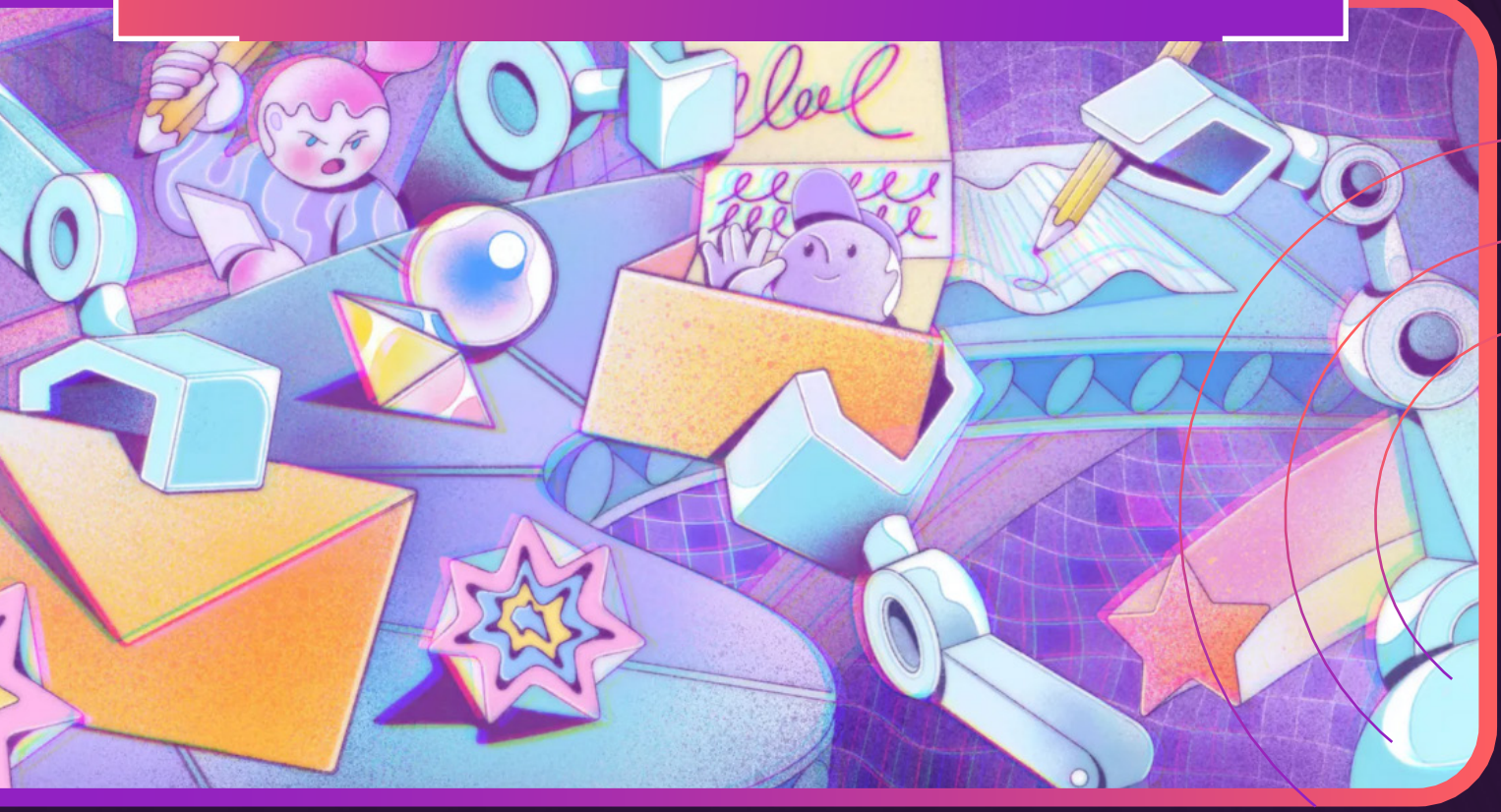
Having a solid analytics workflow around your notification data will help you gain valuable insights into how your users experience their notification journey with your product or service. You can then use that information to craft the ideal notification experience for your users. While manually building a bespoke notification experience can be cost-prohibitive and time-consuming, automation tools can make this process almost effortless, as we explore in the final chapter of this ebook.

CHAPTER 5

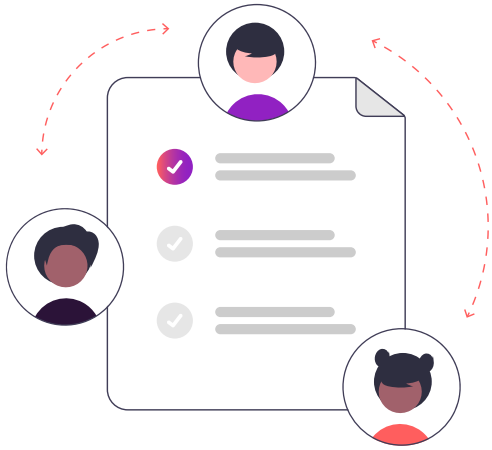
How to optimize your notification logic with automations

From the user's viewpoint, the line between being engaged by relevant, timely notifications and feeling harassed can be thin. Chapter 2 of this ebook looked at giving users agency in defining how and when they want to be notified, and the huge positive impact this can have on how they perceive the notification experience. But there's much more that can be done through automation.

Using automation tools, your developers can implement the kind of thoughtful, tailored notification experience that your users want. In this chapter, you will learn how low-code and no-code automation tools help you build a better notification experience while taking the burden of implementing the complex logic off your engineers' shoulders.



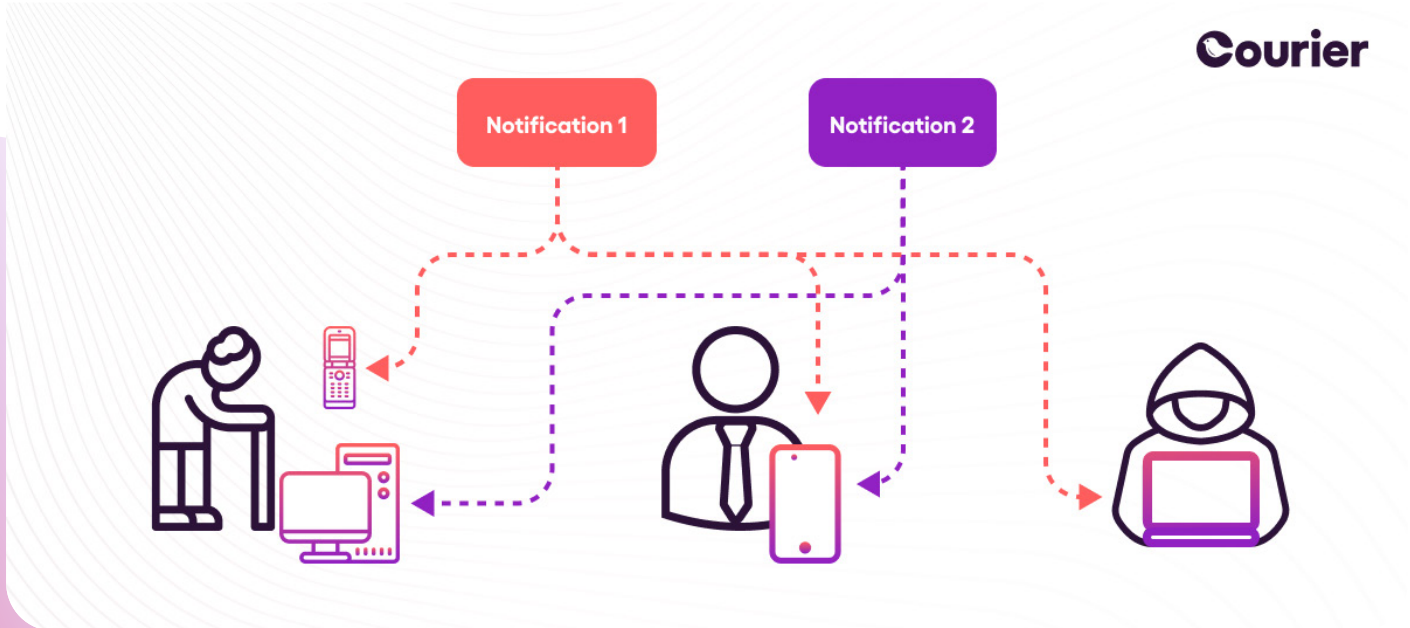
Customization through automation



Automated functionality like audience filters, complex workflows, and channel routing helps you craft notification experiences that your users will happily engage with. Historically, these features come at a cost — the more customizability you want to include in your notification experience, the more resources are required to implement and maintain those features in your codebase.

As a product manager, you need to think about how you can enable your developers to set up a notification logic that provides your users with a bespoke and smooth notification experience — without having to significantly rework your codebase to support every customization you'll potentially require. Integrating existing tools with robust automation is key to achieving this.

The first step towards automated customization is to understand what your users want and need. To that end, it's useful to analyze how your users perceive the messages you send them — both through data analytics, as we discussed in the last chapter, and by consulting your users directly. You can then use that knowledge to send them helpful tips at the right time, as opposed to just sending out blanket notifications.

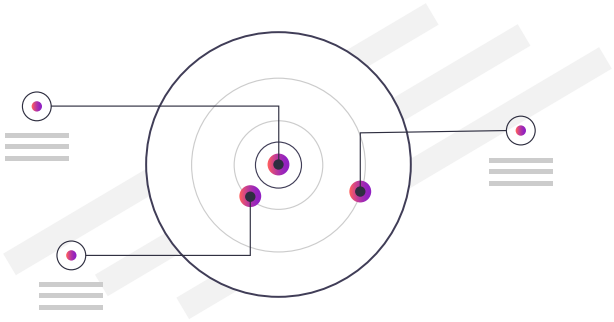


Once you have established your requirements, you need to choose and implement the right tools. Automation tools come with dynamic configuration capabilities that let you tailor your notification experience to the individual needs of different users. By integrating finer-grained customization into your notification logic, you can ensure the relevance of your notifications to your customers. Here are a few examples of a customized notification experience:

- **Targeted announcements:** Rather than sending out feature announcements indiscriminately, they only arrive in the inboxes of customers whose user profile signals that they might have an interest — and who are not already using the feature.
- **Multi-channel notifications:** Critical alerts should reach the right person at the right time — with multi-channel notifications, the notification logic picks the most suitable channel and sending time to ensure that users see the message and can react accordingly.
- **Tailored content:** To avoid spamming users with content they have no interest in, organizations can tailor the subject matter, channel, and timing of their digests so that the people who receive them will actually want to engage with them.

Strategies for automation

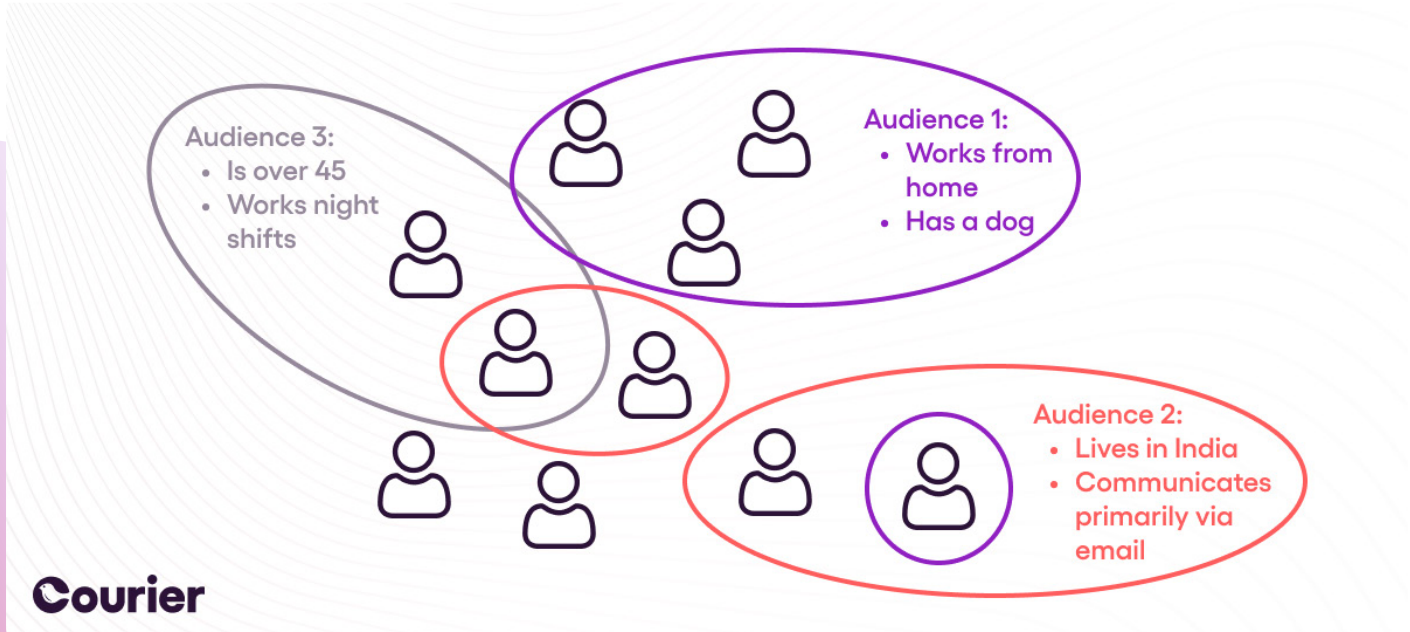
To make sure that your users get the best notification experience possible, your developers should employ multiple customization strategies to achieve a much more personal feel. The three core strategies behind user notification customization are explained below.



Audience filters

Many notification systems use lists to organize their user data. Lists allow you to put user profiles into cohorts, according to different criteria — for instance, their profession or their place of residence. You can then use these lists to send out notifications that are targeted towards a specific group and thus feel more relevant than a one-for-all message. But do you know what's even better than lists? Audiences!

With audiences, you can define user cohorts dynamically based on a set of criteria. Rather than working with static lists of users, you can quickly filter your users' profiles to assemble an audience. For instance, you'd only want to send out notifications about a holiday-related sale to people located somewhere where that particular holiday is celebrated. Since your users' location data is subject to change, it makes sense to use a dynamic audience filter rather than a static list.



Another useful basis for filtering is usage data. For example, sending instructions for an established feature of your application may be a godsend to new users, but would be an annoyance to those already proficient with your product.

Audience filters are a prime example of how user data, when collected and stored in a safe and GDPR-compliant manner, can be used to improve your users' lives by offering them a more tailored and meaningful notification experience.

Multi-channel routing

When and where are two factors that contribute immensely to how a user perceives the notification experience. As we've highlighted throughout this ebook, receiving a message at the wrong time or through the wrong channel can create a lot of frustration for users. Choosing the right channel and time, on the other hand, can go a long way towards providing a satisfying notification experience that integrates seamlessly into your users' lives.

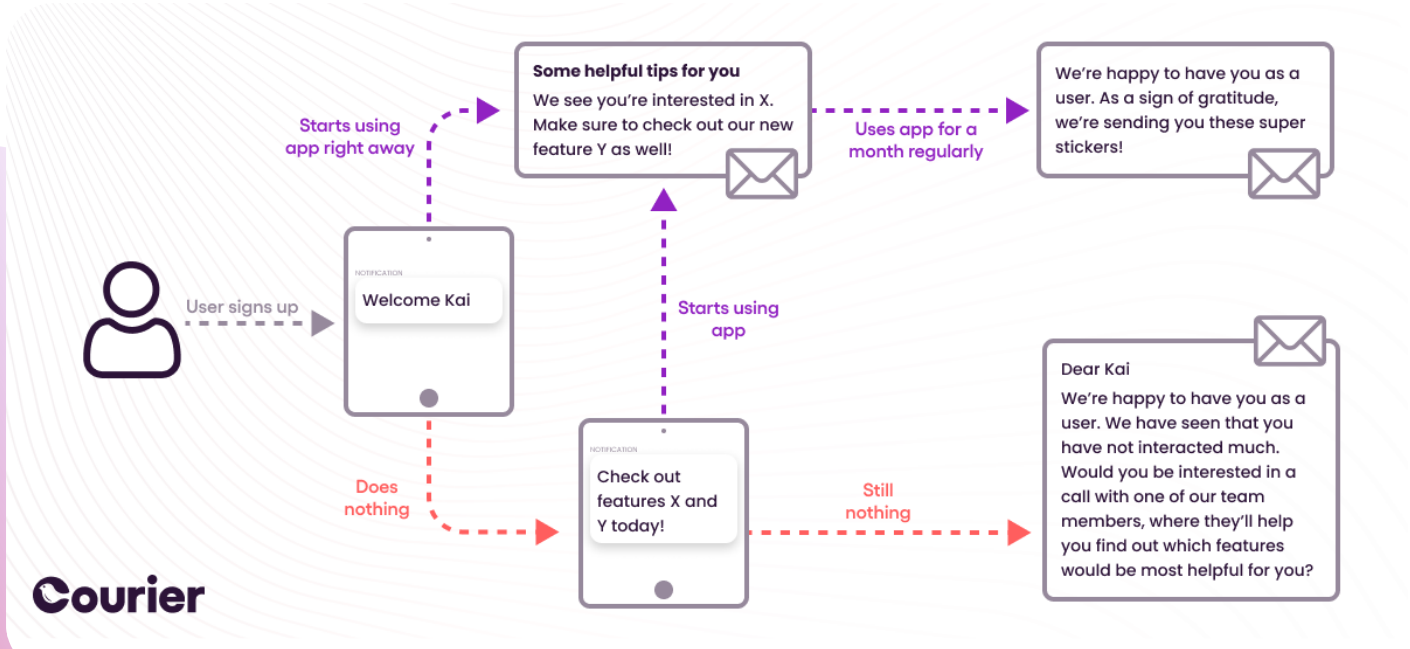
In addition to offering its users options for granular preference management, your organization can apply its learnings about which channels work best under which circumstances to build an intelligent routing system.

With multi-channel routing, you can specify a ranking order for notification channels. For example, you can show a notification in the browser first, and if the user isn't online, or has disabled in-browser notifications, you can send them a push notification to their phone.

Complex workflows

Notifications should rarely be statically defined. Rather, you should identify where your users are in their product journey, and send them messages that support, help and encourage them. Workflows help you respond to the different paths a user's journey with your product might take.

As an example, consider someone signing up as a new user to your service. You'll probably want to send them a friendly welcome message through their preferred notification channel. Then, if your new sign-up doesn't interact with your application within a certain time span, you might want to opt for a gentle reminder by sending them a push notification or Slack message that details some first actions they could take. Finally, you could reach out to them via email (perhaps after a week or so of inactivity) and suggest that they schedule an appointment with a team member to address the issues that are keeping them from using your product. This gentle escalation encourages your new user to engage without bombarding them with alerts.



Thanks to triggers and other notification logic, you can build complex workflows that allow you to match the different experiences your users might have with your product. Such complex workflows are an example of how automation can actually provide a more personalized feel than mainly static notification procedures that were implemented manually.

When does it make sense to send notifications manually?

Automation should be at the core of your notification functionality. As the number of users grows, it becomes much harder to manually implement the notification logic required to cultivate a steady and happy user base.

Nevertheless, there are cases when it makes sense to send notifications without the help of an automation tool. You primarily need manual notifications for one-time events — such as device login notifications, password resets, product updates, and company announcements.

Notably, such notifications are relatively independent of your users' actual product journey, as they are determined by what your organization needs to communicate to its users regardless of their notification preferences. For example, a user must be made aware of a potential data leak irrespective of their notification preferences. However, this type of notification should be few and far between. The focus of notifications should always remain the user's experience with the product — and everything related to that can actually be automated.

Make your product more fun, personal, and engaging with tailored notifications

As a product manager, your job is to guide your application development to meet your users' needs, while weighing the technical costs of implementing advanced features. This ebook has talked about the various ways in which you can shape and improve the notification experience that users have with your product. We've discussed how an ideal notification experience, which integrates seamlessly into your users' lives, can vastly improve their overall product journey and lead to higher engagement, as well as higher retention rates.

By separating your notification logic from your general codebase, offering granular preference management to users, and applying insights from data-based analytics, you can build an even more customizable and intelligent notification infrastructure powered by automation, and implement a smooth, pertinent notification experience.

Courier achieves all of this for your business — providing multichannel notifications with user preferences at the core, and omnichannel analytics for the best possible insights into your notifications' effectiveness. Notification logic is abstracted away from your codebase, so that your notification methodologies can advance with your understanding of your users, without causing drag on your development process.

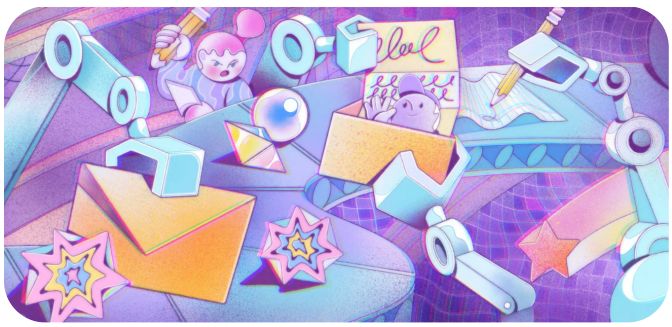
Try out Courier today and see just how easy it is to level up your users' notification experience.

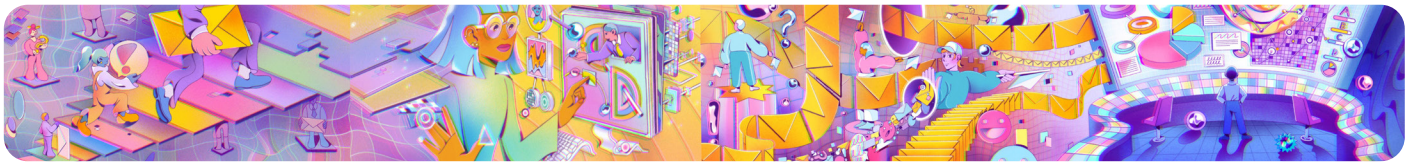
Artist's Statement

When I was approached to illustrate The Notifications for Product Managers, I immediately knew I wanted the imagery to link together and create a grand narrative that sort of mythologizes the design process that's being described. As someone who's very drawn to conceptual depictions of cyberspace and digital worlds, making sure that the Courier Dream Land felt vast but still full and teeming with warm human presence became the main priority.

Rebekka Dunlap

https://rrebekkaa.com/






Courier
www.courier.com