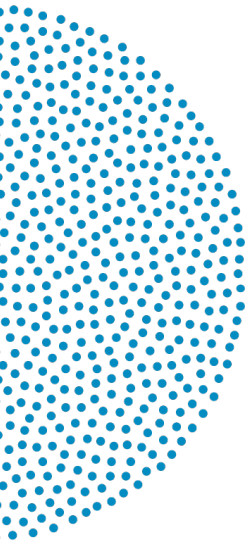
Build an onboarding engine for product and customer success

Five best practices for product teams

pendo

User onboarding in B2B applications has become a critical function. The growth of SaaS software has dramatically reduced switching costs for customers and shifted purchasing power away from centralized IT teams to individual teams and lines of business. User satisfaction is established very quickly in the customer lifecycle, and is difficult to improve after a bad first impression. When onboarding is effective, it can help new users quickly become proficient in an application, it can help them perceive value more quickly, and it can lead to higher levels of satisfaction. A poor onboarding experience can exacerbate usability issues, turn off users, and set accounts on the path to churning. This guide walks through some of the important considerations, and best practices that product teams should leverage when developing both their onboarding processes and in-application experience



Why are you interested in onboarding? – the user lens and goal setting

Before building out or revising an onboarding experience in your product, it's important to have clear objectives. Onboarding initiatives are sometimes hastily put together in response to internal fears—fear that users don't understand how to use the product, fear that they won't be able to get value out of it. This mindset can lead to poorly designed experiences, and a temptation to load too much content and too much learning material into an onboarding program. Instead, onboarding should be built around the user as an extension of their experience in the product. It should be noted however that even good onboarding won't make up for a confusing, or poor user experience.

Onboarding projects should begin with a couple of key questions:

- What are the key things that a user needs to learn in order to reach a minimum level of proficiency in the application?

- What settings or set-up elements need to be in place for users to access those key functions?

Companies should address these questions in 2 ways:

- **Onboarding process:** This can happen within and outside of the application. This process is a specific set of steps needed to address the settings or set-up elements. These elements may be completely self-service on the part of the users, but teams should document and understand what help mechanisms are in place and which people (whether Sales, Support, or Customer Success) are responsible for delivering the help.

- **Onboarding experience:** This typically happens within the application, and involves the instruction mechanism that is used to educate users about the key features they need to learn. This experience is typically owned by the Product Management and UX team with input from Support.

The two components should be highly complementary and developed in parallel. The key to success here is to keep the user perspective in mind, and more specifically the "minimum level of proficiency" measurement. If the process is too internally-driven, there can be a temptation to pack too much education into an initial experience. Focus only on what is critical for initial success. Progressive learning can come later.

Onboarding process, stakeholders, and a framework approach

For new customers in the B2B world, onboarding is often more encompassing than just an in-app training experience. It is an entire process that covers the initial setup, the installation and any settings that are required for the customer to begin using the application. As we discussed earlier, this process should be documented and studied along with any in-app onboarding experience. Depending on the industry and use case the steps in the process might be entirely self-service on the part of the user, or they might be driven entirely by various teams on the vendor-side. Most likely, it is a mixture of both. In any case there are usually multiple stakeholders in the organization that have some level of responsibility for each of the elements in the process.

A best practice when documenting the process is to clearly identify the steps in the onboarding process along with the responsibility level of the various stakeholders. One good way to do this is through a Responsibility Asset (or RASI) matrix. As an example, consider the following simplified onboarding process:

1. **Installation / Implementation–** Create client user accounts and provision storage. Establish integrations with other applications and systems of record.

2. **First 90 Days–** Provide guided training and support to ensure successful client uptake and continued usage of the application.

3. **Ongoing Support–** Offer technical support for any additional issues the client encounters during ongoing use, as well as for new individual user training on the client account.

4. **Renew & Expand–** Schedule client engagement and negotiation around renewal. Identify and close expansion or upsell opportunities.

In each of the described steps various roles including Sales, Services, Support, and Customer Success have some level of responsibility over the outcome. The same process deployed in a RASI matrix might look like the following:

	SALES	SERVICES	SUPPORT	CS
INSTALLATION / IMPLEMENTATION				
KICK-OFF CALL	ACCOUNTABLE	INFORMED	SUPPORT	SUPPORT
ACCOUNT SETUP		RESPONSIBLE	INFORMED	ACCOUNTABLE
INTEGRATION CONFIGURATION		RESPONSIBLE	INFORMED	ACCOUNTABLE
VALIDATION & TESTING		SUPPORT	RESPONSIBLE	ACCOUNTABLE
FIRST 90 DAYS				
INITIAL TRAINING		SUPPORT	RESPONSIBLE	ACCOUNTABLE
DEDICATED SUPPORT			ACCOUNTABLE	INFORMED
WEEKLY CHECK-INS			INFORMED	RESPONSIBLE
ONGOING SUPPORT				
INCIDENT-BASED SUPPORT			RESPONSIBLE	ACCOUNTABLE
QUARTERLY BUSINESS REVIEWS			INFORMED	RESPONSIBLE
RENEW & EXPAND				
VALUE ASSESSMENT	INFORMED		INFORMED	RESPONSIBLE
OPPORTUNITY EXPLORATION	RESPONSIBLE		INFORMED	INFORMED
PROPOSAL	RESPONSIBLE	INFORMED	INFORMED	ACCOUNTABLE
NEW ROLLOUT	INFORMED	RESPONSIBLE	SUPPORT	ACCOUNTABLE

Working through a matrix like this with all of the key stakeholders helps to codify the relationships and responsibilities to each phase of customer onboarding, and ensure that no activity in the process is dropped.

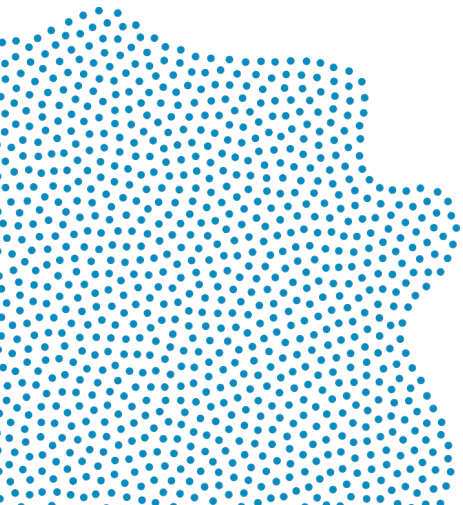
The onboarding experience and progressive disclosure

The onboarding experience takes place in the application itself and is a critical component of the overall process. This is where new users will begin to experience the application, and where they will form first impressions. Users can easily be too overwhelmed to move forward when presented with an “empty state” of the application—no activities or work accomplished and an unknown array of options. Even the best training won’t be effective if there are too many items to learn about at once. Enter the idea of progressive disclosure.

Rather than thinking about the onboarding experience as complete user training on every single function, instead think of the experience as a filter. An experience where all of the options are initially culled down to the ones that are most important to making the user quickly successful. In this environment the user has only a few functions to learn and can begin to use them immediately to drive some sort of direct feedback or result in the application. They are not distracted by all of the other features of the application that might be initially available to them. And as tasks are mastered, more and more of the application’s functions become available and are worked into the onboarding experience. This ensures that the user has measurable successes to achieve at each level of learning.

As a simple example, consider a procurement solution—large scale procure-to-pay applications have a massive array of features geared towards many different user roles in the organization. Attempting to educate a new customer about all of them at once would be overwhelming, difficult, and likely leave the user with a negative impression about the usability of the application. A progressive disclosure approach would look at a specific activity in the application, such as adding a vendor into the system, and only show features related to that activity when a user first enters the application.

The key to a progressive disclosure approach to the onboarding experience is a clear understanding of what defines a minimum level of proficiency in the application, and, more importantly, how that standard changes across different cohorts of users.



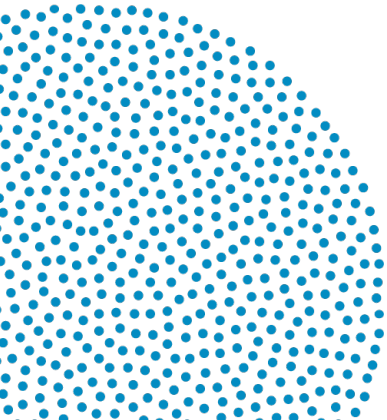
Practice #1: Tailor the onboarding experience for user cohorts

The challenge with progressive disclosure is that the activities that make up minimum proficiency are likely to be different for different cohorts. Going back to the procurement software example, a buyer will have a very different set of activities in their initial task list than an accountant with auditing responsibilities. In fact, the core feature set may not overlap for these users at all. If the onboarding experience is going to limit functions that are initially shown to a user, it will need to be significantly customized for each major cohort.

In our procurement solution—the process of adding a vendor to the system is a key function that a user in a "buyer" role would want to know. However, it is unlikely that a user who is a facilities manager would ever use that feature. He or she would be much better served by an onboarding experience that starts with how to place an order. Context is critical to keep users engaged within the onboarding process. Before mapping the process, think through the following considerations:

- What are the different user cohorts that will be using the application?
- How are they defined– are they role based, time based (i.e. started using the application in a particular quarter,) account based, or defined through some other measurement?
- What are the key features and functions of an application that each cohort needs to know in order to be successful? Are they the same?
- How much user context is available when a new user logs in for the first time?

Thinking through these questions will help to determine how much of the onboarding experience should be customized, and the extent to which it can be customized. A first step may simply be to determine how to get more user context upfront—perhaps by polling users when the first come into the application.



Practice #2: Differentiate between new users and new accounts

Another important consideration is the new user versus new account relationship. Typically in a B2B application, a single account or customer will have multiple users associated with it. If the onboarding is for a new account then there is likely a larger amount of setup associated with the initial onboarding. Account creation, integrations, or other more systematic steps may be required before the user can begin exploring the features of an application.

But, what about a new user at an existing account? This is another risk associated with a "one-size-fits-all" onboarding approach. If an account is already established, then a new user is less likely to encounter an "empty state" problem. The application will have recorded data, activities, and workflows of other users in the account. For a new user in this scenario, the onboarding process should be streamlined. Initial setup is not a concern, and the user can dive right into specific feature-based learning activities. This type of user context is readily available within the application, and should be used to tailor the onboarding experience.

Practice #3: Structure the learning path to user needs and learning styles

In the same way that the best onboarding experiences tailor the content based on user context, they also adjust the delivery of the content based on the user's learning style. The end goal is always to end up with proficient users, but the path to get there may vary. Product teams should think through different ways that users might progress, and be encouraged to complete, the onboarding experience. A couple of different ways to structure the onboarding content can include:

- **Snackable" or highly modularized content:** The idea here is to break up the onboarding experience into a number of small modules that don't necessarily run in sequence. Show users a quick "win" without working through a complete application setup, and then allow (and remind) users to come back. A good example of this is LinkedIn. After completing the profile basics, users are allowed to begin exploring the platform. Periodically, LinkedIn reminds them of the "completeness" of their profile and encourages them to go back and add additional information.

- **Gamification:** This approach uses elements of gaming or scoring to encourage users to complete an onboarding process. In this approach, the onboarding can include certain levels, scores, and opportunities to earn rewards, or "unlock" new features. Think again to LinkedIn's profile completeness score. Users get both a numerical score, and a named level (i.e. "All Star") based on onboarding progress. For users within an account, the scores could even be shared publicly using a leaderboard approach to encourage all users to progress through onboarding.
- **Progression:** This is probably the most common approach. Here tasks are arranged sequentially, showing the progress (and any remaining steps) throughout the process. The "progress bar" or a similar concept allows users to understand where they are, and how much more onboarding content they can expect before they are "turned loose" in the software. Some level of progression should always be used, regardless of how the onboarding content is structured.

The best approach to structure content is completely dependent on the learning style of the users. In some cases there may be a clear approach for some applications that works best—for example, most tax preparation software relies on straight progression, but sometimes a mix of the different approaches works best.

Practice #4: Let users leave and come back

Ultimately, a certain number of users are going to want to opt-out of the onboarding experience. Product teams should be aware of, and plan for this. Without an option to opt-out, these users will quickly become frustrated with the experience. However, just because a user dismisses an onboarding guide when they first come into a product does not mean that they are uninterested in some sort of structured learning.

Often these users can be re-engaged in the onboarding experience, either when they have more time for training or realize that they've hit a wall in the product. Good onboarding design makes it easy for users to pick-up onboarding guidance regardless of where they are in the application, or how much time has elapsed since they began to use the product. A common approach is to integrate and trigger the onboarding experience from within the help or support menus of the application.

Practice #5: Treat onboarding as a distinct user experience

It's important to remember that onboarding is at some level a UX problem. It is fundamentally a user experience that is both a part of, and yet distinct, from the overall application. Onboarding design should be approached and tested using normal UX practices. This means starting with a customer journey map or story map. This allows the Product team to really reflect on what the users goals are, how they interact with the application, and where they would likely encounter challenges. This helps to identify the key points or functions that should be exposed in the onboarding flow.

Another element of UX practices that should be incorporated into onboarding design is user testing. It's difficult to predict the exact path that a user will take through the application, or how they will consume an onboarding experience. Observational user study provides insight into how effective an onboarding experience is, and even more importantly, how engaging it is. It can also help provide insight into the likely learning style of users, and provide direction on how best to structure the content. Given that the onboarding experience is often the first thing a new user interacts with, testing this experience can and probably should be given preference over other aspects of the user experience.

ONBOARDING ISN'T SIMPLE

If nothing else, the key takeaway should be that onboarding isn't simple. There are a number of best practices discussed in this paper, but the ones that work best for a particular application and a particular set of users will vary. There is an optimal mix of offline process, in-application guidance, customization, and progression that can maximize customer success, but the right mix won't always be readily apparent. Product teams must be open to experimentation and trial and error.

Try different types of content, try different progression paths, and test the boundaries of customization. Each of these experiments provides valuable learnings and brings the optimal experience into better focus. Remember that onboarding is a critical part of the user experience and treat it accordingly. Test and measure how users interact with onboarding modules. A/B test different options and then refine. In the same way that a successful product is a constant work in process, so is successful onboarding.

Pendo is a data driven platform for product engagement that enables companies to improve on boarding, understand product usage and help retain customers. Pendo combines enterprise guidance and user insights to enable product teams to understand and influence their customer's experience. Pendo installs in minutes with no coding required.