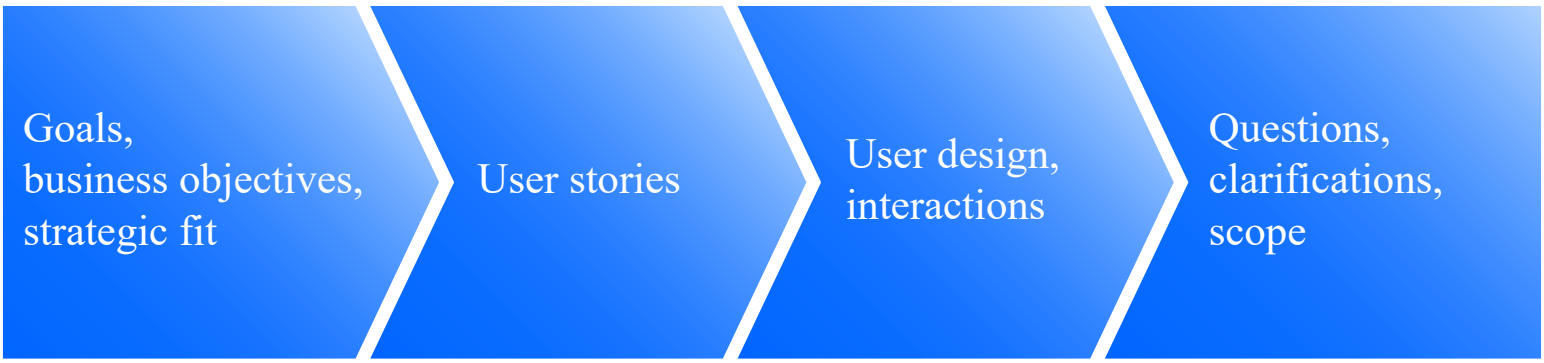
Building a great product requires tons of research and comprehensive planning. But where do you start? Product managers often start with a **product requirements document** (PRD).

A product requirements document defines the product you are about to build: It outlines the product's purpose, its features, functionalities, and behavior.



Next, you share the PRD with (and seek input from) stakeholders - business and technical teams who will help build, launch or market your product.

Once all stakeholders are aligned, the PRD serves as a compass, providing clear direction toward a product's purpose while creating a shared understanding among business and technical teams.

Gathering requirements in an agile world

What does the requirements gathering process look like in an agile world? It sounds tricky - and it is. But don't worry. At Atlassian, we know all about creating PRDs in an agile environment. Here are a few things to keep in mind:

Agile requirements are a product owner's best friend. Product owners who don't use agile requirements get caught up with spec'ing out every detail to deliver the right software (then cross their fingers hoping they've spec'ed out the right things). Agile requirements, on the other hand, depend on a shared understanding of the customer that is shared between the product owner, designer, and the development team. That shared understanding and empathy for the target customer unlocks hidden bandwidth for product owners. They can focus on higher-level requirements and leave implementation details to the development

team, who is fully equipped to do so – again, because of that shared understanding. (Boom.)

Creating a shared understanding among teams

If you're excited by the idea of a shared understanding, but haven't a clue how to create it, try some of these tips:

- When conducting customer interviews, include a member of the design and development teams so they can hear from a customer directly instead of relying on the product owner's notes. It will also give them the chance to probe deeper while the topic is fresh in the customer's mind.
- Make developing and using customer personas a team effort. Each team member has unique perspectives and insights, and needs to understand how the personas influences product development.
- Make issue triage and backlog grooming a team sport as well. These are great opportunities to make sure everyone is on the same page, and understand why the product owner has prioritized work the way they have.

Want to give it a try? Sign up or log into Confluence >>

Create a customer interview template to document your customer insights. Follow our tutorial to get started on conducting valuable customer interviews:

- Create insighful customer interview pages
- Turn info into insights with the Customer Interview Pyramid

> When the team and product owner share the same mindset, you don't need ultra-detailed requirements.

> ANTI-PATTERNS TO WATCH FOR
>
> - The entire project is already spec'd out in great detail before any engineering work begins
> - Thorough review and iron-clad sign-off from all teams are required before work even starts
> - Designers and developers don't know when requirements have been updated

> - Requirements are never updated in the first place (because everyone signed off on them, remember?)
> - The product owner writes requirements without the participation of the team

Ok: you've discussed a set of user stories with your engineer and designer. Gone back and forth, had a few whiteboard sessions, and concluded there are a few more dimensions you need to consider for this feature that you are working on. You need to flesh out some assumptions you're making, think deeper about how this fits in the overall scheme of things and keep track of all the open questions you need to answer. What next?

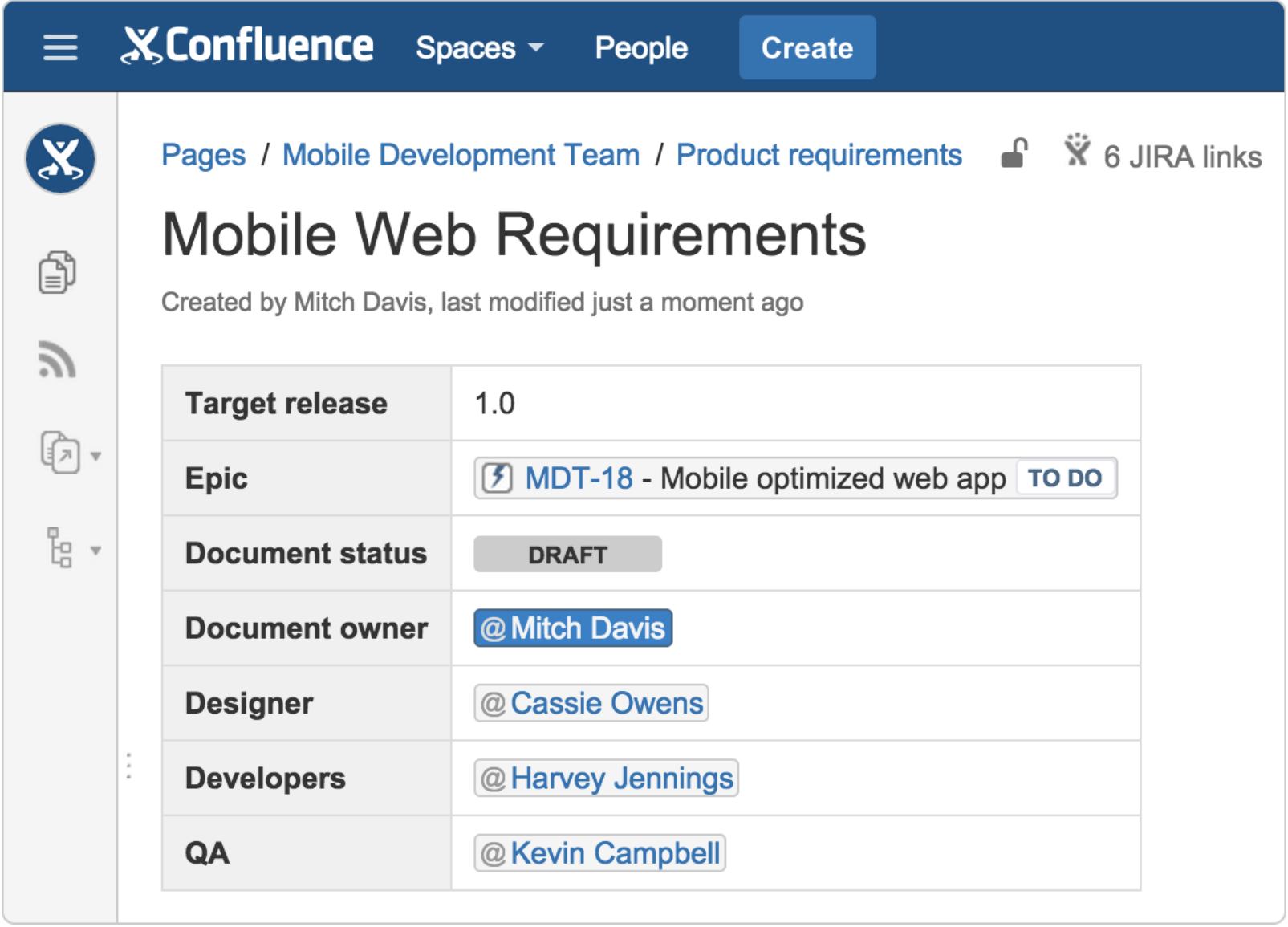
Keeping requirements lean with a one-page dashboard

When writing a requirements document, it's helpful to use a consistent template across the team so everyone can follow along and give feedback. At Atlassian, we use Confluence to create product requirements with the product requirements document template. We've found that the section below provides "just enough" context to understand a project's requirements and its impact on users:

1. Define project specifics

We recommend including high-level direction at the top of the page as follows:

- Participants: Who is involved? Include the product owner, team, stakeholders
- Status: What's the current state of the program? On target, at risk, delayed, deferred, etc.
- Target release: When is it projected to ship?



2. Team goals and business objectives

Get straight to the point. Inform, but don't bore.

3. Background and strategic fit

Why are we doing this? How does this fit into the overall company objectives?

Goals

- Our goal is to create a mobile version of the website. Sometimes users click on a link in an email notification using their mobile phone and need to be able to access our application from mobile Chrome or Safari.
- We want to meet feature parity with most functions - except we can skip creating events.

Background and strategic fit

We all know mobile is on the rise. A recent survey to customers showed that 85% of users use their mobile on a daily basis. Most of our customers also use competitor apps, so this is something we need to have. We will be able to measure our success through analytics and can use the website today as a baseline.

Customer research

- Customer interview - Netflix
- Customer interview - Homeaway
- Customer interview - Bitbucket

4. Assumptions

List the technical, business, or user assumptions you might be making.

5. User Stories

List or link to the user stories involved. Also link to customer interviews, and inclused screenshots of what you've seen. Provide enough detail to make a complete story, and include success metrics.

Requirements

#	User story title	User story description	Priority	Notes
1	Facebook Integration JIRA | MDT-13	A user wants to sign up via Facebook	MUST HAVE	• We will need to talk to Cassie Owens about this one. • There has also been some research done on this (see Facebook integration prototype)
2	Activity Stream JIRA | MDT-14	A user wants to view the latest updates via the mobile dashboard so that they can get a better understanding of what is in place	MUST HAVE	
3	Post Updates JIRA | MDT-15	A user wants to be able to post status updates on the go	MUST HAVE	The key things we will need to support: • Text status updates • Mentions • Support for images • Smart embedding for things like YouTube videos etc.
4	API JIRA | MDT-16	A developer wants to integrate with the mobile app so that they can embed the activity stream on their website	SHOULD HA...	• We should chat to Team Dyno as they did something similar.

6. User interaction and design

After the team fleshes out the solution for each user story, link design explorations and wireframes to the page.

User interaction and design

Description	Login screen	Activity stream
Mockup		

7. Questions

As the team understands the problems to solve, they often have questions. Create a table of

"things we need to decide or research" to track these items.

8. What we're not doing

Keep the team focused on the work at hand by clearly calling out what you're not doing. Flag things that are out of scope at the moment, but might be considered at a later time.

> PRO TIP:
>
> The agile manifesto reminds us that we can be flexible with how we create requirements. Some teams do user story mapping exercises to identify problems and solutions. Sometimes the full product triad (product owner, developer, and designer) visits a customer and then brainstorms solutions to a particular problem that the customer mentioned.
>
> No matter how requirements are born, it's important that the team considers them to be one of many ways they can define and communicate customer problems. See our section on agile design to learn how product owners can use Keynote and Powerpoint to mock up real experiences as requirements.

Example of a one-page PRD

Here's a look at a fully fleshed out product requirements document that we created using Confluence. Remember, no two product requirements will be exactly the same. Use this example to understand the different elements that should be included in your PRD, but not as the definitive way to do it.

Mobile Web Requirements

Created by Mitch Davis, last modified just a moment ago

Target release	1.0
Epic	MDT-18 - Mobile optimized web app TO DO
Document status	DRAFT
Document owner	@Mitch Davis
Designer	@Cassie Owens
Developers	@Harvey Jennings
QA	@Kevin Campbell

Background and strategic fit

We all know mobile is on the rise. A recent survey to customers showed that 85% of users use their mobile on a daily basis. Most of our customers also use competitor apps, so this is something we need to have.

Customer research

- Customer interview - Netflix
- Customer interview - Homeaway
- Customer interview - Bitbucket

Requirements

#	User story title	User story description	Priority	Notes
1	Facebook Integration MDT-13 TO DO	A user wants to sign up via Facebook	Must Have	• We will need to talk to Cassie Owens. • There has also been some research done on this (see Facebook integration prototype)
2	Activity Stream MDT-14 TO DO	A user wants to view the latest updates via the mobile dashboard so that they can get a better understanding of what is in place	Must Have	
3	Post Updates MDT-15 TO DO	A user wants to be able to post status updates on the go	Must Have	The key things we will need to support: • Text status updates • Mentions • Support for images • Smart embedding for YouTube vids
4	API MDT-16 TO DO	A developer wants to integrate with the mobile app so that they can embed the activity stream on their website	Should Have	• We should chat to Team Dyno as they did something similar.

User interaction and design

Description	Login screen	Activity stream
Mockup		

Questions

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome
What about Google Apps	• We think this is important, but not for version one. • We can look at this at a later stage. • It might be worth someone looking into a shared notification library to do this.
Are we supporting Blackberry?	• Again, not for initial version - but we haven't had much demand for this.
Should we have an offline mode?	• We've talked about the pros and cons. In brief: Seamless experience for customers, they won't notice if there is a connection issue Most of our competitors don't have this Could be expensive to build Should we spike this at a later sprint?

Not Doing

- Google Apps Authentication - out of scope, see above for details
- Blackberry support - we won't look at doing this, if demand picks up we can look at it.
- Native app. We are starting with a mobile web view first and get back to a native app depending on feedback that we get.

requirements

> **Want to give it a try?** Sign up or log into Confluence >>
> Once you're in, choose the product requirements blueprint and then follow the tutorial below to help you get started setting up your requirements:
> - How to create a product requirements document

Key takeaways from the one-page approach:

If you are to take away anything from this blog, understand the "why" – the not "what" – because the "why" will help you explore what is best for your team. Here are the benefits and challenges we've observed with the one-page dashboard approach:

1. One page, one source

Keeping it simple. The product requirements document becomes the "landing page" for everything related to the set of problems within a particular epic. Having something that is the central go-to location saves your team members time in accessing this information and gives them a concise view.

2. Extra agility

One of the awesome things about using a simple page to collaborate (verses a dedicated requirements management tool) is that you can be agile about your documentation! You don't have to follow the same format every time – do what you need, when you need it, and be agile about it. Chop and change as needed.

3. Just enough context and detail

We often forget how powerful a simple link can be. We embed a lot of links within our product requirements documents. It helps abstract out the complexity and progressively disclose the information to the reader as needed. Linking detailed resources may include such things as:

- Customer interviews for background, validation or further context for the feature
- Pages or blogs where similar ideas were proposed
- Previous discussion or technical documentation and diagrams
- Videos of product demos or other related content from external sources

4. Living stories

I see a lot of customers do this as well. Once the stories have been roughly thought out and entered as issues in Jira Software, we link to them in our page (which, conveniently, creates a link from the issues back to the page as well). The two-way syncing between Confluence and Jira Software means we automatically get to see each issue's current status right from the requirements page.

5. Collective wisdom

Capturing product requirements in Confluence makes it easy for other people in different teams to contribute and make suggestions. I've been amazed at the number of times someone from another team jumped into the conversation with a comment providing great feedback, suggestions, or lessons learned from similar projects. It helps a large organization feel like a small team.

6. Engaging "extras"

Diagrams made with tools like Visio, Gliffy, or Balsamiq better communicate the problems to your team. You can also embed external images, videos, and dynamic content.

7. Collaboration!

The most important aspect of all this is getting everyone involved. Never write a product requirements document by yourself – you should always have a developer with you and write it together. Share the page with your team and get feedback. Comment, ask questions, encourage others to contribute with thoughts and ideas. This is especially important for distributed teams who don't often get a chance to discuss projects in person.

Challenges

With every approach there are down-sides. Here there are two main challenges we've experienced and observed from customers as well:

1. Documentation can go stale

What happens when you implement a story and get feedback and then modify the solution? Does someone go back and update the requirements page with the final implementation? This is a challenge with any type of documentation, and it's always worth questioning whether such trade-offs are worthwhile. Talk to your team about what you would do in a scenario like this.

2. Lack of participation

"What can I do to encourage people to comment?", "How can I encourage people to write more specs and stories on our intranet?". This is a tough nut to crack, and it comes down to various wiki adoption techniques in your organization. There are plenty of resources to help you here. There may be deeper cultural issues at play here, too.

Now get to work!

When requirements are nimble, the product owner has more time to understand and keep pace with the market. And keeping them informative-but-brief empowers the development team to use whatever implementation fits their architecture and technology stack best.

Once a project's requirements are reasonably well-baked, we recommend linking the user stories in section 5 above to their corresponding stories in the development team's issue tracker. This makes the development process more transparent: it's easy to see the status of each piece of work, which makes for more informed decisions from the product owner, as well as downstream teams like marketing and support.

> PROTIP:
>
> **Don't track the user stories that come from project requirements in one system and defects in another. Managing work across two systems is needlessly challenging and just wastes time.**

Remember, be agile in your evolution of requirements for a project. It's okay to change user stories as the team builds, ships, and gets feedback. Always maintain a high quality bar and a healthy engineering culture – even if it means shipping fewer features.