Product Requirement Document (PRD) Template

Welcome to the Product Requirement Document (PRD) Template from Mastering Product. This comprehensive template will help you document product requirements clearly and effectively, ensuring alignment between product, design, and engineering teams.

> **About This Template**
>
> This template is designed for product managers who need to create clear, structured product requirements. It includes all essential sections needed for effective product documentation, with instructions and examples to guide you through the process.

How to Use This Template

1. Review the entire template to understand all sections
2. Customize sections based on your product's specific needs
3. Fill in each section with your product details
4. Share with stakeholders for feedback and alignment
5. Update as needed throughout the product development process

> **Pro Tip:** A good PRD should be clear, concise, and focused on the "what" rather than the "how." It should provide enough detail for designers and engineers to understand requirements without prescribing specific implementation details.

Template Sections

1. Document Information

This section provides basic information about the document, including authorship, version history, and approvals.

Title	*[Product/Feature Name] Product Requirements Document*
Author	*[Your Name], Product Manager*
Last Updated	*[Date]*
Version	*[e.g., 1.0]*
Status	*[Draft/In Review/Approved]*

Stakeholders

Role	Name	Approval Status
Product Owner	*[Name]*	*[Pending/Approved]*
Engineering Lead	*[Name]*	*[Pending/Approved]*
Design Lead	*[Name]*	*[Pending/Approved]*
QA Lead	*[Name]*	*[Pending/Approved]*

Document History

Version	Date	Author	Description of Changes

0.1	*[Date]*	*[Name]*	Initial draft
0.2	*[Date]*	*[Name]*	*[Description of changes]*
1.0	*[Date]*	*[Name]*	Approved version

2. Product Overview

This section provides context about the product or feature, including the problem it solves, target users, and business objectives.

2.1 Problem Statement

Clearly articulate the problem this product or feature solves. What pain points does it address? Why is this important to solve now?

Example: Our users struggle to track their project tasks across multiple teams, resulting in missed deadlines and duplicated work. They need a centralized task management system that provides visibility across teams and integrates with their existing tools.

2.2 Target Users and Customers

Describe the primary users and customers for this product or feature. Include relevant demographic information, job roles, and key characteristics.

Example: This feature targets project managers (primary users) and department heads (key stakeholders) in mid-sized companies (100-1000 employees) who manage cross-functional projects involving 3+ teams.

2.3 Business Objectives and Success Metrics

Outline the business goals this product or feature supports and how success will be measured. Include specific, measurable metrics.

Example:
Business Objectives:

- Increase user engagement with our project management platform
- Reduce customer churn by addressing a top user complaint
- Differentiate from competitors with superior cross-team functionality

Success Metrics:

- 30% increase in daily active users within 3 months of launch
- 25% reduction in support tickets related to cross-team visibility
- 15% improvement in customer retention rate

2.4 Project Scope and Constraints

Define what is in and out of scope for this project. Include any constraints (time, budget, technical, etc.) that impact the project.

Example:
In Scope:

- Cross-team task visibility dashboard
- Task assignment and status tracking
- Basic reporting functionality

Out of Scope:

- Advanced analytics and custom reports
- Mobile application (will be addressed in future release)
- Third-party integrations beyond our current API partners

Constraints:

- Must launch within Q3 to align with annual customer conference
- Engineering team has limited bandwidth due to parallel projects

- Must maintain compatibility with existing database architecture

3. User Stories and Requirements

This section details the user stories, personas, and specific requirements that drive the product development.

3.1 User Personas

Include brief descriptions of the primary user personas for this product or feature. Focus on their goals, pain points, and relevant behaviors.

Example:
Maria, Project Manager (Primary User)
Demographics: 35-45 years old, 10+ years of experience
Goals: Deliver projects on time, maintain visibility across teams, efficiently allocate resources
Pain Points: Struggles with siloed information, spends too much time in status meetings, lacks real-time updates
Behaviors: Checks project status multiple times daily, coordinates across 4-6 teams, reports to executive stakeholders weekly

3.2 User Stories

List the key user stories that this product or feature addresses. Use the format: "As a [user role], I want to [action/goal], so that [benefit]."

Example:

1. As a project manager, I want to view tasks across all teams in a single dashboard, so that I can quickly identify bottlenecks and dependencies.
2. As a team lead, I want to assign tasks to members of other teams, so that I can manage cross-functional work without switching contexts.
3. As a department head, I want to see progress metrics for all projects in my department, so that I can report accurate status to executives.

3.3 User Flows and Journeys

Describe the key user flows and journeys that this product or feature enables. Include diagrams if helpful.

Example:
Cross-Team Task Assignment Flow:

1. User navigates to project dashboard
2. User clicks "Create Task" button
3. User fills in task details (name, description, due date)
4. User selects team from dropdown menu
5. User selects team member from filtered list
6. User clicks "Assign" button
7. System sends notification to assigned team member
8. Task appears in both project view and assignee's personal task list

3.4 Acceptance Criteria

Define the specific conditions that must be met for each key user story or feature to be considered complete.

Example:
For User Story #1: "View tasks across all teams in a single dashboard"

- Dashboard displays tasks from all teams the user has access to
- Tasks can be filtered by team, status, priority, and due date
- Dashboard updates in real-time when task status changes
- User can click on any task to view full details
- Dashboard loads in under 2 seconds with up to 500 tasks
- Dashboard is accessible on desktop and tablet devices

4. Functional Requirements

This section details the specific functionality that the product or feature must provide.

4.1 Feature Descriptions

Provide detailed descriptions of each feature or component of the product. Include purpose, functionality, and any specific behaviors.

Example:
Feature: Cross-Team Dashboard
Purpose: Provide a unified view of tasks across multiple teams
Description: The dashboard displays tasks from all teams in a customizable grid view with filtering and sorting options. Users can see task status, assignee, due date, and priority at a glance, with the ability to drill down for more details.

4.2 User Interactions

Describe how users will interact with each feature, including inputs, actions, and responses.

Example:
Task Filtering Interaction:

- User clicks on filter icon in dashboard header
- System displays filter panel with options for team, status, priority, and date range
- User selects desired filter criteria
- System updates dashboard in real-time to show only matching tasks
- Selected filters appear as removable tags at top of dashboard
- User can click on a filter tag to remove that filter
- User can save filter configurations for future use

4.3 System Behaviors

Outline how the system should behave in various scenarios, including edge cases and error conditions.

Example:
Task Assignment Behaviors:

- When a task is assigned, the system sends a notification to the assignee

- If the assignee has notification preferences set, the system respects those settings
- If a task is reassigned, both the previous and new assignees receive notifications
- If a user attempts to assign a task to someone without proper permissions, the system displays an error message
- If a team is at capacity (based on workload settings), the system displays a warning before allowing assignment

4.4 Dependencies

Identify any dependencies between features or on external systems, APIs, or services.

Example:
Feature Dependencies:

- The cross-team dashboard requires the user permission system to be updated to support cross-team visibility
- Real-time updates depend on the WebSocket service being operational
- Team capacity warnings require the workload calculation service to be implemented first
- Email notifications depend on the notification service and email delivery system

5. Non-Functional Requirements

This section covers requirements related to system quality, performance, security, and other non-functional aspects.

5.1 Performance Requirements

Specify performance expectations such as response times, throughput, and resource utilization.

Example:

- Dashboard must load in under 2 seconds for up to 500 tasks
- Filter operations must complete in under 500ms
- System must support up to 1,000 concurrent users without degradation

- Task updates must propagate to all users within 3 seconds
- API endpoints must respond within 200ms for 95% of requests

5.2 Security Requirements

Define security requirements including authentication, authorization, data protection, and compliance needs.

Example:

- All data must be encrypted in transit using TLS 1.2 or higher
- User permissions must be enforced at both UI and API levels
- Authentication must support single sign-on (SSO) integration
- System must maintain audit logs for all task assignments and status changes
- Sensitive data must be encrypted at rest

5.3 Compliance Requirements

Specify any regulatory or compliance requirements that the product must meet.

Example:

- System must comply with GDPR requirements for user data
- Audit logs must be retained for a minimum of 1 year
- System must support data export for user data requests
- All third-party components must be reviewed for security compliance

5.4 Accessibility Requirements

Define accessibility standards and requirements that the product must meet.

Example:

- UI must comply with WCAG 2.1 AA standards
- All interactive elements must be keyboard accessible

- Color contrast must meet minimum ratios for readability
- All images must include appropriate alt text
- System must be compatible with screen readers

6. Design and Technical Considerations

This section covers design guidelines, technical constraints, and integration requirements.

6.1 UI/UX Guidelines

Provide guidance on user interface and experience design, including any specific design requirements or constraints.

Example:

- Design must follow the company design system and component library
- Dashboard should prioritize information density while maintaining clarity
- Color coding should be used consistently to indicate task status and priority
- Interactive elements should provide appropriate feedback on hover and click
- Mobile views should adapt to smaller screens by prioritizing essential information

6.2 Technical Constraints

Identify any technical limitations or constraints that impact implementation.

Example:

- Must use existing React component library for UI elements
- Must maintain compatibility with IE11 for enterprise customers
- Backend changes limited to approved API endpoints due to legacy system constraints
- Must work within current database schema with minimal changes
- Must support offline functionality for basic task viewing

6.3 Integration Points

Specify how the product will integrate with other systems, services, or APIs.

Example:

- Must integrate with company SSO service for authentication
- Must connect to notification service for email and in-app alerts
- Must support webhook integration for third-party tools
- Must use existing analytics service for user behavior tracking
- Must integrate with calendar services (Google, Outlook) for due date synchronization

6.4 Migration Considerations

If applicable, describe any data migration or user transition considerations.

Example:

- Existing tasks must be migrated to new schema without data loss
- User permissions need to be updated to support cross-team visibility
- Feature should be rolled out gradually using feature flags
- Legacy dashboard views should remain accessible during transition period
- Data migration scripts must include validation and rollback capabilities

7. Implementation Plan

This section outlines the approach to implementing and releasing the product or feature.

7.1 Phasing and Milestones

Break down the implementation into phases or milestones with clear deliverables for each.

Example:
Phase 1: Foundation (Weeks 1-3)

- Update permission model to support cross-team visibility
- Create API endpoints for cross-team data access
- Develop basic dashboard UI with filtering capabilities

Phase 2: Core Functionality (Weeks 4-6)

- Implement task assignment across teams
- Add real-time updates via WebSockets
- Develop notification system for task changes

Phase 3: Enhancement and Polish (Weeks 7-8)

- Add saved filters and customizable views
- Implement performance optimizations
- Complete accessibility improvements

7.2 Release Criteria

Define the criteria that must be met before the product or feature can be released.

Example:

- All acceptance criteria for priority 1 user stories are met
- Performance requirements validated in staging environment
- Security review completed with no critical or high issues
- Accessibility audit passed with no major issues
- QA testing completed with no critical bugs
- Beta testing with select customers completed successfully

7.3 Rollout Strategy

Describe how the product or feature will be rolled out to users, including any phased approach.

Example:

- Week 1: Alpha release to internal teams for testing
- Week 2: Beta release to 10% of customers who opted into early access
- Week 3: Expand to 25% of all users if no major issues
- Week 4: Full release to all users
- Feature will be controlled via feature flags for quick rollback if needed
- In-app announcements and email notifications will introduce the feature to users

7.4 Monitoring Plan

Outline how the product or feature will be monitored after release to ensure it meets expectations.

Example:

- Key metrics to monitor: dashboard load time, task update latency, error rates, user adoption
- Set up alerts for performance degradation or error rate increases
- Monitor support tickets related to the new features
- Conduct user surveys 2 weeks post-launch to gather feedback
- Schedule review meeting 1 month after full release to assess performance against success metrics

8. Appendix

This section includes additional information that supports the main document but isn't essential to understanding the core requirements.

8.1 Supporting Research

Summarize or reference relevant research that informed the product requirements.

Example:

- User interviews conducted with 15 project managers (see Research Report #123)
- Analysis of support tickets revealed cross-team visibility as top pain point

- Competitive analysis of 5 leading project management tools (see Competitive Analysis Doc)
- Usage data showing 78% of projects involve multiple teams

8.2 Competitive Analysis

Provide information about how competitors address similar problems or features.

Example:
Competitor A: Offers basic cross-team visibility but requires manual setup for each project
Competitor B: Strong in cross-team collaboration but lacks real-time updates
Competitor C: Recently launched similar feature but with limited filtering capabilities
Our Differentiation: Automatic cross-team visibility with real-time updates and advanced filtering

8.3 Technical Specifications

Include or reference more detailed technical specifications if needed.

Example:

- API Specification Document: [link to detailed API docs]
- Database Schema Changes: [link to schema documentation]
- Architecture Diagram: [link to system architecture]

8.4 Glossary of Terms

Define any specialized terms, acronyms, or concepts used in the document.

Example:
Cross-team visibility: The ability to view and manage tasks across multiple teams or departments
Task assignment: The process of allocating a task to a specific user for completion
Real-time updates: Changes that are immediately reflected for all users without requiring page refresh
Workload capacity: The maximum number of tasks or hours assigned to a team based on availability

Best Practices for Using This Template

- **Be Clear and Concise:** Use simple, direct language that all stakeholders can understand.
- **Focus on the What, Not the How:** Describe what needs to be accomplished, not how it should be implemented technically.
- **Include Visuals:** Add wireframes, mockups, or diagrams where they help clarify requirements.
- **Prioritize Requirements:** Indicate which features or requirements are must-haves versus nice-to-haves.
- **Get Stakeholder Input:** Share drafts with key stakeholders early to ensure alignment.
- **Keep It Updated:** Treat the PRD as a living document that evolves as you learn more.
- **Link to Supporting Documents:** Reference research, designs, and technical specs rather than duplicating them.

> **Pro Tip:** The best PRDs are collaborative documents. Involve engineering and design early in the process to ensure requirements are feasible and well-understood.
