PRODUCT DISCOVERY WORKBOOK

# WordPress Plugin Scope & Requirements Planning Checklist

A decision-ready workbook for founders, product teams, agencies, and developers planning a maintainable WordPress plugin.

**Built to be used**

Print it, annotate it in discovery, attach it to a brief, or use it as a release gate. Every checkbox should produce a decision, owner, or piece of evidence.

## What is inside

| Module | Purpose |
|---|---|
| 01 | Problem and outcome definition |
| 02 | Users, roles, and workflows |
| 03 | Data, integrations, and architecture |
| 04 | Quality, security, and operations |
| 05 | Delivery, acceptance, and release readiness |
| 06 | Decision log and handover worksheet |

Edition: 2026.1 | Publisher: WPStack | wpstack.online

# 1. Start with the problem, not the feature list

A useful scope explains who is blocked, what happens today, what must improve, and how success will be measured. A feature list without these decisions pushes ambiguity into development, where it becomes expensive.

## Problem statement

- [ ] Name the primary user and the recurring problem in one sentence.
- [ ] Describe the current workaround, including tools, manual steps, and failure points.
- [ ] Define the business outcome: time saved, risk reduced, revenue enabled, or errors prevented.
- [ ] Write three measurable success indicators and how they will be observed.
- [ ] List what the plugin will explicitly not solve in version 1.
- [ ] Identify the decision-maker, product owner, technical owner, and support owner.

> **Scope test** A new team member should be able to explain why the plugin exists without seeing a mock-up.

## Discovery record

| Decision | Your answer / evidence | Owner |
|---|---|---|
| Primary user | | |
| Cost of the current problem | | |
| V1 success metric | | |
| Explicit non-goals | | |

# 2. Map users, permissions, and complete workflows

WordPress roles are a starting point, not a requirements model. Document capabilities at the action level and trace each workflow from entry to success, failure, retry, and recovery.

## User and capability inventory

| Actor | Reads | Creates/changes | Approves/deletes |
|---|---|---|---|
| Administrator | | | |
| Editor / manager | | | |
| Customer / member | | | |
| External system | | | |

## Workflow checklist

- [ ] Document the trigger, prerequisites, happy path, result, and notification for every workflow.
- [ ] Define empty, loading, partial-success, validation-error, permission-denied, and system-error states.
- [ ] Decide whether each action is synchronous, queued, scheduled, or manually retried.
- [ ] Specify who can reverse an action and what audit evidence must remain.
- [ ] Identify accessibility, keyboard, localization, timezone, and multisite expectations.
- [ ] Define behavior when a dependency, remote API, or payment provider is unavailable.

# 3. Define data ownership before choosing screens

Choose storage based on access patterns, volume, lifecycle, relationships, portability, and privacy. The database structure should make expected operations safe and predictable.

## Data model worksheet

| Data object | System of record | Retention / deletion | Sensitive? |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## Architecture decisions

- [ ] Choose options, post meta, custom post types, taxonomies, users, or custom tables from documented query needs.
- [ ] Define unique keys, indexes, maximum expected rows, and high-frequency queries.
- [ ] Specify schema versioning, migrations, rollback behavior, and interrupted-upgrade recovery.
- [ ] Define uninstall behavior separately from deactivation; never delete valuable data unexpectedly.
- [ ] Namespace functions/classes and define compatibility with other plugins, themes, and object caches.
- [ ] List background jobs, cron frequency, concurrency controls, batch size, and observability.

## Integration contract

| System/API | Direction | Auth | Timeout/retry | Failure owner |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

# 4. Make quality attributes testable

Words such as fast, secure, scalable, and reliable are not requirements until they have a measurable threshold and a test method.

| Attribute | Define | Example evidence |
|---|---|---|
| Performance | Dataset, traffic, page and latency budget | Query count, p95 response, memory |
| Compatibility | WP/PHP/browser, multisite, themes, plugins | Test matrix and results |
| Security | Trust boundaries, capabilities, input/output rules | Threat review and negative tests |
| Privacy | Personal data, consent, export, erase, retention | Data inventory and policy mapping |
| Reliability | Retries, idempotency, rollback, recovery | Failure-injection results |
| Accessibility | Target standard and key journeys | Keyboard and assistive-tech checks |

## Release-quality checklist

- [ ] Sanitize and validate inputs, check capabilities, use nonces for CSRF where applicable, and escape output by context.
- [ ] Avoid blocking remote calls and unbounded queries in normal admin or front-end requests.
- [ ] Test fresh install, upgrade from supported versions, deactivation, reactivation, and uninstall.
- [ ] Test realistic data volume, slow dependencies, network failures, duplicate requests, and interrupted jobs.
- [ ] Document logging without exposing secrets or personal data.
- [ ] Prepare support diagnostics that users can share safely.

# 5. Turn scope into delivery and acceptance

Split delivery into demonstrable increments. Every milestone should end with working behavior, review evidence, and a clear decision about what changes next.

## Milestone plan

| Milestone | Demonstrable outcome | Acceptance evidence | Owner |
|---|---|---|---|
| Discovery | | | |
| Architecture | | | |
| Working slice | | | |
| Release candidate | | | |

## Commercial and operational decisions

- [ ] Confirm IP ownership, source access, third-party licensing, credentials, and account ownership.
- [ ] Define what constitutes a change request and how it affects schedule and price.
- [ ] Agree maintenance coverage, response targets, support channel, and excluded work.
- [ ] Assign responsibility for hosting, monitoring, backups, incident response, and release approval.
- [ ] Record dependencies that can change cost: data migration, legacy compatibility, external APIs, and compliance review.

**Acceptance rule**

Write acceptance criteria as observable behavior: Given a state, when an actor performs an action, then a specific result and audit effect occur.

# 6. Release readiness and handover

A plugin is not finished when the code works on one site. It is ready when another responsible person can install, operate, diagnose, update, and recover it.

## Final release gate

- [ ] Version, changelog, supported WordPress/PHP versions, license, and plugin headers are correct.
- [ ] Readme covers purpose, installation, configuration, support boundaries, privacy, and external services.
- [ ] Build artifact is reproducible and excludes development secrets, tests, caches, and private files.
- [ ] Automated checks and manual journeys pass on the agreed compatibility matrix.
- [ ] Backup and rollback steps are documented and have been rehearsed.
- [ ] Monitoring, logs, support escalation, and maintenance ownership are active.
- [ ] Known limitations, deferred risks, and roadmap decisions are signed off.

## Decision log

| Date | Decision / assumption | Reason and evidence | Owner |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

**Next step**

Use this completed workbook as the source document for a technical discovery call. WPStack can turn it into an architecture proposal, delivery plan, and fixed planning range at wpstack.online/custom-plugin-development/

Reference basis: WordPress Plugin Developer Handbook - plugin basics, best practices, security guidance, directory guidelines, and planning/maintenance guidance. Documentation changes over time; validate current platform requirements before release.