Build Mart Nepal

V45 Upgrade & Deployment Guidebook

Pagination • Product Quick View • Version Continuity • Vercel Deployment

A practical owner, developer and release-manager handbook

Build Mart Nepal
buildmartnepal.com
Building • Living • Better

Version 45.0.0 | July 2026

Production note: complete staging tests and environment configuration before public launch.

GUIDEBOOK MAP

Contents

How to use this guide

Owners can use sections 1, 6, 8, 11 and 12. Developers should follow every section in order. Release managers should treat sections 8–11 and the final checklist as approval gates.

SECTION 1

Executive overview

Build Mart Nepal V45 is an additive reliability and usability upgrade. It keeps the full V44 application and introduces stronger pagination, a clearer product quick view, package compatibility checks, Vercel deployment controls and a version-continuity system.

Business outcomes

- Visitors can navigate large blog and product catalogues without loading every record at once.
- Filters, sorting, search terms, page size and page number remain shareable through URL parameters.
- Quick view gives the buyer essential information without overcrowding the screen.
- Deployment checks detect incompatible package versions and unsupported Lucide icon imports.
- A generated union manifest prevents older application files from disappearing during a future version upgrade.
- Public guide pages explain the marketplace, onboarding, project, finance, mobile and deployment workflows.

Preservation rule

V45 must not be treated as a clean rebuild. Continue from the latest complete codebase, keep database identities and routes, and approve any intentional removal separately.

Release boundaries

The codebase is prepared for staging and Vercel deployment. Production credentials, live database migration, payment verification, email delivery, real-device mobile testing and a full production build must still be completed in the target environment before launch.

SECTION 2

What V45 changes

Area	Before	V45
Blog listing	Large lists with no complete page-window control	12 posts per page, result range, filters and canonical pagination
Product listing	Fixed page size and basic navigation	12/24/48 page size, nearby pages, first/last and URL preservation
Quick view	Dense modal content could overflow	Viewport-bounded, internally scrollable, compact information hierarchy
Package safety	Manual review	Automated Next, React, React DOM, Prisma and Lucide compatibility check
Lucide icons	Known invalid import fixed manually	Source scan plus installed-module export validation after npm ci
Version retention	Relied on memory and release notes	1,266-file V42–V44 continuity union manifest
Guidance	Version notes spread across files	Public guide centre plus this owner/deployment guidebook

New public routes

- /platform-guides and fifteen detailed guide pages
- /deployment-readiness
- /version-history
- /platform-status

These routes are included in the generated sitemap and linked from the footer. They do not expose private diagnostics or secrets.

SECTION 3

Blog pagination

The blog listing now requests one page from the data source instead of loading the whole library. The default page size is 12.

1	**Read URL parameters** The server reads q, category and page from the request.
2	**Build database conditions** Published state, category and text-search conditions are applied before counting.
3	**Count and fetch** The server counts matching records and fetches only the current page using skip and take.
4	**Correct invalid pages** An out-of-range page redirects to the final valid page while preserving filters.
5	**Render navigation** Previous, next, nearby page numbers and ellipses are generated by the shared pagination utility.

SEO behaviour

- Page 1 uses the canonical /blog URL.
- Later unfiltered pages use a page-specific canonical query.
- Filtered search pages are noindex, follow to avoid thin or duplicate result pages.
- Article pages remain independent and are not merged into the listing.

API compatibility

Existing callers still receive a plain post array when no pagination parameter is supplied. New callers can request paginated output with page, limit, q, category or paginated=true.

SECTION 4

Product pagination, sorting and filters

The product listing keeps database-backed filtering and sorting and now supports page sizes of 12, 24 or 48. The selected state remains in the URL so users can share, bookmark and navigate with browser back and forward controls.

URL parameter	Purpose	Example
q	Product search term	?q=cement
categories	Comma-separated categories	?categories=Cement,Steel
brand	Selected brand	?brand=BMN
minPrice / maxPrice	Price range	?minPrice=500&maxPrice=5000
inStock	Availability filter	?inStock=true
sort	Database sorting mode	?sort=price-asc
limit	Items per page	?limit=24
page	Current page	?page=3

User experience rules

- Changing a filter, search term, sort option or page size resets to page 1.
- Changing only the page preserves all other parameters.
- An invalid or excessive page redirects safely.
- The result range states exactly which items are visible.
- Mobile users receive the same state through responsive sorting and filter controls.

Performance note

Pagination helps, but production speed also depends on database indexes, query selection, image optimisation and cache strategy. Validate real data volumes in staging.

SECTION 5

Decluttered product quick view

Quick view is designed for a fast buying decision, not to replace the full product page. It now follows a strict information hierarchy.

1	**Identity** Product name, brand and category.
2	**Availability** Stock, installation and verified-supplier badges.
3	**Commercial summary** Price or request-price status.
4	**Decision details** Supplier, delivery, unit/stock and rating.
5	**Short overview** A limited description and up to three key features.
6	**Primary actions** Add to cart, full details and request quotation remain visible in the sticky footer.

Responsive behaviour

- The dialog uses the dynamic viewport height and cannot exceed the screen.
- The content area scrolls internally while the header and action footer stay usable.
- The image uses a safe local fallback when the source fails.
- Mobile layouts stack the image and details.
- Compare and save actions remain available without crowding the title row.
- Focus trapping, Escape close and focus restoration are provided by the dialog component.

Do not overload quick view

Technical documents, full specifications, long reviews, supplier policies and detailed media belong on the complete product page.

SECTION 6

How to merge versions without losing changes

Version loss usually happens when a developer copies only selected new files onto an older base. V45 replaces that informal process with a repeatable continuity workflow.

1	**Choose the newest complete base** Start from the highest version that contains all previous runtime features, not from a partial export.
2	**Keep immutable backups** Preserve the source ZIP, database backup and media backup before editing.
3	**Compare recent release folders** Create the union of application routes, components, libraries, Prisma files, public assets, scripts and configuration.
4	**Generate a manifest** Store each required path and the source versions in which it appeared.
5	**Apply the new upgrade additively** Modify existing files intentionally and add new files without deleting unrelated modules.
6	**Run continuity checks** Fail the release when any retained file is missing or empty.
7	**Run route and link checks** Confirm that preserved files are still connected and reachable.
8	**Document intentional removals** Remove a manifest entry only after review, migration planning and written approval.

V45 implementation

The file scripts/version-union-manifest-v42-v44.json contains the accessible runtime union. Run npm run verify:continuity before every release.

Included	Excluded intentionally
App and API routes	Old ZIP archives
Web and mobile components	Generated screenshots
Prisma schema and migrations	Historical validation output
Libraries, constants and data	Temporary QA render folders
Public runtime assets	Version-specific checksums
Primary configuration and scripts	node_modules and build output

SECTION 7

Package, React and Lucide compatibility

V45 keeps the package set that is already represented by the exact root lockfile. React and React DOM remain aligned at 18.3.1. Next.js remains 15.5.12. Lucide React is pinned to 0.372.0 rather than using an open caret range.

Package	V45 declaration	Verification
Next.js	15.5.12	Pinned to lockfile; peer range checked
React	18.3.1	Matches React DOM and Next peer range
React DOM	18.3.1	Matches React and Next peer range
lucide-react	0.372.0	Pinned and checked against React peer range
@prisma/client	6.19.2	Pinned to generated client package
Prisma CLI	6.19.2	Matches @prisma/client

Preventing icon build errors

Lucide icons are named exports. An icon name that does not exist in the installed package causes a static build error. V45 scans all Lucide named imports, explicitly blocks ChartNoAxesCombined and, after npm ci, compares every imported name with the installed module exports.

1. **Before installation**

   Run npm run verify:lucide to detect forbidden known imports.

2. **After installation**

   Run the same command again to verify every named export against node_modules.

3. **When adding an icon**

   Confirm the exact exported name in the installed package or use an existing verified icon.

4. **After package updates**

   Regenerate the lockfile with the selected package manager and rerun compatibility and build checks.

One package manager only

Keep package.json and package-lock.json in the repository root. Do not add Yarn, pnpm or Bun lockfiles to this project.

SECTION 8

Deploying V45 to Vercel

1	**Create a release branch** Commit the V45 source, package.json, package-lock.json and migration files together.
2	**Create a staging Vercel project** Use the repository root as the root directory and choose the Next.js framework preset.
3	**Add environment variables** Use Vercel project settings. Never put live secrets into Git.
4	**Prepare the database** Back up production, test migrations in staging and run prisma migrate deploy.
5	**Run pre-deployment verification** Complete package, Lucide, continuity, route, link, sitemap, type and build checks.
6	**Deploy to staging** Use the supplied vercel.json install and build commands.
7	**Complete smoke tests** Test public, authenticated, admin, payment, email, storage and mobile workflows.
8	**Promote to production** Promote only after approval and preserve the previous healthy deployment for rollback.

Commands

```
npm ci
npm run verify:deploy
npx prisma generate
npx prisma migrate deploy
npm run typecheck
npm run build:vercel
```

What Vercel reads

- The root package-lock determines npm installation.
- vercel.json supplies deterministic install and build commands.
- The build script generates Prisma Client before compiling Next.js.
- Environment variables come from Vercel settings, not from committed production files.

SECTION 9

Environment and database safety

Use the project example files as a checklist. Actual variable names may differ by provider, so confirm them against the code before production launch.

Area	Typical configuration	Production rule
Database	DATABASE_URL and direct/migration URL where required	Use TLS, restricted credentials and backups
Authentication	Auth URL, secret and provider credentials	Use strong unique secrets and correct canonical domain
Storage	S3/Cloudinary bucket, region and keys	Separate public and private access
Email	SMTP or transactional provider	Verify sender domain and bounce handling
Payments	eSewa, Khalti, ConnectIPS, Stripe or PayPal	Use server verification and signed callbacks
Monitoring	Error and performance provider	Do not log secrets or payment credentials
Social APIs	Official platform app credentials	Never scrape or invent accounts

Database release method

1. **Back up**

   Create and verify a restorable backup.

2. **Review migrations**

   Confirm they are additive and match the schema.

3. **Test staging**

   Run the same database engine and representative data.

4. **Deploy migrations**

   Use prisma migrate deploy, not migrate dev.

5. **Verify records**

   Check users, products, posts, profiles, orders and payments.

6. **Monitor**

   Watch errors, slow queries and failed transactions.

Seeds

Development seeds are useful for staging demonstrations but must not run automatically in production. Never replace authorised marketplace profiles with generated data.

SECTION 10

Testing and smoke-check plan

Layer	Checks	Release evidence
Static integrity	Syntax, imports, routes, links, sitemaps, redirects	Audit JSON and validation report
Packages	Lock sync, package peers, Lucide exports	Verification command output
Types and build	TypeScript and Next production build	Successful CI/Vercel build
Database	Generate, migration status and staging deploy	Migration log and record checks
Public UX	Home, services, products, blog, guides, sitemaps	Desktop/tablet/mobile screenshots
Accounts	Register, login, reset, roles and sessions	Test-account checklist
Commerce	Cart, RFQ, checkout, callback, invoice	Sandbox transaction references
Admin	Permissions, content, products, profiles and route health	Role-based test record
Integrations	Email, storage, notification and official social APIs	Provider delivery logs
Mobile	Deep links, offline drafts, upload and push	Device test matrix

Pagination smoke tests

- Open page 1, middle page and final page.
- Change page size and confirm page resets to 1.
- Apply search, category, brand, price, stock and sort filters.
- Use browser back and forward.
- Paste an excessive page number and confirm safe redirection.
- Test empty results and a single-page result set.

Quick-view smoke tests

- Open from every product card layout.
- Test local, remote and broken images.
- Test 320px, 390px, tablet and desktop viewports.
- Use keyboard Tab, Shift+Tab and Escape.
- Confirm cart, quote and full-details actions.

SECTION 11

Rollback and recovery

A code rollback and a database rollback are different operations. Vercel can restore a previous deployment quickly, but database changes require a tested plan.

1	**Pause risky writes** When necessary, disable the affected workflow through a safe feature flag or maintenance control.
2	**Assess scope** Identify whether the incident affects code, configuration, database, storage or an external provider.
3	**Rollback application** Promote the previous healthy Vercel deployment.
4	**Handle database safely** Prefer a forward-fix migration. Restore a backup only after reviewing the recovery point and transaction impact.
5	**Verify integrations** Confirm callbacks, webhooks, emails and storage access after rollback.
6	**Record incident** Document symptoms, timeline, affected records, resolution and prevention action.

Never improvise on production data

Do not run prisma db push, migrate reset or ad-hoc destructive SQL during an incident. Obtain a backup and review the recovery plan first.

Minimum recovery assets

- Previous healthy deployment
- Current and previous source ZIP/checksum
- Database backup and migration history
- Media-storage backup or versioning
- Environment-variable inventory
- Provider account access and incident contacts

SECTION 12

Owner and admin operating guide

Routine	What to review	Suggested frequency
Marketplace quality	Pending products, missing images, supplier evidence, stale stock	Daily
Leads and RFQs	New requests, distribution, response time and conversion	Daily
Professional network	Applications, licences, portfolio and verification	Daily/weekly
Content	Drafts, broken links, metadata, related content and schedule	Weekly
SEO	Indexing, sitemaps, canonical URLs and top landing pages	Weekly
Finance	Payments, invoices, failed callbacks and reconciliation	Daily
Support	Open tickets, priority, response time and satisfaction	Daily
System health	Route, database, storage, email and payment configuration	Daily after deploy
Backups	Backup success and restore test	Daily / scheduled test

Publishing quality gate

- Use a unique human-readable slug.
- Add a useful title, excerpt and cover image with alt text.
- Use correct heading order and a readable table of contents.
- Link to relevant products, services, professionals, tools or guides.
- Set canonical and social metadata.
- Preview mobile and desktop before publishing.

Product quality gate

- Confirm name, category, brand, unit, price mode and stock status.
- Upload clear images and a local fallback.
- Add specifications, application, standards, warranty and documents.
- Confirm supplier, delivery area and quotation terms.
- Review quick view and full product page separately.

SECTION 13

Common errors and solutions

Symptom	Likely cause	Action
Lucide export not found	Icon name is unavailable in installed version	Replace with a verified exported icon and run verify:lucide
npm ERESOLVE	Peer ranges or lockfile conflict	Use the pinned V45 package set; remove extra lockfiles; run npm ci
Prisma client error	Client not generated or schema mismatch	Run prisma generate and review migration status
Build needs environment value	Production variable missing	Configure it in Vercel; do not commit the secret
Sitemap 404 or conflict	HTML and metadata sitemap routes overlap	Keep /sitemap page separate from /sitemap.xml metadata output
Pagination repeats records	Unstable database ordering	Use deterministic secondary ordering and unique IDs
Quick view image missing	Invalid remote URL or provider restriction	Use SafeProductImage and permitted domains
Older feature disappeared	Upgrade started from partial source	Restore from latest full base and rerun continuity union
Payment shown but not verified	Client redirect treated as success	Verify on server using gateway callback/reference

Diagnostic sequence

1	**Read the exact first error** Later errors may only be consequences.
2	**Confirm version and lockfile** Check Node, npm, Next, React, Prisma and Lucide.
3	**Run targeted verification** Use package, Lucide, continuity, route and content scripts.
4	**Run typecheck and production build** Development mode may not expose every static error.
5	**Reproduce in staging** Use production-like environment and data.
6	**Document the fix** Add a regression check when practical.

SECTION 14

Final launch checklist

☐ Source backup and database backup are verified.

☐ V42–V44 continuity check passes.

☐ Root package.json and package-lock.json are synchronised.

☐ No extra package-manager lockfile exists.

☐ Package compatibility and installed Lucide export checks pass.

☐ Prisma Client is generated and migrations pass in staging.

☐ TypeScript and production build pass.

☐ Routes, static links, API links, redirects and sitemaps pass.

☐ Blog and product pagination work with filters, page size and browser history.

☐ Quick view works on mobile, tablet, desktop and keyboard.

☐ Professional registration, contact and RFQ persist real records.

☐ Authentication and every server-side role permission are tested.

☐ Images, documents and private downloads use correct access controls.

☐ Payment callbacks are verified in sandbox and production configuration is reviewed.

☐ Email delivery, notification and monitoring are tested.

☐ HTML and XML sitemaps are reachable on the canonical domain.

☐ Robots, metadata, canonical URLs and social previews are reviewed.

☐ Mobile deep links, offline drafts and push configuration are tested on devices.

☐ Previous healthy deployment and rollback owner are confirmed.

☐ Release notes and known limitations are approved.

Release decision

Do not label the system production-ready until every applicable item is completed and the result is recorded honestly.

Key project files

- START_HERE_V45.md
- V45_RELEASE_NOTES.md
- V45_DEPLOYMENT_GUIDE.md
- V45_VERSION_CONTINUITY_REPORT.md
- V45_PRODUCTION_CHECKLIST.md
- V45_VALIDATION_REPORT.md
- scripts/version-union-manifest-v42-v44.json