

FROM LOVABLE TO WORDPRESS

Test, Launch, Deliver

VOLUME 3

Take a custom WordPress theme from a local build to a tested, live, client-ready website.

Yukesh Chaudhary

Practical AI-assisted WordPress development

yukesh.com.np

Working series title. The title may be changed before publication.

ABOUT THIS VOLUME

Ship the website, not just the files

A theme is not finished when it activates. It is finished when the important pages work, the client can manage the right content, the live site is secure, and you can explain what was delivered.

What this book does

This is the final volume of a practical workflow. We continue the fictional ABC Construction project from the first two volumes. The design already exists. The WordPress theme, project archive, custom fields, menus, and settings already exist. Now we prove that they work.

I use AI as a careful assistant. I never treat its output as finished code. The human developer still decides what should change, checks the result, and accepts responsibility for the release.

About the author

Yukesh Chaudhary writes about practical web building, WordPress, SEO, and AI-assisted workflows. His approach connects client needs to working systems in plain language.

Website

<https://www.yukesh.com.np>

First edition, October 2026. Tool interfaces, prices, and free-tier limits can change. Check current provider terms before use.

Series promise: The value is the whole pipeline: requirements, UI, code, WordPress architecture, testing, launch, and delivery. No single tool is the method.

How to use it

Keep your project open while you read. Complete each checklist with the real site. When a step fails, stop, record the exact result, fix it, and repeat the test. Do not carry a known error into the next stage.

CONTENTS

Table of contents

1	Testing in LocalWP 4	12	Six-day client workflow 15
2	ABC Construction test setup 5	13	The method and handoff 16
3	Debugging AI-generated themes 6	14	Prompt engineering formula 17
4	Better bug reports 7	15	Prompt clinic 18
5	Functionality checklist 8	16	Prompt library: theme and admin 19
6	Responsive, brand, accessibility 9	17	Prompt library: content and data 20
7	SEO release gate 10	18	Prompt library: quality and fixes 21
8	Performance and security 11	19	Do not let AI do this 22
9	Prepare the migration 12	20	30-step workflow, part 1 23
10	Move Local to hosting 13	21	30-step workflow, part 2 24
11	DNS, SSL, final checks 14	22	Your first real delivery 25

A good launch is quiet. The site works, the client understands it, and nothing important is left to chance.

TESTING IN LOCALWP

Build confidence before the site is public

1

Local development gives you a private WordPress site on your own computer. You can activate a theme, break it, inspect it, and repair it without affecting visitors.

- 1 **Install Local.** Download Local from its official website and complete the installation for your operating system.
- 2 **Create a new site.** Choose a clear project name such as abc-construction. Let Local create WordPress and save the local admin login safely.
- 3 **Open the site folder.** In Local, use the option that reveals the site files. Open `app/public/wp-content/themes/`.
- 4 **Copy the theme folder.** Place the complete theme folder inside `themes`. The folder should contain `style.css` and the required templates.
- 5 **Activate it.** Open WordPress Admin, go to Appearance and then Themes, and activate the custom theme.
- 6 **Observe before editing.** Load the homepage, an ordinary page, a post, an archive, search, and the 404 page. Write down what fails.

Safe testing ground: LocalWP is not the client's public hosting. It is your workshop. Production hosting is a separate service and usually a client cost.

The first rule of testing

Change one thing at a time. If five files change and the site breaks, you do not know which change caused it. Small changes make the next decision easier.

RUNNING EXAMPLE

ABC Construction gets a clean test site

Our project needs Home, About, Services, Projects, Team, and Contact. Projects use a custom post type with Industry and Location taxonomies. Global settings hold contact details and social links.

Theme files



Local site



WordPress data



Test results

Create representative content

- A short project title and a very long project title
- A project with no featured image
- A project with every field complete
- An empty taxonomy archive
- A services page with six cards
- A contact form with required fields

Assign WordPress controls

- Set the homepage under Reading settings
- Assign the primary and footer menus
- Save contact and social settings
- Upload the logo and site icon
- Refresh Permalinks once
- Confirm project archive and singles

Test data should be awkward

Perfect sample content hides defects. Use long headings, missing images, large images, empty fields, many menu items, and several paragraphs. If the design survives awkward content, it is more likely to survive the client's real content.

ABC check: A project must still look complete when its optional Location field is empty. The template should hide the empty row, not print an empty label.

Write a baseline

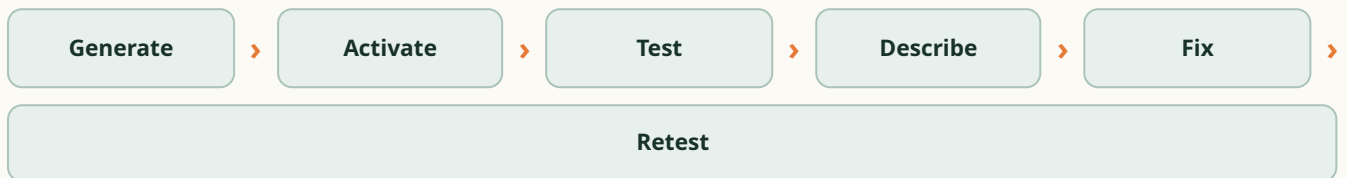
Before asking AI to fix anything, record the WordPress version, active theme, active plugins, PHP version, browser, screen width, page URL, and exact error. This becomes the clean starting point for every repair.

DEBUGGING AI-GENERATED THEMES

Use a calm loop, not random edits

2

AI can write convincing code that is incomplete, placed in the wrong file, or based on an assumption that does not match your project.



- 1 **Reproduce the fault.** Find the shortest repeatable path that causes it.
- 2 **Describe what you expected.** Name the visible or functional result you wanted.
- 3 **Describe what happened.** Quote the error text or explain the exact behavior.
- 4 **Name the location.** Give the page, template, component, and screen size.
- 5 **Limit the repair.** Tell AI what must remain unchanged.
- 6 **Request a test.** Ask it to explain how you can verify the fix.
- 7 **Retest nearby behavior.** A menu fix must be checked on desktop and mobile, with keyboard and pointer.

Do not paste secrets

Never include passwords, private keys, database credentials, client personal data, or production tokens in an AI prompt. Use safe example values and approved secret storage.

Turn on useful development signals

In a development copy, WordPress debugging can reveal warnings and deprecated calls. Keep public error display off on a live site. The WordPress Theme Handbook explains the available debugging constants and recommends fixing PHP, WordPress, HTML, CSS, and JavaScript errors before release.

BUG REPORTS

Exact language produces testable repairs

WEAK

“Menu broken.”

The AI must guess the page, device, action, expected result, and failure.

STRONG

Describe the observable fault

“On the homepage at 390px width, the mobile menu button is visible. Clicking it does nothing. No menu panel opens. Preserve the desktop header and current styles. Inspect the existing JavaScript enqueue and button event, fix only the mobile toggle, then give me a five-step test.”

A reusable bug report

```
Project: ABC Construction WordPress theme
Page: /projects/riverside-office/
Environment: LocalWP, PHP [version], WordPress [version]
Screen: 390 x 844, [browser]
Action: Tap the mobile menu button
Expected: Navigation panel opens and focus moves inside it
Actual: Nothing changes
Console or PHP error: [paste exact message, with secrets removed]
Constraint: Do not redesign the header
Test: Explain how to verify pointer, keyboard, Escape, and desktop behavior
```

If the site shows a fatal error

1. Record the exact message and file line.
2. Switch to a default theme if admin access remains available.
3. Restore the last known working change if needed.
4. Ask for the smallest repair.

If the page looks wrong

1. Capture the URL and viewport.
2. Inspect whether markup or CSS is missing.
3. Name the first visible difference.
4. Check after the fix at nearby widths.

My rule: A repair is not complete because the error disappeared. It is complete when the intended behavior works and nearby behavior still works.

PRE-DELIVERY CHECKLIST

3

Functionality must pass as one system

Navigation and pages

- ☐ Logo links to the homepage
- ☐ Primary menu uses WordPress menus
- ☐ Mobile menu opens, closes, and traps no focus
- ☐ Current page state is visible
- ☐ Header and footer links are correct
- ☐ All buttons reach the intended page
- ☐ Search returns useful results
- ☐ 404 page gives a way forward

Forms and content

- ☐ Required fields are clear
- ☐ Validation messages are understandable
- ☐ Success message appears once
- ☐ Submission reaches the correct recipient
- ☐ Spam protection is appropriate
- ☐ Phone and email links work
- ☐ Empty optional fields disappear cleanly
- ☐ Images have useful alternative text

Projects content type

- ☐ Project can be created and edited
- ☐ Archive lists published projects
- ☐ Single project uses the correct template
- ☐ Industry and Location filters work
- ☐ Pagination keeps the design
- ☐ Featured image fallback is intentional
- ☐ Related projects do not repeat the current item

WordPress behavior

- ☐ Editor content matches the front end
- ☐ Menus can be reassigned
- ☐ Settings save and sanitize values
- ☐ Theme has no visible debug notices
- ☐ Logged-in admin bar does not break layout
- ☐ User roles have only needed access
- ☐ Core, plugins, and theme are current

Release gate: Mark each item Pass, Fail, or Not applicable. "Not tested" is not the same as "Pass."

PRE-DELIVERY CHECKLIST

Check the experience at real sizes

View	What to inspect	ABC Construction example
Wide desktop	Line length, empty space, large images, full navigation	Projects grid stays balanced at four cards
Laptop	Header fit, card widths, forms, sticky elements	Contact form remains beside office details
Tablet	Two-column collapse, touch targets, landscape	Service cards become two columns
Mobile	Menu, reading order, button width, overflow	Project metadata stacks in a clear order
Very small	Long words, 320px width, zoom, horizontal scroll	Long project title wraps without clipping

Brand

- ☐ Correct logo on light and dark areas
- ☐ Favicon and site icon display
- ☐ Brand colors are consistent
- ☐ Type hierarchy is consistent
- ☐ Default social image is uploaded
- ☐ No starter text or old client name remains

Accessibility basics

- ☐ Keyboard reaches every control
- ☐ Focus indicator is visible
- ☐ Headings follow a useful order
- ☐ Form fields have labels
- ☐ Text contrast is readable
- ☐ Motion does not block use

A five-minute keyboard pass

Start at the browser address bar and press Tab. You should see focus move through links and controls in a logical order. Open and close the mobile menu with the keyboard. Submit the form with a missing required field. Press Escape where a panel or dialog should close.

Zoom test: Increase browser zoom to 200 percent. Content should reflow without losing controls or forcing the reader to scroll sideways for ordinary text.

PRE-DELIVERY CHECKLIST

SEO is a release gate, not decoration

Rank Math can help manage metadata, sitemaps, canonical URLs, schema, and social previews. The plugin does not choose the right business meaning for you. Review every important template and page.

Indexing and URLs

- ☐ Production discourages search engines setting is off
- ☐ Local and staging sites remain blocked
- ☐ Preferred domain and HTTPS are consistent
- ☐ Permalinks are readable and saved
- ☐ Canonical URLs point to the live page
- ☐ Old URLs have planned redirects
- ☐ XML sitemap opens and lists wanted content

Schema

- ☐ Organization or LocalBusiness data is accurate
- ☐ Pages use a suitable content type
- ☐ Breadcrumbs match visible navigation
- ☐ Project schema is truthful, not invented
- ☐ No duplicate schema fights another plugin

Page metadata

- ☐ Every key page has a unique title
- ☐ Descriptions explain the page honestly
- ☐ Only one clear H1 is used
- ☐ Open Graph title, description, and image work
- ☐ Social image is not stretched or missing
- ☐ Author, dates, and business details are correct
- ☐ No local domain appears in source or metadata

ABC Construction

- ☐ Services page targets services, not the company name alone
- ☐ Project pages have unique titles
- ☐ Industry archives have useful introductions
- ☐ Contact details match the website
- ☐ Project images have descriptive alt text

Do not chase a plugin score

A green indicator does not prove that a page is useful, accurate, or ready to rank. Use it as a checklist signal, then read the page as a real visitor.

PRE-DELIVERY CHECKLIST

Speed and security need controlled changes

Performance

- ☐ Compress oversized images
- ☐ Serve appropriate image dimensions
- ☐ Remove unused libraries and assets
- ☐ Load scripts only where needed
- ☐ Test page cache on public pages
- ☐ Retest forms, menus, sliders, and filters after optimization
- ☐ Measure key pages on mobile

Production safety

- ☐ HTTPS works without warnings
- ☐ WordPress and Site Address use HTTPS
- ☐ Strong, unique admin access is in place
- ☐ Unused admin accounts are removed
- ☐ Core, plugins, and theme are updated
- ☐ Backups are configured and restorable
- ☐ Debug display is off on production

Use one caching system deliberately

LiteSpeed Cache is a sensible option when the host runs compatible LiteSpeed server caching. Enable a conservative baseline first. Purge the cache, then test. Add CSS, JavaScript, image, or object-cache optimizations one group at a time. A fast page that loses its menu or form is not an improvement.

WP Rocket and other premium tools are optional. They may make configuration easier, but they are not required to understand the workflow. Do not run several page-cache plugins together.

Change, purge, test: After every optimization change, purge all relevant caches and check a logged-out browser session. Test the homepage, contact form, one project archive, and one project single.

Client cost: Local development can be done with free tools. A public website still needs production hosting, a domain, and possibly paid services. Put these costs in the proposal instead of hiding them.

GOING LIVE

Prepare before you move anything

4

Migration is not “upload and hope.” It is a controlled change from a known local state to a known production state.

Collect access

- Hosting control panel
- Production WordPress Admin
- Domain and DNS management
- SSL controls
- Backup destination

Record the release

- Local site URL
- Live domain
- Theme version or commit
- Plugin list and versions
- Migration file name and date

Take a pre-migration backup

If a live site already exists, back it up before import. Confirm how to restore it. An export that has never been tested is only a hopeful file.

Choose the launch window

Use a period when the client can review and when you can fix problems. Avoid starting a migration just before you lose access or the client begins a campaign.

Import replaces data

A full-site migration can replace the destination database, themes, plugins, and media. Confirm the destination and backup before you press Import.

Prepare ABC Construction

- ☐ Client has approved the latest local review
- ☐ All project records are present
- ☐ Contact form destination is confirmed
- ☐ Production domain is confirmed
- ☐ Old website backup can be restored
- ☐ Maintenance message and rollback owner are agreed

MIGRATION WORKFLOW

Move Local to production in a controlled order

- 1 Export the local site.** In Local WordPress Admin, use All-in-One WP Migration to export to a file. Keep the resulting .wpress file safely.
- 2 Prepare live WordPress.** Create or confirm the destination installation. Install and activate the same migration plugin.
- 3 Import on the live site.** Select the export file. Read the replacement warning carefully and confirm only when the destination is correct.
- 4 Sign in again.** The imported database can replace destination users. Use the credentials from the imported site if required.
- 5 Save permalinks.** Open Settings, then Permalinks, and save the selected structure again.
- 6 Clear caches.** Purge plugin, server, content delivery network, and browser caches as relevant.
- 7 Check URLs.** Search for the old local domain in content, menus, widgets, metadata, canonicals, and serialized settings.
- 8 Run a production test.** Repeat the critical path against the public domain before inviting the client.

File size limits: Hosting upload limits can block a large migration file. Check the host's current limits and the migration plugin's current capabilities before launch. Do not promise a method until the actual file imports successfully.

What I never delete immediately

I keep the verified local copy, the last production backup, and the migration export until the live site has passed review. The client should also have an ongoing backup plan after handoff.

AFTER MIGRATION

DNS and SSL are part of the website

DNS tells visitors where the domain should go. SSL provides the certificate that allows the site to load securely over HTTPS. The hosting provider and domain provider may use different names for these controls.

Domain and HTTPS

- ☐ Domain points to the correct hosting
- ☐ Both preferred and alternate hostname behavior is intentional
- ☐ Valid SSL certificate is active
- ☐ HTTP redirects to HTTPS
- ☐ No mixed-content warning appears
- ☐ WordPress and Site Address use HTTPS

Final live path

- ☐ Homepage opens in a private window
- ☐ Primary menu works
- ☐ Project archive and single open
- ☐ Contact form delivers a real test
- ☐ Search and 404 work
- ☐ Sitemap opens
- ☐ Social preview uses the right image

ABC Construction smoke test

Action	Expected result
Open Home, Services, Projects, Contact	All load over HTTPS with no local URL
Filter Projects by Industry	Matching cards appear and URL is sensible
Open Riverside Office project	Fields, image, related work, and CTA appear
Send contact form	Success message appears and recipient confirms delivery
Open a fake URL	Helpful branded 404 page returns

DNS can take time to update. Do not repeatedly change records because one network still shows the old site. Confirm the intended records, then test from more than one network or resolver.

FULL CLIENT WORKFLOW

5

ABC Construction in six working days

Day	Work	Proof before moving on
1	Interview client. Define users, pages, actions, content model, admin needs, SEO needs, and success criteria.	Approved requirement document and sitemap.
2	Write the Lovable prompt. Generate the main UI, then review structure before details.	Homepage and key page direction approved.
3	Refine responsive states and components. Sync to GitHub and record the final UI version.	Complete page set, no unrequested features.
3-4	Ask the coding AI to inspect first. Convert the UI into a native WordPress theme.	Theme activates and core templates render.
4	Add Projects, taxonomies, fields, menus, and global settings. Keep lasting content structures portable.	Client-editable content works from admin.
5	Test in LocalWP. Repair exact bugs. Configure SEO and a conservative performance baseline.	Release checklist passes.
6	Back up, migrate, connect HTTPS, test production, invite client review, and prepare handoff.	Signed review or recorded change list.

This is an example schedule, not a promise. Project size, feedback, content readiness, integrations, and hosting access can change the timing.

Where projects usually slow down

The code is not always the slow part. Missing content, unclear approval, late access, and undefined admin behavior can stop delivery. The method makes these dependencies visible early.

THE METHOD

The Prompt-to-Production WordPress Method

This is my name for the complete workflow. It turns a client need into a working WordPress site through a sequence of decisions and proof.



What stays human

- Understanding the client
- Choosing the content model
- Deciding what must be editable
- Setting constraints
- Testing the result
- Accepting release responsibility

What AI accelerates

- Explaining an unfamiliar codebase
- Drafting repetitive theme files
- Mapping UI parts to templates
- Producing a first repair
- Refactoring after tests
- Writing clear handoff notes

Client handoff package

- ☐ Live site URL and approved scope
- ☐ Administrator access shared securely
- ☐ Theme and companion plugin source
- ☐ Backup and restore instructions
- ☐ How to edit Projects, menus, and global details
- ☐ Plugin and hosting renewal responsibilities
- ☐ Known limits and future improvement list
- ☐ Support window and change-request process

Delivery is teaching. Record a short walkthrough or hold a live session. Ask the client to create one test project while you watch. Their success proves the admin experience is understandable.

PROMPT ENGINEERING

6

Give AI six pieces of a real task

CONTEXT + LOCATION + ACTION + REQUIREMENT +
CONSTRAINT + TEST

Context

What project is this? What architecture and purpose already exist?

Location

Which file, template, function, component, or screen is involved?

Action

What exactly should change?

Requirement

What behavior must the finished result provide?

Constraint

What must not change? What standards or boundaries apply?

Test

How will the reader prove that the change works?

One complete prompt

```
Context: This is the ABC Construction native WordPress theme.  
Location: The primary navigation is currently hard-coded in header.php.  
Action: Replace it with a registered WordPress navigation menu.  
Requirement: I will assign the menu from Appearance > Menus, and it must support the current  
desktop and mobile navigation.  
Constraint: Preserve the existing HTML classes, spacing, hover styles, and mobile layout. Do not  
redesign the header.  
Test: Explain how to assign the menu, then verify desktop links, the mobile toggle, keyboard  
focus, Escape, and the fallback when no menu is assigned.
```

The formula does not need these labels every time. The important part is that all six ideas are present.

PROMPT CLINIC

Turn vague requests into bounded work

WEAK

“Make the header editable.”

No location, control type, preserved behavior, or test is named.

PRECISE

“In header .php, replace only the hard-coded logo, phone number, and primary menu with WordPress-controlled values. Use the existing logo setting, a sanitized global phone setting, and the registered primary menu. Preserve all classes and responsive behavior. Then list the admin steps and test the header at 390px and 1440px.”

WEAK

“Add projects.”

PRECISE

“In the ABC companion plugin, register a public project post type with archive slug projects, title, editor, thumbnail, and REST support. Register Industry and Location taxonomies. Do not place this content model in the theme. Explain how to refresh permalinks and test the admin, archive, and one single project.”

WEAK

“Fix mobile.”

PRECISE

“On single-project.php at 320px to 480px, the metadata grid creates horizontal scrolling. Keep desktop unchanged. Make the grid one column below 640px, allow long values to wrap, and preserve a 44px touch target for the CTA. Test at 320, 390, 480, 768, and 1440px.”

Before modification: For a new codebase, first ask AI to inspect the project and explain its structure without changing files. Agreement on the map prevents careless rewrites.

PROMPT LIBRARY

Theme conversion and admin controls

7

Replace names, paths, and breakpoints with your project values. Ask the AI to show changed files and a verification plan.

1. Inspect first: Inspect this WordPress project without modifying files. Map the theme entry points, templates, assets, menus, settings, dynamic content, and likely risks. Then propose a conversion plan.

2. Native conversion: Convert this UI into a native WordPress theme. Preserve the visual system. Do not embed the original app. Map reusable UI into template parts and WordPress data.

3. Theme foundation: Create the minimum valid classic theme foundation with style.css, functions.php, index.php, header.php, footer.php, and correct asset enqueueing. Explain each file.

4. Homepage: Map the existing homepage into front-page.php. Keep content order and responsive behavior. Use WordPress-controlled content only where an administrator reasonably needs it.

5. Standard page: Create page.php for ordinary pages. Preserve the site shell and render the page title and content safely.

6. Post single: Create single.php with title, author, date, featured image, content, categories, and accessible previous and next links.

7. Archive: Create archive.php with an accessible heading, post cards, empty state, and WordPress pagination. Preserve existing card classes.

8. 404: Create a branded 404 template with a plain explanation, search form, homepage link, and no dead-end controls.

9. Template part: Extract the repeated project card into template-parts/card-project.php. Pass data through

WordPress functions and escape output.

10. Assets: Audit current CSS, JavaScript, fonts, and images. Enqueue only needed local assets with correct dependencies and cache-busting versions.

11. Primary menu: Replace the hard-coded header links with a registered primary menu. Preserve current markup, classes, desktop layout, and mobile behavior.

12. Footer menu: Register and render a footer menu. Keep legal links separate. Provide an intentional fallback when no menu is assigned.

13. Logo: Replace the fixed logo file with WordPress custom-logo support. Preserve displayed dimensions and provide a text site-title fallback.

14. Contact settings: Add sanitized global fields for phone, email, and address. Use them only in the existing header, footer, and contact components.

15. Social settings: Add URL fields for approved social networks. Hide an icon when its URL is empty. Open external links safely and label them clearly.

16. Footer text: Make the footer credit editable with a sensible default. Escape output and do not allow unsafe markup.

17. Theme supports: Review required theme supports for the target WordPress version. Add only those the theme actually uses and explain why.

18. Admin guide: Write a one-page admin guide for assigning menus, changing the logo, editing global contact details, and updating footer links.

PROMPT LIBRARY

Content models, archives, and queries

The UI tells you what data is needed. Long-lived client content should remain available even if the visual theme changes.

19. Project CPT: In the companion plugin, register a Project post type with archive support, REST support, title, editor, excerpt, and featured image. Prefix identifiers.

20. Industry taxonomy: Register a hierarchical Industry taxonomy for Projects. Add clear labels, REST support, and archive URLs. Explain permalink refresh.

21. Location taxonomy: Register a non-hierarchical Location taxonomy for Projects and attach it only to the Project post type.

22. Custom slug: Change the Project public archive slug to projects while keeping the internal identifier unchanged. State the redirect and permalink steps.

23. Project fields: Add fields for client, completion year, location summary, and project size. Sanitize on save, check permissions, and handle missing values.

24. Archive template: Create archive-project.php using the existing project card. Include heading, pagination, empty state, and useful filter links.

25. Single template: Create single-project.php with hero, project facts, gallery, description, related projects, and CTA. Hide empty optional fields.

26. Taxonomy template: Create taxonomy-project_industry.php with the term name, description, matching projects, pagination, and empty state.

27. Related projects: Query three related Projects by shared Industry, exclude the current post, avoid duplicates, and reset post data.

28. Featured projects: Add a boolean featured field and query up to three featured Projects on the homepage with a fallback to recent Projects.

29. Project ordering: Add an admin-selectable project order field and sort the archive by that value, then date, without changing admin search.

30. Pagination: Replace the static load-more control with accessible WordPress pagination that preserves the current archive design.

31. Empty state: Create an honest empty state for the Projects archive with a link to Services and no fake project cards.

32. Search scope: Include Projects in front-end search results while leaving admin queries and unrelated archives unchanged.

33. Query audit: Inspect every WP_Query in the theme for pagination, post status, performance, escaping, and wp_reset_postdata. Report before editing.

34. Field fallback: For each project field, define whether to hide the row, show a neutral fallback, or block publication. Implement only approved behavior.

35. Content portability: Move Project registration, taxonomies, and lasting metadata from the theme into a companion plugin without losing existing records.

36. Migration check: List every place the old local URL might remain after migration, including content, menus, media, options, metadata, canonicals, and caches.

PROMPT LIBRARY

Quality, accessibility, and fixes

A prompt should name an observable failure or outcome. Replace bracketed fields with facts from your site.

37. PHP fatal: Diagnose this exact PHP fatal error: [message]. Inspect the named file and line, explain the cause, make the smallest safe fix, and give a regression test.

38. PHP warning: Fix this warning without hiding errors: [message]. Explain which value can be missing and add correct validation or fallback.

39. JavaScript failure: On [page] at [width], [control] does [actual] instead of [expected]. Inspect enqueueing and console errors. Preserve current markup and styles.

40. Mobile menu: Repair the mobile menu toggle. Support keyboard activation, visible expanded state, Escape to close, and focus return. Keep desktop unchanged.

41. Horizontal scroll: Find the exact element causing horizontal scroll at 320px. Fix its sizing or wrapping without applying overflow-x hidden to the whole page.

42. Responsive grid: Make [grid] use one, two, and three columns at approved breakpoints. Preserve card order and prevent stretched images.

43. Form labels: Audit the contact form for visible labels, instructions, required state, errors, keyboard use, and success feedback. Fix only confirmed issues.

44. Heading order: Audit one rendered page for heading order and landmark structure. Do not choose headings for visual size. Report changes.

45. Focus styles: Add a visible focus style for links, buttons, form controls, and menu toggles that meets the existing palette and is not removed by mouse styles.

46. Alt text behavior: Use the WordPress attachment alt text for meaningful images. Use empty alt for purely decorative images. Do not copy filenames into alt text.

47. Image sizes: Review each template image request. Use appropriate WordPress image sizes and srcset, preserve aspect ratio, and avoid loading full originals unnecessarily.

48. Asset audit: List CSS and JavaScript loaded on each page. Remove duplicates and load feature-specific assets only where needed.

49. Cache regression: After cache optimization, test navigation, forms, filters, logged-in admin behavior, and dynamic project pages. Identify any script that must be excluded.

50. SEO output: Inspect rendered head output for one title, one meta description, canonical, robots, Open Graph, and schema. Report conflicts before editing.

51. Local URL scan: Search project files and database-safe exports for the local hostname. Do not replace serialized data with a raw text edit. Propose a safe method.

52. 404 regression: Test a missing page, missing Project, empty taxonomy, and removed URL. Confirm the correct status and a useful 404 page.

53. Release review: Review the change set for debug code, unfinished comments, sample content, secrets, unused assets, and accidental design changes. Report findings only.

54. Handoff notes: Write plain-language client instructions for editing Projects, menus, contact details, images, and SEO fields. Include what not to change.

DO NOT LET AI DO THIS

8

Protect the project from confident shortcuts

1. Rewrite everything

Vague: "Clean up the theme."

Precise: "Audit first. Change only duplicated project-card markup. Preserve public classes and behavior. Show the changed files and test archive, search, and related projects."

2. Hard-code client content

Vague: "Add the company phone."

Precise: "Read the phone from the approved global setting. Sanitize on save, escape on output, and hide the link when empty."

3. Put business data in the theme

Vague: "Create a Projects CPT in functions.php."

Precise: "Register lasting Project content and taxonomies in the companion plugin so the records remain portable when the theme changes."

4. Claim success without a test

Vague: "Fix the form."

Precise: "Fix the required-email validation, then test empty, invalid, and valid submissions. Confirm visible errors, keyboard focus, success feedback, and actual delivery."

Four boundaries I keep

1. **No secret sharing.** Credentials stay out of prompts, code, repositories, screenshots, and logs.
2. **No silent redesign.** A functional change does not authorize a visual rewrite.
3. **No invented content.** Empty states are better than fake testimonials, projects, or business facts.
4. **No untested "done."** Every change names the evidence that proves it works.

AI output is a draft. Read changed files, test the result, and keep a rollback point. Speed comes from a disciplined loop, not from skipping review.

TEAR-OUT WORKFLOW

The 30-Step AI WordPress Workflow

9

- 1 **Collect requirements.** Record business, users, goals, content, and constraints.
- 2 **Define the sitemap.** List pages and navigation.
- 3 **Define the content model.** Separate pages, posts, custom types, fields, and taxonomies.
- 4 **Define functionality.** Forms, search, filtering, logins, and integrations.
- 5 **Prepare the Lovable prompt.** Give pages, sections, brand, behavior, and exclusions.
- 6 **Generate the UI.** Start with structure.
- 7 **Review the UI.** Compare it with the approved requirement.
- 8 **Iterate.** Fix layout, type, components, and responsive behavior.
- 9 **Finalize pages.** Confirm every state and page type.
- 10 **Sync GitHub.** Create a traceable handoff from design to code.
- 11 **Download the repository.** Keep a clean local copy.
- 12 **Create a LocalWP site.** Set up private WordPress.
- 13 **Create the theme directory.** Use a clear, unique slug.
- 14 **Place the project.** Put source and assets in the intended working location.
- 15 **Ask AI to inspect.** No modifications yet.

Gate after step 15: You can explain the current UI structure, the planned WordPress architecture, what becomes editable, what stays presentational, and the main conversion risks.

Print this page and the next page together. Tick a step only when its output exists.

TEAR-OUT WORKFLOW

Steps 16 to 30

- 16 Convert the UI.** Build native WordPress theme files.
- 17 Create templates.** Cover home, page, post, archive, search, and 404.
- 18 Register menus.** Replace static navigation.
- 19 Create global settings.** Expose only useful controls.
- 20 Create custom post types.** Keep portable content in a plugin.
- 21 Create taxonomies.** Model useful grouping.
- 22 Create fields.** Match the information shown by the UI.
- 23 Build archives.** Include pagination and empty states.
- 24 Build single views.** Handle complete and missing data.
- 25 Test WordPress behavior.** Edit, save, query, navigate, search.
- 26 Test responsive behavior.** Include keyboard, zoom, and small screens.
- 27 Configure SEO and performance.** Start conservatively and retest.
- 28 Export and migrate.** Back up first.
- 29 Verify production.** Check HTTPS, URLs, forms, metadata, and caches.
- 30 Deliver and teach.** Give access, files, instructions, responsibilities, and support terms.

Gate after step 30: The live site passes its critical paths, the client can perform normal updates, a restore path exists, and any remaining work is written down.

The method in one sentence

Turn a requirement into a design, turn the design into WordPress, prove the system locally, move it carefully, then teach the client to own it.

FINISH WELL

Your first real delivery starts small

Choose a project you can understand completely. A clear five-page business site with one useful content type will teach you more than a large build you cannot test.

Before you say yes to the client

- ☐ You can state the scope in one page
- ☐ You know who supplies words, images, and legal text
- ☐ You know which content the client must edit
- ☐ You know which external services are required
- ☐ Hosting, domain, plugins, and renewal costs are clear
- ☐ Approval points and support boundaries are written

Use AI to move faster, but let evidence decide when the work is ready.

Reference desk

Interfaces and plugin behavior change. These official or project-maintained pages are starting points for current details:

- [WordPress Theme Handbook: Theme Testing](#)
- [WordPress Theme Handbook: Debugging](#)
- [WordPress Plugin Handbook: Registering Custom Post Types](#)
- [WordPress Plugin Handbook: Working with Custom Post Types](#)
- [All-in-One WP Migration plugin page](#)
- [LiteSpeed Cache plugin page](#)

Product names belong to their respective owners. This book is an independent practical guide and does not imply endorsement. "From Lovable to WordPress" is a working series title and may be changed before publication.

Now build one project. Keep the requirements close, make one controlled change at a time, and finish every step with a real test.