WordPress Developer Handoff Guide (Comprehensive)

A complete, practical guide for transferring a WordPress project from one team to another. It merges strategic, legal, operational, and technical handover material into one definitive reference, with checklists and templates you can use immediately.

A successful handover is not just a transaction of files and passwords; it is a comprehensive process of transferring knowledge, responsibility, and control. The stakes of a poorly executed handover are high: security vulnerabilities from lingering access, operational downtime due to unknown dependencies, legal liabilities from ambiguous ownership, and escalating costs from undocumented technical debt. This guide is designed to mitigate those risks.

Use this as a living document during the handover. Bring it to the handover meeting, check off items as you verify them, and keep the inventories up to date.

How to use this guide

- Start with the Quickstart and Final Handover Checklist to frame the transfer.
- Work top-to-bottom. Each section ends with concrete actions and data to collect.
- Log every credential, license, and account in the Master Inventory table.
- **Security Protocol:** Share credentials securely (e.g., via a password manager shared vault like 1Password/Bitwarden or a one-time secret link). **Never send passwords via email.**
- **Immediate Action:** Change passwords and enable 2FA (Two-Factor Authentication) as soon as access is confirmed.
- **Final Step:** Remove the outgoing developer's access everywhere before you sign off.

Table of contents

Quickstart & Success criteria

Minimum bar for a successful handoff:

- **Ownership:** You have full administrative and legal control of the domain, DNS, hosting, WordPress, and all integrated services. Billing and renewal responsibility are in your accounts.
- **Access:** All credentials are in your password manager. Every admin login tested successfully.
- **Security:** Passwords rotated, 2FA enabled, previous access fully revoked, and API keys regenerated where applicable.
- **Backups:** Verified working backups (stored off-site) and a documented, tested restore procedure.
- **Operations:** Clear, documented deployment process, update cadence, and monitoring in place.
- **Knowledge:** Architecture, customizations, dependencies, and "quirks" are documented; training delivered.

Strategic overview & History

Understand the purpose and history before changing anything. The technical architecture can only be judged against the business goals it was intended to achieve.

- **Core Purpose:** What problem does the site solve? Primary objectives (e.g., e-commerce, lead gen, publishing).
- **Audience:** Who is the audience, and how has it evolved? What user personas were defined?
- **History & Evolution:** Key milestones, major feature additions, pivots, redesigns, or platform migrations.
  - *Rationale:* A project's history often reveals its compromises. Understanding historical pivots (e.g., a site that had e-commerce "bolted on" later) helps uncover technical debt and anticipate areas of fragility.
- **KPIs:** KPIs for success (conversions, revenue, engagement) and how they're measured.
- **Business Rules & Compliance:** Any unique business rules, compliance constraints (e.g., industry-specific regulations), or seasonal patterns.

ACTIONS:

- [ ] Capture the answers above in the project notes.
- [ ] Identify any misalignments between current architecture and business goals.

Stakeholders & roles

Map the influence and responsibility network to ensure clear communication.

- **Key Personnel:** Decision makers and points of contact (product, content, marketing, tech).
- **Workflows:** Approval workflow for content and technical changes; escalation path for incidents.
- **Historical Knowledge:** Who managed discovery/planning? (These individuals can clarify historical decisions and original requirements).
- **Post-Handover Management:** Who will own the site post-handover (content management, technical maintenance, business decisions).

ACTIONS:

- [ ] Record stakeholder list with roles and contact info.
- [ ] Agree on post-handover support window (duration, scope, rates) and communication channels.

Ownership & legal

This is critical. Failure to secure unambiguous ownership of all assets can have catastrophic long-term consequences.

- **Intellectual Property (IP):** Written confirmation of 100% ownership of website, content, custom code, graphics, design elements, and data upon final payment. This is non-negotiable. Identify exceptions (e.g., stock photos).
- **Licenses:** List of all premium themes/plugins and other software licenses.
  - *Rationale:* Active licenses are crucial for receiving security updates and support. Expired licenses leave the site vulnerable. Detail transfer or repurchase requirements.
- **Contracts and SLAs:** Ongoing obligations (hosting, maintenance, support); renewal dates, costs, and billing contacts.
- **Privacy Compliance (GDPR/CCPA):** Data collection points (forms, e-commerce), PII (Personally Identifiable Information) storage location and security, retention policies, cookie consent mechanisms, and data sharing agreements.

ACTIONS:

- [ ] Secure written confirmation of IP transfer.
- [ ] File copies of agreements; update billing to new owner.
- [ ] Populate license inventory (Appendix) with keys, renewal dates, and accounts.
- [ ] Review privacy compliance documentation.

Domain & DNS

Control over the domain registrar and DNS is paramount. Losing control can result in the loss of the domain name.

- **Registrar:** Provider (e.g., GoDaddy, Namecheap), account owner, renewal date, and auto-renew status.
- **DNS Provider:** (e.g., registrar, Cloudflare, etc.) and complete record set (A, AAAA, CNAME, MX, TXT).
- **Email Security Records:** SPF, DKIM, DMARC (essential for email deliverability).
- **Scope:** Subdomains, redirects, staging domains, and special rules.

ACTIONS:

- [ ] Gain registrar and DNS admin access; transfer account ownership and billing.

- [ ] Enable 2FA on registrar and DNS accounts immediately.
- [ ] Export current DNS zone file and attach to this document.

Hosting & server infrastructure

The hosting environment dictates performance, security, and scalability.

- **Provider & Plan:** Hosting provider, plan name, cost, renewal date, and geographic region.
- **Environment Type:** (Shared, VPS, Dedicated, Managed WordPress).
  - *Rationale:* This defines the level of control and performance isolation (e.g., Shared hosting risks "noisy neighbors"; Managed hosting may restrict plugins).
- **Resource Limits:** CPU, RAM, storage, bandwidth. Crucially, PHP memory limits and execution time limits (insufficiency causes errors).
- **Server Stack & Versions:** OS, Web server (Apache/Nginx), PHP + required extensions, MySQL/MariaDB. Control panel (cPanel/Plesk/custom).
- **Server-level Security:** Web Application Firewall (WAF/ModSecurity), firewall rules, IP restrictions, and allowlists.
- **Custom Configurations:** Any modified `php.ini` settings, custom `.htaccess` or `nginx.conf` rules.

ACTIONS:

- [ ] Verify access to hosting dashboard and control panel.
- [ ] Enable 2FA on the hosting account.
- [ ] Document any non-standard server configs and required PHP extensions (crucial for future migrations).

Database access & structure

The database stores all content, settings, and user information.

- **Connection Details:** DB host, port, engine/version, database name(s), users and privileges.
- **Access Method:** (phpMyAdmin, Adminer, CLI) and credentials.
- **Structure:** Any custom tables or stored procedures beyond standard WordPress schema (document purpose and schema).
- **Maintenance:** Optimization procedures or cleanup routines for logs/temporary data.

ACTIONS:

- [ ] Perform a read-only smoke check of DB access.
- [ ] Export schema for reference.
- [ ] Rotate database user passwords.

WordPress access & user management

Ensure a clean audit trail and secure access control by enforcing the Principle of Least Privilege (PoLP).

- **New Administrator Account:** Provide a NEW administrator account for the new owner/team. **Do not re-use existing logins.**
- **User Inventory:** Review all users and roles (Admins, Editors, Authors, etc.). Identify accounts to remove or demote.
- **PoLP Enforcement:** Ensure users only have the minimum access required for their role (e.g., writers do not need plugin installation access).
- **Security Features:** 2FA enforcement, login restrictions (IP whitelisting, login attempt limits), password policies.

ACTIONS:

- [ ] Create and test new admin(s).
- [ ] Enable and enforce 2FA for all admin/editor roles.
- [ ] Rotate passwords for all necessary service accounts.
- [ ] Remove previous developer/admin access after confirmation.

Server access (FTP/SFTP/SSH)

Direct file system access is necessary for advanced troubleshooting, deployment, and managing permissions.

- **Connection Details:** SFTP/SSH host, port, users/keys, and required security protocols.
- **File Structure:** Document root path, location of logs, backup directories, symlinks.
- **Permissions:** Special file permissions (ownership settings, permissions for uploads/cache directories).
- **SSH Specifics:** Sudo privileges and restrictions; location of `authorized_keys`.

ACTIONS:

- [ ] Exchange SSH public keys via a secure channel; validate login and permissions.
- [ ] Store keys and connection profiles in your password manager.
- [ ] Remove previous developer's SSH keys and FTP accounts.

Email configuration

Document both domain email hosting (mailboxes) and transactional email paths (website-sent emails).

- **Domain Email Hosting:** Where mailboxes are hosted (e.g., Google Workspace/365/hosted). Admin access, account list, alias/forward rules.
- **Transactional Email Path:** How the website sends emails (WordPress default mail, SMTP plugin, or service like SendGrid/Postmark).
- **Credentials/API Keys:** SMTP settings or API keys for the transactional service.

ACTIONS:

- [ ] Verify SPF, DKIM, DMARC records are correctly configured (See Domain & DNS section).
- [ ] Document SMTP settings and rate limits.
- [ ] Regenerate transactional email API keys or SMTP passwords.

Technical Architecture: Themes, Plugins, and Custom Code

Understanding how the website was built is a prerequisite for maintaining and extending it. This section creates the technical blueprint.

Source Code Version Control (Git)

The use of Git is a hallmark of professional development.

- **Repository:** Location (GitHub, Bitbucket, GitLab) and access.
  - *Rationale:* The absence of version control is a major red flag, suggesting risky "cowboy coding" practices (e.g., editing live via FTP).
- **Branching Model:** Strategy used (e.g., GitFlow, feature branches). Which branch represents production?
- **History:** Confirm the repository contains the complete commit history.

Theme Structure

The theme controls visual presentation. How it is structured impacts future updates.

- **Active Theme:** Is it commercial (provide source/license) or custom?
- **Parent/Child Usage:** Confirm a child theme is used for all customizations.
  - **CRITICAL RISK:** Modifications made directly to a parent theme will be overwritten during updates. If the parent theme is edited, the site becomes difficult or impossible to update safely.
- **Customization Locations:** Where custom CSS, JavaScript, and PHP are stored.

Plugin Ecosystem

Plugins add functionality but also complexity and potential points of failure.

- **Inventory:** Complete list (active/inactive) with version, purpose (rationale for choice), premium/free status, and license details.
- **Known Issues:** Any known conflicts or performance bottlenecks.
- **Custom Plugins:** Any plugins developed specifically for the site. Where is the source code and documentation?

Custom Development & Content Structure

- **Code Locations:** Custom PHP in `functions.php`, code snippets plugins, or custom theme files.
- **Content Structure:** Custom post types (CPTs), taxonomies, and custom fields (e.g., ACF). How were they created (plugin or code)? How are configurations managed (e.g., ACF JSON sync)?
- **Page Builders:** If used (e.g., Elementor, Divi), document global settings and templates.
- **Coding Standards:** Were WordPress Coding Standards followed? Is the code well-commented?
  - *Rationale:* Uncommented code is significantly more expensive and risky to maintain.

ACTIONS:

- [ ] Populate the Plugin & Theme Audit table (Appendix), highlighting risks.
- [ ] Verify child theme usage and integrity of parent theme files.
- [ ] Confirm access to the Git repository and understand the branching strategy.

Third-party services & integrations

Modern websites are ecosystems of interconnected services. Each requires ownership transfer.

- **Inventory:** Comprehensive list of all integrated services:
  - Payments (Stripe/PayPal), CDN/WAF (Cloudflare), Email Marketing (Mailchimp), CRM (Salesforce/HubSpot), Analytics (GA/GTM), Search, Maps, Forms, Shipping/Tax, Transactional Email (SendGrid), etc.
- **For each service:** Admin access, API keys/tokens, billing ownership, renewal dates, integration method, and transfer steps.
  - *Rationale:* Legal and administrative ownership must be transferred, not just login details.

ACTIONS:

- [ ] Transfer ownership of all third-party accounts.
- [ ] Enable 2FA on all third-party services.
- [ ] Regenerate all API keys and tokens after ownership transfer.
- [ ] Update webhooks and callback URLs as needed.

Backups & disaster recovery

A robust backup strategy is the single most important defense against catastrophic events.

- **Backup Solutions:** (Host level, plugin (e.g., UpdraftPlus), third-party). Relying on a single method is risky.
- **Scope & Schedule:** What's included (files, DB, both). Frequency and retention policy.
- **Storage Locations:** Where are backups stored?
  - **CRITICAL:** Backups must be stored off-site (e.g., S3, Dropbox, Google Drive). Backups stored only on the same server are useless if the server fails.
- **Restoration Procedure:** A documented, step-by-step full restoration procedure.
- **Testing:** Date of the last successful restore test. An untested backup strategy is not a reliable strategy.

ACTIONS:

- [ ] Verify off-site backup storage is configured and working.
- [ ] Perform a test restore on a staging environment to validate backup integrity and the procedure.

Security posture

A proactive security posture is a fundamental requirement of ownership.

- **Security Tools:** Security plugins (Wordfence, Sucuri, iThemes Security) and key configurations (Firewall rules, malware scanning schedules, login security).
- **SSL/TLS:** Provider, renewal process (confirm auto-renewal is active), and certificate monitoring. An expired SSL certificate will kill traffic.
- **Access Control:** Adherence to the Principle of Least Privilege (PoLP).
- **Security History:** Past incidents (breaches, malware infections), resolutions, and preventive measures implemented.
- **Monitoring:** File integrity monitoring and suspicious activity detection.

ACTIONS:

- [ ] Review and harden security plugin configurations and WAF/CDN firewall rules.
- [ ] Verify SSL certificate validity and auto-renewal.
- [ ] Audit user roles and permissions.

Performance & caching

Website speed is critical for user experience and SEO.

- **Caching Layers:** Identify all active layers: Plugin (WP Rocket/W3TC), Server-level (Redis/Memcached/Varnish), CDN caching, and Browser caching settings.
- **Cache Purge Procedures:** Step-by-step process to clear all cache layers. Automated cache clearing triggers/hooks.
  - *Rationale:* Essential for troubleshooting when changes do not appear on the live site.
- **Optimizations:** Image optimization/compression, asset minification/concatenation, database optimization.

ACTIONS:

- [ ] Document the exact steps required to "clear all caches" across every layer (Plugin, Server, CDN).

- [ ] Review CDN configuration and page rules.

Analytics & SEO

Tools for understanding traffic, user behavior, and search performance.

- **Access:** Administrative access to Google Analytics (GA4), Google Tag Manager (GTM), and Google Search Console (GSC).
- **Tracking Implementation:** E-commerce tracking, custom events, goals, and cross-domain tracking.
- **SEO Configuration:** SEO plugin used (Yoast/RankMath), XML sitemap location, meta templates, schema markup implementation.
- **Status:** Current rankings and any history of Google penalties or manual actions.

ACTIONS:

- [ ] Verify property ownership and administrative access to all tools.
- [ ] Confirm data streams are active.
- [ ] Capture baseline metrics before making changes.

Environments, version control & deployment

The process of moving changes to the live site is risk-prone. A disciplined, documented workflow is essential for stability.

Environments

- **List:** Development and Staging sites (URLs, access details).
  - *Rationale:* A staging site is crucial for testing updates safely. The absence of one indicates high-risk practices (testing on production).
- **Sync Procedures:** How environments are synchronized (files and database).

Version Control (Git)

> *See also Technical Architecture section*

- **Repository Access & Branching Model.**

Deployment Process

- **Procedure:** The exact, step-by-step process from commit to production (manual vs. automated/CI/CD).
  - **RISK:** An ad-hoc, undocumented, memory-based process (e.g., "I FTP these three files...") is fragile, highly susceptible to human error, and a massive operational risk.
- **Database Changes:** How database changes are managed and migrated between environments (e.g., migration scripts, plugins like WP Migrate DB Pro). DB sync is complex and must be documented.
- **Checklists:** Pre-deploy testing checklist and rollback procedures for failed deployments.

ACTIONS:

- [ ] Document an end-to-end deployment runbook (See Appendix for template).
- [ ] Confirm the staging environment is functional and synchronized.
- [ ] Strictly avoid making changes via FTP directly to production.

Documentation & training

Comprehensive documentation and training are the bridge to confident, independent management.

- **Existing Documentation:** Technical specs, architecture diagrams, admin user manuals, brand/style guides. Where is it stored?
  - *Rationale:* The existence and quality of documentation are strong indicators of a project's health.
- **Training:** Video walkthroughs for common tasks (publishing content, managing products) and custom workflows.
- **Known Issues:** Documentation of known bugs and technical debt.

ACTIONS:

- [ ] Consolidate all documentation in a central location (e.g., with the repository or a knowledge base).
- [ ] Schedule a live training/Q&A session.

Maintenance & monitoring

A website requires continuous care. Outdated software is the leading cause of WordPress vulnerabilities.

- **Update Cadence:** Established schedule for WordPress core, theme, and plugin updates.
- **Testing Procedures:** Updates must be tested on staging before being applied to the live site.
- **Monitoring:** Uptime monitoring (e.g., Uptime Robot, Pingdom), performance monitoring, and error logging.
- **Alerts:** Who receives alerts if the site goes down or errors occur?

ACTIONS:

- [ ] Establish a regular maintenance schedule and testing process.
- [ ] Set up monitoring tools and configure alerting thresholds/recipients.

Final handover checklist

Use this on handover day. Confirm each item before signing off.

- [ ] Schedule and conduct the formal handover meeting.
- [ ] **Verify Admin Access (Log into every service):**
  - [ ] Domain Registrar
  - [ ] DNS Provider (e.g., Cloudflare)
  - [ ] Hosting Control Panel
  - [ ] WordPress Admin Dashboard
  - [ ] SFTP/SSH
  - [ ] Database (e.g., phpMyAdmin)
  - [ ] Email Services (Domain and Transactional)
  - [ ] CDN/WAF
  - [ ] Analytics & SEO Tools (GA, GSC, GTM)
  - [ ] Payment Gateways
  - [ ] Marketing/CRM tools
  - [ ] Git Repository
  - [ ] All other third-party services listed in the Inventory.
- [ ] **Secure Accounts:**
  - [ ] Change ALL passwords and store them in a secure password manager.
  - [ ] Rotate ALL API keys and tokens.
  - [ ] Enable 2FA on all services that support it.
- [ ] **Revoke Previous Access:**
  - [ ] Remove/demote the previous developer’s access from ALL systems listed above.
- [ ] **Validate Operations:**
  - [ ] Validate backups are running and stored off-site.
  - [ ] Perform a test restore on staging.
  - [ ] Smoke-test critical user journeys (purchase/checkout, contact forms, login, search).
- [ ] **Confirm Configuration:**
  - [ ] Confirm monitoring/alerts are going to the correct recipients.
  - [ ] Update the Master Inventory with final ownership status and renewal details.

Exit interview questions

Ask these open-ended questions to uncover “unknown unknowns” and institutional knowledge often missed in checklists.

- **Fragility:** What do you consider the most fragile or complex parts of the site? Are there any known pitfalls or stability issues?
- **Technical Debt:** If you had more time or budget, what would you improve or refactor first, and why? (This reveals the developer's assessment of technical debt).
- **Quirks & Workarounds:** Are there any quirks, oddities, or manual workarounds we absolutely must know about? (e.g., "To update X, you have to do this one weird thing first.") This is often the most valuable question.
- **Lessons Learned:** What were the biggest challenges during development (plugin conflicts, brittle integrations, environment gotchas)? What lessons did you learn?

Appendix: Inventories & templates

Use these tables as your single source of truth. Keep them current.

Master Credentials & Services Inventory

Instructions: Fill a row for every integration. Use a secure method for sharing credentials. After transfer, regenerate API keys/tokens and mark status as ✅ Completed.

Service/Platform	Purpose	Login URL	Username/Email	Password/Key Location	Cost	Renewal Date	Owner	Transfer Status
WordPress Admin	CMS	domain.com/wp-admin	new_admin_user	Password Manager	—	—	New Owner	⏳ Pending
Domain Registrar	Domain Control	[URL]	[Email]	Password Manager	$/yr	YYYY-MM-DD	New Owner	⏳ Pending
DNS Provider	DNS/WAF/CDN	[URL]	[Email]	Password Manager	$/mo	Monthly	New Owner	⏳ Pending
Hosting	Server/cPanel	[URL]	[Email]	Password Manager	$/yr	YYYY-MM-DD	New Owner	⏳ Pending
SFTP/SSH	File Access	[Host/IP]	[User]	SSH Key/PM	—	—	New Owner	⏳ Pending
Database	Data Storage	[URL/Host]	[User]	Password Manager	—	—	New Owner	⏳ Pending
Email (Domain)	Mailboxes	[URL]	[Email]	Password Manager	$/mo	Monthly	New Owner	⏳ Pending
Email (Transactional)	Website Emails	[URL]	[Email]	API Key in PM	$/mo	Monthly	New Owner	⏳ Pending
Backup Service	Backups	[URL]	[Email]	Password Manager	$/mo	Monthly	New Owner	⏳ Pending
Analytics (GA/GTM)	Tracking	[URL]	[Email]	Password Manager	—	—	New Owner	⏳ Pending
Payment Gateway	Payments	[URL]	[Email]	Password Manager	Fees	—	New Owner	⏳ Pending
Git Repository	Version Control	[URL]	[Email]	Password Manager/SSH	$/mo	Monthly	New Owner	⏳ Pending

Plugin & Theme Audit & Risk Assessment

Component	Type	Version	Purpose	Premium?	License Status	Renewal Date	Notes/Risks/Dependencies
[Theme Name]	Parent Theme	x.x.x	Base framework	Premium	Active	YYYY-MM-DD	Core visual framework. **DO NOT** edit directly.
[Child Theme Name]	Child Theme	x.x.x	Customizations	Custom	N/A	N/A	All customizations (CSS/PHP) live here.
WooCommerce	Plugin	x.x.x	E-commerce	Free	N/A	N/A	Critical for sales; complex settings. Test updates thoroughly.
Wordfence	Plugin	x.x.x	Security/WAF	Premium	Active	YYYY-MM-DD	Firewall rules configured. Do not deactivate.
WP Rocket	Plugin	x.x.x	Caching	Premium	Active	YYYY-MM-DD	Key for site speed. Purge after deployment.
ACF PRO	Plugin	x.x.x	Custom fields	Premium	**Expired/Missing**	YYYY-MM-DD	**CRITICAL RISK:** Site depends on this. Cannot update without a valid license.
Classic Editor	Plugin	x.x.x	Editor UI	Free	N/A	N/A	Indicates site may not be compatible with Block Editor (Technical Debt).

Server & Infrastructure Details

Component	Details	Notes
Hosting Plan	Shared/VPS/Dedicated/Managed	Resource limits (CPU, RAM, Storage)
OS	Linux/Windows + version	
Web Server	Apache/Nginx + version	Configuration file location (.htaccess/nginx.conf)
PHP Version	Version	php.ini location, memory limit, execution time limit
PHP Extensions	List of required extensions	Critical for specific plugin functionality
Database	MySQL/MariaDB + version	
Server Caching	Redis/Memcached/Varnish	Configuration details

DNS Records Snapshot

Attach exported Zone File.

Name	Type	Value	TTL	Notes (Purpose)
@	A	[IP Address]	Auto	Main website server
www	CNAME	@	Auto	Redirect/Alias
@	MX	mail.server.com	Auto	Email routing
@	TXT	v=spf1...	Auto	Email Security (SPF)
selector._domainkey	TXT	v=DKIM1...	Auto	Email Security (DKIM)
_dmarc	TXT	v=DMARC1...	Auto	Email Security (DMARC)

Deployment Runbook (Template)

Objective: Deploy changes from Staging to Production.

1. **Preparation:**
   - Work from a feature branch; open Pull Request (PR) to `main`/`production`.
   - Verify staging environment is synced with production data.
   - Run automated tests and perform manual QA on staging.
2. **Backup:**
   - Take fresh backups of Production (files + DB).
   - Confirm a valid restore point exists.
3. **Deployment:**
   - Merge PR.
   - Deploy via CI/CD pipeline or manual Git pull. (Avoid FTP uploads).
4. **Post-Deployment:**
   - Run DB migrations or configuration synchronization steps (if any).
   - Purge all caches (Plugin, Server-level, CDN).
5. **Verification & Logging:**
   - Smoke-test critical user journeys (checkout, forms).
   - Monitor error logs. Roll back immediately if critical issues arise.
   - Log changes in the Change Log (below).

Change Log (Template)

Date	Environment	Change Summary	By	Link (Ticket/PR)	Rollback Notes
YYYY-MM-DD	Prod	[Description]	[Name]	[URL]	[Steps taken]

Handover Sign-Off

- **Date of Handover:** __________
- **Parties Present:** __________
- **Scope Covered:** (e.g., Full website and infrastructure)
- **Outstanding Risks/Issues:** (e.g., ACF license needs renewal by X date)

- **Post-handover Support Agreement:**
  - Duration: _____ Days/Weeks
  - Scope: (e.g., Critical bug fixes only)
  - Rates for additional work: $_____/hr

Signatures:

- Outgoing Developer/Agency Representative: ____________________
- New Owner/Incoming Representative: ____________________

Conclusion

A successful handover delivers complete control, deep system knowledge, and safe, repeatable operations. It is built on three principles: Complete Ownership Transfer, Comprehensive Knowledge Transfer, and Documented Operational Procedures. The handover is only complete when ownership is fully transferred, all credentials have been rotated, previous access is revoked, and critical operational paths (backups, deployment) have been tested end-to-end.