Lord's Gym Website - Phase 1 Final Steps

Document Version: 1.0
Date: January 27, 2026
Status: Pre-Launch Checklist
Purpose: Complete Phase 1 implementation and prepare for production launch

---

Table of Contents

---

1. Overview

This document outlines the final steps required to complete Phase 1 of the Lord's Gym website project. These steps include domain migration, third-party integrations, user acceptance testing, and formal documentation of all changes.

Objectives

- ✅ Migrate domain from Shopify to CloudFlare for improved performance and control
- ✅ Integrate Go High Level (GHL) for contact forms, calendar, and CRM functionality
- ✅ Conduct comprehensive User Acceptance Testing (UAT) with client
- ✅ Document all changes and configurations for future reference
- ✅ Prepare for production launch

Timeline

Estimated Completion: 2-3 weeks from document approval

Milestones:

- Week 1: Domain migration and Go High Level setup
- Week 2: Integration testing and UAT preparation
- Week 3: UAT execution and documentation finalization

---

2. Domain Migration: Shopify to CloudFlare

2.1 Overview

Migrating the domain from Shopify's DNS management to CloudFlare provides:

- **Improved Performance:** CloudFlare's global CDN and caching
- **Better Control:** Advanced DNS management and security features
- **Cost Savings:** Free SSL certificates and DDoS protection

- **Analytics:** Enhanced DNS and traffic analytics

2.2 Prerequisites

Required Information:

- Current domain registrar account access
- Shopify admin access
- CloudFlare account (create at cloudflare.com if needed)
- DNS records documentation from Shopify

Current Configuration:

- **Shopify Store URL:** `https://lords-gym-auburn.myshopify.com`
- **Domain:** (To be confirmed with client)

2.3 Step-by-Step Migration Process

Step 1: Document Current DNS Configuration

Action Items:

1. Log into Shopify admin panel
2. Navigate to **Settings** > **Domains**
3. Document all DNS records:
    - A records
    - CNAME records
    - MX records (email)
    - TXT records (SPF, DKIM, DMARC)
    - SRV records (if any)

Documentation Template:

```
DNS Record Documentation
Domain: [DOMAIN_NAME]
Date: [DATE]

A Records:
- Name: @, Value: [IP], TTL: [TTL]
- Name: www, Value: [IP], TTL: [TTL]

CNAME Records:
- Name: [NAME], Value: [TARGET], TTL: [TTL]

MX Records:
- Priority: [PRIORITY], Value: [MAIL_SERVER], TTL: [TTL]

TXT Records:
- Name: @, Value: [TEXT], TTL: [TTL]
```

Step 2: Create CloudFlare Account and Add Domain

Action Items:

1. Create CloudFlare account at https://cloudflare.com
2. Click **Add a Site**

3. Enter domain name
4. Select plan (Free plan is sufficient for most needs)
5. CloudFlare will scan existing DNS records

Step 3: Update Nameservers at Domain Registrar

Action Items:

1. Log into domain registrar account (where domain was purchased)
2. Navigate to DNS/Nameserver settings
3. Replace existing nameservers with CloudFlare nameservers:
    - CloudFlare will provide two nameservers (e.g., `ns1.cloudflare.com`, `ns2.cloudflare.com`)
4. Save changes
5. **Wait 24-48 hours** for DNS propagation

Important Notes:

- DNS propagation can take up to 48 hours
- Website may be temporarily unavailable during propagation
- Email may be affected during migration
- Plan migration during low-traffic period

Step 4: Configure DNS Records in CloudFlare

Action Items:

1. In CloudFlare dashboard, navigate to **DNS** > **Records**
2. Add all documented DNS records:
    - **A Records:** Point to Shopify IP addresses
    - **CNAME Records:** Point to Shopify store
    - **MX Records:** Configure email routing
    - **TXT Records:** Add SPF, DKIM, DMARC records

Critical DNS Records for Shopify:

```
Type: A
Name: @
Content: [Shopify IP - typically 23.227.38.65]
Proxy: Enabled (orange cloud)

Type: CNAME
Name: www
Target: shops.myshopify.com
Proxy: Enabled (orange cloud)

Type: CNAME
Name: admin
Target: shops.myshopify.com
Proxy: Disabled (grey cloud)

Type: CNAME
Name: checkout
Target: shops.myshopify.com
Proxy: Disabled (grey cloud)
```

Email Configuration (if using custom email):

```
Type: MX
Name: @
Priority: 10
Mail Server: [EMAIL_PROVIDER_MX]
Proxy: Disabled

Type: TXT
Name: @
Content: v=spf1 include:[EMAIL_PROVIDER] ~all

Type: TXT
Name: _dmarc
Content: v=DMARC1; p=none; rua=mailto:[EMAIL]
```

Step 5: Configure CloudFlare Settings

SSL/TLS Settings:

1. Navigate to **SSL/TLS** > **Overview**
2. Set encryption mode to **Full** (not Full Strict initially)
3. Enable **Always Use HTTPS**
4. Enable **Automatic HTTPS Rewrites**

Performance Settings:

1. Navigate to **Speed** > **Optimization**
2. Enable **Auto Minify** (HTML, CSS, JavaScript)
3. Enable **Brotli** compression
4. Configure **Caching** rules

Security Settings:

1. Navigate to **Security** > **WAF**
2. Enable basic security rules
3. Configure **Rate Limiting** (if needed)
4. Set up **Firewall Rules** (if needed)

Step 6: Update Shopify Domain Settings

Action Items:

1. In Shopify admin, go to **Settings** > **Domains**
2. Verify domain connection
3. Update SSL certificate settings if needed
4. Test domain connection

Step 7: Verification and Testing

Test Checklist:

- [ ] Domain resolves correctly (use `nslookup` or `dig`)
- [ ] Website loads at custom domain
- [ ] HTTPS/SSL certificate is valid
- [ ] Email delivery works (send test email)

- [ ] Shopify storefront loads correctly
- [ ] Checkout process works
- [ ] Admin panel accessible
- [ ] DNS propagation complete (check with multiple DNS servers)

Verification Commands:

```
# Check DNS resolution
nslookup [DOMAIN_NAME]
dig [DOMAIN_NAME]

# Check SSL certificate
openssl s_client -connect [DOMAIN_NAME]:443 -servername [DOMAIN_NAME]

# Check CloudFlare status
curl -I https://[DOMAIN_NAME]
```

Step 8: Update Application Configuration

Action Items:

1. Update environment variables if domain changed
2. Update `constants.ts` if Shopify URL changed
3. Update any hardcoded domain references
4. Update sitemap.xml with new domain
5. Update robots.txt if needed

Files to Update:

- `constants.ts` - Shopify store URL
- `env.example` - Environment variable documentation
- `.github/workflows/pages.yml` - Deployment configuration (if needed)
- Any hardcoded domain references

2.4 Rollback Plan

If Migration Fails:

1. Revert nameservers at domain registrar to original Shopify nameservers
2. Wait 24-48 hours for DNS propagation
3. Document issues encountered
4. Schedule retry after resolving issues

Emergency Contacts:

- Domain Registrar Support
- CloudFlare Support
- Shopify Support

2.5 Post-Migration Tasks

Action Items:

1. Monitor DNS resolution for 48 hours
2. Monitor website performance

3. Monitor email delivery
4. Update documentation with new DNS configuration
5. Train team on CloudFlare dashboard access

---

3. Go High Level Integrations

3.1 Overview

Go High Level (GHL) integration will enable:

- **Contact Form Submissions** → GHL CRM
- **Calendar Bookings** → GHL Calendar
- **Lead Capture** → GHL Pipeline
- **Automated Workflows** → GHL Automation
- **Email Marketing** → GHL Campaigns

3.2 Prerequisites

Required Access:

- Go High Level account with API access
- GHL API credentials (API Key)
- GHL Location ID
- GHL Form/Calendar IDs

Current Status:

- Contact form exists but uses mailto: links
- Calendar system exists but not integrated with GHL
- GHLForm component exists but is deprecated

3.3 Integration 1: Contact Form

Step 1: Create GHL Form

Action Items:

1. Log into Go High Level dashboard
2. Navigate to **Sites** > **Forms**
3. Create new form or use existing form
4. Configure form fields to match ContactForm component:
    - First Name (required)
    - Last Name (required)
    - Email (required)
    - Phone (optional)
    - Inquiry Type (dropdown: Gym Tour, 1on1 Coaching, Membership Question, Outreach/Volunteering, Billing Question)
    - Message (textarea, required)

Form Configuration:

- Form Name: "Lord's Gym Contact Form"
- Form Type: Standard Form
- Submission Action: Create Contact
- Auto-responder: Enable (optional)

Step 2: Get GHL Form Integration Code

Action Items:

1. In GHL form editor, navigate to **Settings** > **Integration**
2. Copy form embed code or API endpoint
3. Note form ID and API endpoint URL
4. Generate API key if needed

API Endpoint Format:

```
POST https://api.gohighlevel.com/v1/forms/{FORM_ID}/submissions
```

Step 3: Update ContactForm Component

File: `components/ContactForm.tsx`

Changes Required:

1. Replace mailto: implementation with GHL API call
2. Add GHL API configuration
3. Add error handling
4. Add success/error states
5. Add loading states

Implementation Example:

```
// Add to constants.ts or env variables
export const GHL_API_KEY = import.meta.env.VITE_GHL_API_KEY || '';
export const GHL_LOCATION_ID = import.meta.env.VITE_GHL_LOCATION_ID || '';
export const GHL_FORM_ID = import.meta.env.VITE_GHL_FORM_ID || '';

// Update ContactForm.tsx handleSubmit function
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setStatus('submitting');

  try {
    const response = await fetch(
      `https://api.gohighlevel.com/v1/forms/${GHL_FORM_ID}/submissions`,
      {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${GHL_API_KEY}`,
        },
        body: JSON.stringify({
          locationId: GHL_LOCATION_ID,
          formId: GHL_FORM_ID,
          firstName: formData.firstName,
          lastName: formData.lastName,
          email: formData.email,
          phone: formData.phone,
          customFields: [
```

```
            {
              fieldId: 'inquiry_type',
              value: formData.inquiryType,
            },
            {
              fieldId: 'message',
              value: formData.message,
            },
          ],
        }),
      }
    );

    if (!response.ok) {
      throw new Error('Form submission failed');
    }

    setStatus('success');
    setFormData({
      firstName: '',
      lastName: '',
      email: '',
      phone: '',
      inquiryType: '',
      message: '',
    });
  } catch (error) {
    console.error('Form submission error:', error);
    setStatus('error');
  }
};
```

Step 4: Configure Environment Variables

Action Items:

1. Add to `.env.local` (development):

```
VITE_GHL_API_KEY=your_ghl_api_key_here
VITE_GHL_LOCATION_ID=your_location_id_here
VITE_GHL_FORM_ID=your_form_id_here
```

2. Add to GitHub Secrets (production):
   - `VITE_GHL_API_KEY`
   - `VITE_GHL_LOCATION_ID`
   - `VITE_GHL_FORM_ID`
3. Update `env.example` :

```
VITE_GHL_API_KEY=your_ghl_api_key_here
VITE_GHL_LOCATION_ID=your_location_id_here
```

```
VITE_GHL_FORM_ID=your_form_id_here
```

Step 5: Test Contact Form Integration

Test Checklist:

- [ ] Form submission creates contact in GHL
- [ ] All form fields map correctly
- [ ] Error handling works
- [ ] Success message displays
- [ ] Auto-responder sends (if configured)
- [ ] Contact appears in GHL CRM
- [ ] Pipeline automation triggers (if configured)

3.4 Integration 2: Calendar & Bookings

Step 1: Configure GHL Calendar

Action Items:

1. In GHL dashboard, navigate to **Calendar**
2. Create calendar or use existing calendar
3. Configure calendar settings:
    - Calendar name: "Lord's Gym Classes"
    - Timezone: Pacific Time (PT)
    - Business hours
    - Buffer times
    - Booking rules

Step 2: Create Calendar Events in GHL

Action Items:

1. Create calendar events matching website calendar:
    - Strength Classes
    - Cardio Classes
    - Recovery Classes
    - Community Events
2. Assign instructors
3. Set capacity limits
4. Configure recurring events

Step 3: Set Up Calendar Sync

Option A: Two-Way Sync (Recommended)

- Sync website calendar → GHL calendar
- Sync GHL calendar → website calendar
- Requires webhook setup

Option B: One-Way Sync (Simpler)

- Website calendar → GHL calendar only
- Bookings on website create appointments in GHL

Step 4: Update Calendar Booking Component

File: `components/CalendarBookingForm.tsx`

Changes Required:

1. Add GHL API integration
2. Create appointment in GHL when booking is made
3. Sync booking status
4. Handle conflicts

Implementation Example:

```
// Add to CalendarBookingForm.tsx
const createGHLAppointment = async (eventId: string, userId: string) => {
  const response = await fetch(
    `https://api.gohighlevel.com/v1/appointments`,
    {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${GHL_API_KEY}`,
      },
      body: JSON.stringify({
        locationId: GHL_LOCATION_ID,
        calendarId: GHL_CALENDAR_ID,
        contactId: userId, // Map to GHL contact ID
        startTime: eventStartTime,
        endTime: eventEndTime,
        title: eventTitle,
        assignedUserId: instructorId,
        notes: eventDescription,
      }),
    }
  );

  return response.json();
};
```

Step 5: Configure Webhooks (Optional)

Action Items:

1. In GHL dashboard, navigate to **Settings** > **Webhooks**
2. Create webhook for:
    - Appointment created
    - Appointment updated
    - Appointment cancelled
3. Configure webhook endpoint (requires backend server or serverless function)
4. Update website calendar when GHL events change

Webhook Endpoint Example:

```
POST https://your-domain.com/api/webhooks/ghl-calendar
```

3.5 Integration 3: Additional Features

Lead Capture Pipeline

Action Items:

1. Configure GHL pipeline stages:
   - New Lead
   - Contacted
   - Qualified
   - Converted
   - Member
2. Set up automation rules:
   - Form submission → Create lead
   - Calendar booking → Move to "Qualified"
   - Membership purchase → Move to "Member"

Email Marketing Integration

Action Items:

1. Create email campaigns in GHL:
   - Welcome email for new contacts
   - Class reminders
   - Membership promotions
   - Outreach updates
2. Set up automation triggers:
   - New contact → Welcome email
   - Class booking → Reminder email
   - Membership inquiry → Follow-up email

SMS Integration (Optional)

Action Items:

1. Configure SMS in GHL
2. Set up SMS automation:
   - Class reminders via SMS
   - Booking confirmations
   - Appointment reminders
3. Get user consent for SMS (required by law)

3.6 Testing Checklist

Contact Form:

- [ ] Form submission creates GHL contact
- [ ] All fields map correctly
- [ ] Auto-responder works
- [ ] Pipeline automation triggers
- [ ] Error handling works

Calendar:

- [ ] Website bookings create GHL appointments
- [ ] Calendar events sync correctly
- [ ] Instructor assignment works
- [ ] Capacity limits enforced
- [ ] Booking conflicts handled

Automation:

- [ ] Lead capture pipeline works
- [ ] Email campaigns send correctly
- [ ] SMS notifications work (if enabled)
- [ ] Workflow triggers function properly

---

4. User Acceptance Testing (UAT)

4.1 Overview

User Acceptance Testing ensures the website meets client requirements and expectations before production launch. This section outlines the UAT process, test cases, and documentation requirements.

4.2 UAT Preparation

Step 1: Create UAT Environment

Action Items:

1. Set up staging environment (if not already exists)
2. Deploy latest code to staging
3. Configure test data
4. Provide client access credentials
5. Create UAT documentation package

Staging Environment:

- URL: (To be configured)
- Test admin credentials provided to client
- Test data populated

Step 2: Prepare UAT Test Plan

Document Components:

- Test scenarios for each feature
- Expected results
- Test data requirements
- Acceptance criteria
- Sign-off forms

4.3 UAT Test Cases

Test Category 1: Public-Facing Website

Test Case 1.1: Home Page

- [ ] Hero section displays correctly
- [ ] Values section shows correct stats
- [ ] Featured products display
- [ ] Navigation menu works
- [ ] Dark mode toggle works
- [ ] Mobile responsive design
- [ ] Images load correctly
- [ ] Call-to-action buttons work

Test Case 1.2: Membership Page

- [ ] All membership tiers display
- [ ] Pricing information correct
- [ ] "Join Now" buttons work
- [ ] Membership FAQ displays
- [ ] Images load correctly
- [ ] Mobile responsive

Test Case 1.3: Shop Page

- [ ] Product catalog displays
- [ ] Category filtering works
- [ ] Product images load
- [ ] Add to cart functionality
- [ ] Shopping cart drawer opens
- [ ] Cart updates correctly
- [ ] Checkout process works

Test Case 1.4: Calendar Page

- [ ] Calendar displays correctly
- [ ] Month/Week/Day/List views work
- [ ] Event details modal opens
- [ ] Search functionality works
- [ ] Calendar export works
- [ ] Booking form displays (if logged in)
- [ ] Recurring events display

Test Case 1.5: Contact Page

- [ ] Contact form displays
- [ ] Form validation works
- [ ] Form submission works (GHL integration)
- [ ] Contact information displays
- [ ] Map loads correctly
- [ ] Operating hours display

Test Case 1.6: Other Pages

- [ ] Training page displays
- [ ] Programs page displays
- [ ] Outreach page displays
- [ ] Community page displays
- [ ] About page displays

Test Category 2: Admin Dashboard

Test Case 2.1: Authentication

- [ ] Login page displays
- [ ] Login with valid credentials works
- [ ] Login with invalid credentials shows error
- [ ] Logout works
- [ ] Session persists correctly
- [ ] Protected routes redirect to login

Test Case 2.2: Dashboard Overview

- [ ] Dashboard loads correctly
- [ ] Metrics display
- [ ] Quick access links work
- [ ] Activity feed displays
- [ ] Navigation works

Test Case 2.3: Content Management

- [ ] Home content editor works
- [ ] Page editor works
- [ ] Rich text editor functions
- [ ] Media library uploads work
- [ ] Image replacement utility works
- [ ] Testimonials management works
- [ ] Version history works
- [ ] Rollback functionality works

Test Case 2.4: Store Management

- [ ] Product CRUD operations work
- [ ] Product categories work
- [ ] Inventory tracking works
- [ ] Featured products work
- [ ] Bulk operations work

Test Case 2.5: Calendar Management

- [ ] Event CRUD operations work
- [ ] Recurring events work
- [ ] Instructor management works
- [ ] Capacity management works

- [ ] Booking oversight works

Test Case 2.6: Settings & SEO

- [ ] Site settings save correctly
- [ ] SEO settings work
- [ ] Schema markup editor works
- [ ] Google Analytics integration works

Test Category 3: Integrations

Test Case 3.1: Go High Level Integration

- [ ] Contact form creates GHL contact
- [ ] Calendar bookings create GHL appointments
- [ ] Pipeline automation works
- [ ] Email campaigns work

Test Case 3.2: Shopify Integration

- [ ] Shopify store URL configured
- [ ] Product links work (if applicable)
- [ ] Storefront integration works

Test Case 3.3: Domain & DNS

- [ ] Domain resolves correctly
- [ ] SSL certificate valid
- [ ] CloudFlare CDN works
- [ ] Performance improved

Test Category 4: Performance & Security

Test Case 4.1: Performance

- [ ] Page load times acceptable (<3 seconds)
- [ ] Images optimized
- [ ] Code splitting works
- [ ] Service worker works
- [ ] Offline functionality works

Test Case 4.2: Security

- [ ] HTTPS enforced
- [ ] Admin routes protected
- [ ] Form validation works
- [ ] XSS protection works
- [ ] CSRF protection works

Test Case 4.3: Accessibility

- [ ] Keyboard navigation works
- [ ] Screen reader compatible

- [ ] Alt text on images
- [ ] Color contrast sufficient
- [ ] Focus indicators visible

4.4 UAT Execution Process

Phase 1: Internal Testing (Development Team)

Duration: 3-5 business days

Action Items:

1. Execute all test cases
2. Document bugs and issues
3. Fix critical issues
4. Re-test fixed issues
5. Prepare UAT package for client

Phase 2: Client UAT

Duration: 5-7 business days

Action Items:

1. Provide client with:
    - UAT test plan document
    - Staging environment access
    - Test credentials
    - Test data
    - UAT feedback form
2. Client executes test cases:
    - Follow test plan
    - Document issues
    - Provide feedback
    - Complete sign-off form
3. Development team:
    - Address client feedback
    - Fix identified issues
    - Re-test fixes
    - Update documentation

Phase 3: Final Review

Duration: 2-3 business days

Action Items:

1. Review all UAT feedback
2. Address remaining issues
3. Final testing
4. Client sign-off
5. Prepare for production launch

4.5 UAT Documentation

UAT Test Plan Document

Sections:

1. Introduction and scope
2. Test environment details
3. Test cases (all categories)
4. Test data requirements
5. Acceptance criteria
6. Sign-off forms

UAT Feedback Form

Fields:

- Test case ID
- Test case name
- Status (Pass/Fail/Blocked)
- Notes/Comments
- Screenshots (if applicable)
- Priority (Critical/High/Medium/Low)
- Tester name
- Date

UAT Sign-Off Form

Sections:

- Test execution summary
- Issues found and resolved
- Client approval
- Sign-off date
- Signatures (Client, Project Manager, Lead Developer)

4.6 UAT Issue Tracking

Issue Categories:

- **Critical:** Blocks launch, must be fixed
- **High:** Major functionality issue, should be fixed
- **Medium:** Minor issue, can be fixed post-launch
- **Low:** Cosmetic issue, enhancement request

Issue Resolution Process:

1. Log issue in tracking system
2. Assign priority
3. Assign to developer
4. Fix issue
5. Re-test
6. Close issue
7. Update UAT documentation

---

5. Change Management & Documentation

5.1 Overview

All changes made during Phase 1 final steps must be formally documented for future reference, maintenance, and handover purposes.

5.2 Documentation Requirements

Technical Documentation

1. Domain Migration Documentation

- DNS configuration before migration
- DNS configuration after migration
- CloudFlare settings
- SSL certificate details
- Rollback procedures

2. Integration Documentation

- Go High Level API configuration
- API endpoints used
- Authentication methods
- Webhook configurations
- Error handling procedures

3. Environment Configuration

- Environment variables list
- Configuration files
- API keys management
- Secrets management

User Documentation

1. Admin User Guide

- How to use admin dashboard
- Content management procedures
- Calendar management
- Store management
- Settings configuration

2. Integration User Guide

- How to use Go High Level integrations
- Contact form management
- Calendar sync procedures
- Pipeline management

5.3 Change Log

Document all changes:

- Date of change
- Change description

- Files modified
- Configuration changes
- Testing performed
- Approval status

Change Log Template:

```
Change Log Entry
Date: [DATE]
Change ID: [ID]
Description: [DESCRIPTION]
Files Modified: [FILES]
Configuration Changes: [CONFIG]
Testing: [TESTING]
Approved By: [NAME]
Status: [STATUS]
```

5.4 Configuration Management

Environment Variables

Document all environment variables:

```
# Supabase Configuration
VITE_SUPABASE_URL=
VITE_SUPABASE_ANON_KEY=

# Go High Level Configuration
VITE_GHL_API_KEY=
VITE_GHL_LOCATION_ID=
VITE_GHL_FORM_ID=
VITE_GHL_CALENDAR_ID=

# Shopify Configuration
VITE_SHOPIFY_STORE_URL=

# Mindbody Configuration
VITE_MINDBODY_SITE_ID=

# Domain Configuration
VITE_DOMAIN=
```

Configuration Files

Files to document:

- `vite.config.ts` - Build configuration
- `.github/workflows/pages.yml` - Deployment configuration
- `constants.ts` - Application constants
- `supabase/config.toml` - Supabase configuration
- CloudFlare settings
- DNS records

5.5 Handover Documentation

Prepare for client handover:

1. **Access Credentials:**
    - Admin dashboard access
    - CloudFlare account access
    - Go High Level account access
    - Domain registrar access
    - GitHub repository access
2. **Support Contacts:**
    - Development team contacts
    - CloudFlare support
    - Go High Level support
    - Domain registrar support
3. **Maintenance Procedures:**
    - Regular maintenance tasks
    - Update procedures
    - Backup procedures
    - Troubleshooting guides

---

6. Pre-Launch Checklist

6.1 Domain & DNS

- [ ] Domain migrated to CloudFlare
- [ ] DNS records configured correctly
- [ ] SSL certificate active and valid
- [ ] CloudFlare settings optimized
- [ ] DNS propagation complete
- [ ] Domain resolves correctly globally

6.2 Integrations

- [ ] Go High Level contact form integrated
- [ ] Go High Level calendar integrated
- [ ] Go High Level pipeline configured
- [ ] Shopify integration configured
- [ ] All API keys configured
- [ ] Webhooks configured (if applicable)
- [ ] Integration testing complete

6.3 Testing

- [ ] Internal testing complete
- [ ] Client UAT complete
- [ ] All critical issues resolved

- [ ] Performance testing complete
- [ ] Security testing complete
- [ ] Accessibility testing complete
- [ ] Cross-browser testing complete
- [ ] Mobile device testing complete

6.4 Documentation

- [ ] Technical documentation complete
- [ ] User documentation complete
- [ ] Change log updated
- [ ] Configuration documented
- [ ] Handover documentation prepared

6.5 Production Readiness

- [ ] Environment variables configured
- [ ] Production build tested
- [ ] Deployment pipeline tested
- [ ] Monitoring configured
- [ ] Backup procedures in place
- [ ] Rollback plan documented
- [ ] Support contacts documented

---

7. Post-Launch Verification

7.1 Immediate Verification (Day 1)

Checklist:

- [ ] Website accessible at production domain
- [ ] All pages load correctly
- [ ] Forms submit successfully
- [ ] Calendar displays correctly
- [ ] Admin dashboard accessible
- [ ] No console errors
- [ ] SSL certificate valid
- [ ] Performance acceptable

7.2 Week 1 Monitoring

Monitor:

- Website uptime
- Form submission success rate
- Calendar booking success rate
- Error rates
- Performance metrics
- User feedback

7.3 Week 2-4 Monitoring

Monitor:

- User adoption
- Feature usage
- Integration performance
- Support requests
- Bug reports
- Performance trends

7.4 Post-Launch Support

Support Plan:

- **Week 1:** Daily check-ins
- **Week 2-4:** Weekly check-ins
- **Month 2+:** Monthly reviews
- **Ongoing:** Issue tracking and resolution

---

8. Appendices

Appendix A: CloudFlare DNS Records Template

```
# A Records
@                   A       23.227.38.65        Proxied
www                 CNAME   shops.myshopify.com Proxied

# CNAME Records
admin               CNAME   shops.myshopify.com DNS only
checkout            CNAME   shops.myshopify.com DNS only

# MX Records (if using custom email)
@                   MX      10 mail.example.com

# TXT Records
@                   TXT     v=spf1 include:example.com ~all
_dmarc              TXT     v=DMARC1; p=none; rua=mailto:admin@example.com
```

Appendix B: Go High Level API Endpoints

```
# Contact Form Submission
POST https://api.gohighlevel.com/v1/forms/{FORM_ID}/submissions

# Create Appointment
POST https://api.gohighlevel.com/v1/appointments

# Get Contacts
GET https://api.gohighlevel.com/v1/contacts
```

```
# Update Contact
PUT https://api.gohighlevel.com/v1/contacts/{CONTACT_ID}
```

Appendix C: UAT Test Plan Template

See separate UAT Test Plan document for detailed test cases.

Appendix D: Change Log Template

```
Change Log Entry
================
Date: [DATE]
Change ID: [ID]
Type: [Domain Migration/Integration/Bug Fix/Enhancement]
Description: [DETAILED DESCRIPTION]
Files Modified: [LIST OF FILES]
Configuration Changes: [CONFIGURATION DETAILS]
Testing: [TESTING PERFORMED]
Approved By: [NAME, TITLE]
Status: [Pending/Approved/Completed]
```

Appendix E: Contact Information

Development Team:

- [To be filled]

CloudFlare Support:

- Website: https://support.cloudflare.com
- Community: https://community.cloudflare.com

Go High Level Support:

- Website: https://support.gohighlevel.com
- Email: [To be filled]

Domain Registrar:

- [To be filled]

Document Approval

Prepared By: Development Team
Date: January 27, 2026
Version: 1.0

Approved By:

- ☐ Client Representative: _____ Date: _____
- ☐ Project Manager: _____ Date: _____
- ☐ Lead Developer: _____ Date: _____

Next Steps

1. **Review this document** with all stakeholders
2. **Approve timeline** and resource allocation
3. **Begin domain migration** process
4. **Set up Go High Level** integrations
5. **Prepare UAT** test plan and environment
6. **Execute UAT** with client
7. **Document all changes** as they occur
8. **Complete pre-launch** checklist
9. **Launch to production**
10. **Monitor and support** post-launch

End of Document