UAGC Digital Experience (DX) Documentation

Version 1.0

None

None

Table of contents

1. UAGC Digital Experience (DX) Documentation 4
   - 1.1 Quick Links 4
   - 1.2 Documentation Overview 4
   - 1.3 Team Members & Responsibilities 5
   - 1.4 Recent Updates 5
   - 1.5 Feedback & Contributions 5
   - 1.6 Why This Documentation Exists 6
   - 1.7 Quick Reference 7
   - 1.8 Documentation by Category 7
   - 1.9 Accessing This Documentation 8
   - 1.10 Contributing 8
   - 1.11 Growth Roadmap 8
2. Overview 9
   - 2.1 Why This Documentation Exists 9
   - 2.2 Who Does What — Roles, Responsibilities & Contacts 10
   - 2.3 Day-to-Day Operations 12
3. Tools & Processes 15
   - 3.1 Asana 15
   - 3.2 "Request Information" Form (Quick-Guide) 17
4. Documentation Guides 19
   - 4.1 Getting Started with Documentation 19
   - 4.2 Add, Remove, or Redirect Pages 21
   - 4.3 Optimizely Tests 23
   - 4.4 SEO Hygiene 25
   - 4.5 QA Smoke Test 28
5. Advanced Guides 30
   - 5.1 Accessibility & Inclusive Design 30
   - 5.2 Drupal Coding Standards & Snippet Library 35
   - 5.3 SEO Redirect Decision Tree 40
   - 5.4 Performance & Core Web Vitals Playbook 44
   - 5.5 Privacy, Consent & Cookie Governance 49
   - 5.6 Release Checklist & Incident Rollback Procedure 55
6. Reference 62
   - 6.1 Analytics Standards 62
   - 6.2 Glossary of Internal Acronyms & Naming Conventions 64

6.3 Documentation Workflow 72
6.4 Growth Roadmap 75
6.5 Site Map 77

1. UAGC Digital Experience (DX) Documentation

Welcome to the central documentation hub for the UAGC Digital Experience team. This site contains everything you need to maintain, update, and improve the uagc.edu website.

Need something specific?

Use the search feature (press /) or browse the navigation menu to find what you need.

1.1 Quick Links

- Getting Started Guide - New to the team? Start here!
- Page Changes - Learn how to add, remove, or redirect pages
- QA Smoke Test - Verify changes before they go live
- Asana Workflow - How we track and manage tasks
- SEO Hygiene - Best practices for search optimization

1.2 Documentation Overview

Our documentation is organized by:

- :material-information-outline: **Overview** Learn about our team, roles, and day-to-day operations [:octicons-arrow-right-24: Why This Exists](why-this-exists.md) - :material-tools: **Tools & Processes** Task management and form handling processes [:octicons-arrow-right-24: Asana Workflow](asana.md) - :material-book-open-page-variant: **Documentation Guides** Core guides for common tasks [:octicons-arrow-right-24: View Guides](guides/getting-started.md) - :material-code-tags: **Advanced Guides** Technical documentation for engineering and SEO [:octicons-arrow-right-24: View Advanced Guides](guides/accessibility.md)

1.3 Team Members & Responsibilities

Person	Title	Core Responsibilities
Thomas	DX Director / Product Owner	Roadmap, priorities, release approval
Brandy	Digital Marketing & Web Operations Manager	DX governance, CMS permissions, training
Jason	Senior Backend Drupal Engineer	Drupal architecture, custom modules, security
Will	Backend Engineer	API endpoints, CI jobs, patch pipeline
Brian	Front-End Dev & QA Lead	Component library, accessibility, QA test plans
Anthony	Front-End Developer & Experiment Engineer	Optimizely tests, dataLayer, QA
Omar	SEO & Tracking Manager	Technical/content SEO, GA4/GTM, BigQuery pipelines

1.4 Recent Updates

- **SEO Redirect Decision Tree** - Updated with new canonical link guidelines
- **Performance & Core Web Vitals** - Added mobile optimization section
- **Drupal Coding Standards** - Updated for latest Drupal version

1.5 Feedback & Contributions

This documentation is continuously improving. If you have suggestions:

1. Create an Asana task using the **Documentation** template
2. Specify what needs to be updated
3. Add any supporting information or examples

> **Documentation Day**
>
> We hold a quarterly "Docs Day" where each team member reviews and updates their documentation sections.
>
> Next scheduled Docs Day: **June 15, 2025**

1.5.1 Getting Started

Start here if you're new to the DX documentation.

Getting Started Guide →

1.5.2 Daily Operations

Core processes for day-to-day website management.

Daily Operations Guide →

1.5.3 Common Tasks

Step-by-step guides for frequent actions.

Add/Remove Pages →

QA Testing →

1.5.4 Reference

Technical specifications and standards.

Glossary →

Site Map →

1.6 Why This Documentation Exists

- **Purpose**: Central home for all repeatable tasks: content edits, page launches, experiments, SEO, tracking, and development standards
- **Audience**: Jason, Will, Brian, Anthony, Omar, Brandy, Thomas + anyone we onboard next
- **Success Metrics**: ↘ time-to-answer, ↘ QA slips, ↘ Slack back-and-forth; ↗ first-call resolution

1.7 Quick Reference

Task	Go To	Owner
Create Asana Task	Asana	Brandy
Update Content	Page Changes	Brian
Set Up A/B Test	Optimizely Tests	Anthony
Fix SEO Issue	SEO Hygiene	Omar
Run QA Check	QA Smoke Test	Brian

1.8 Documentation by Category

1.8.1 Content Management

Add/Remove/Redirect Pages

SEO Hygiene

SEO Redirect Decision Tree

1.8.2 Development & QA

Drupal Coding Standards

QA Smoke Test

Accessibility Guidelines

Release & Incident Procedures

1.8.3 Analytics & Optimization

Analytics Standards

Optimizely Tests

Performance Web Vitals

Privacy & Consent

1.8.4 Team Resources

Who Does What

Asana

RFI Form

Documentation Workflow

Growth Roadmap

1.9 Accessing This Documentation

You can access this documentation in two ways:

1. **Locally**: Run `mkdocs serve` to view at http://localhost:8000
2. **Online**: Visit our hosted documentation at docs.dx.uagc.edu

1.9.1 Keyboard Shortcuts

Navigate faster with these keyboard shortcuts:

- `/` - Open search
- `Home` - Go to top of page
- `End` - Go to bottom of page
- `Tab` then `Enter` - Navigate through links

1.10 Contributing

Learn how to contribute to this documentation in the Documentation Workflow guide.

1.11 Growth Roadmap

Want to know where we're headed? Check out our Growth Roadmap for documentation development.

Success Looks Like: ↘ time-to-answer, ↘ QA slips, ↘ Slack back-and-forth; ↗ first-call resolution.

April 28, 2025

April 28, 2025

2. Overview

2.1 Why This Documentation Exists

2.1.1 Purpose

This documentation server exists as the **central home for every repeatable task** that keeps uagc.edu running, including:

- Content edits
- Page launches
- Experiments
- SEO management
- Tracking implementation
- Development standards

2.1.2 Audience

This documentation is specifically designed for:

- Jason, Will, Brian, Anthony, Omar, Brandy, Thomas
- Any new team members who join our DX crew

2.1.3 Success Metrics

We'll know this documentation is successful when we see:

- ↘ **Time to answer** questions about processes and procedures
- ↘ **QA slips** due to missed steps or requirements
- ↘ **Slack back-and-forth** for routine inquiries
- ↗ **First-call resolution** of issues and questions

Our goal is to create a self-service hub that helps our team work more efficiently while maintaining quality standards.

April 28, 2025

April 28, 2025

2.2 Who Does What — Roles, Responsibilities & Contacts

This page outlines the key roles and responsibilities within our DX team, making it clear who to contact for different types of tasks and questions.

2.2.1 Leadership & Governance

Person	Title	Core Responsibilities
Thomas	DX Director / Product Owner	Sets roadmap & priorities; leads weekly DX stand-ups; final release approver; escalation point
Brandy	Digital Marketing & Web Operations Manager	Owns DX governance; manages Asana board hygiene; CMS permissions & training, implement roadmap guides

2.2.2 Drupal Engineering

Person	Title	Core Responsibilities
Jason	Senior Backend Drupal Engineer	Drupal architecture & custom modules; configuration management & env parity; performance/security remediation
Will	Backend Engineer	Implements API endpoints & unit tests; maintains CI jobs; hotfix/patch pipeline; regression-test liaison

2.2.3 Front-End, QA & Experimentation

Person	Title	Core Responsibilities
Brian	Front-End Dev & QA Lead	Component library & global styles; WCAG 2.2 AA accessibility; QA test plans; visual regression & device farm
Anthony	Front-End Developer & Experiment Engineer	Builds Optimizely tests & templates; maintains dataLayer pushes; coordinates pre-launch QA & analytics

2.2.4 SEO & Analytics

Person	Title	Core Responsibilities
Omar	SEO & Tracking Manager	Technical & content SEO; keyword & URL taxonomy; GA4 & GTM governance; BigQuery pipelines; event tag validation, RFI to Lead API documentation, overall strategy, lead mapping

2.2.5 Contact Protocols

When you have a question or need assistance:

1. Check this documentation first
2. Contact the appropriate team member based on the responsibility area
3. For escalations, reach out to Thomas (DX Director)

April 28, 2025

April 28, 2025

2.3 Day-to-Day Operations

This page provides a high-level overview of routine operations for the DX team.

2.3.1 Content Edits

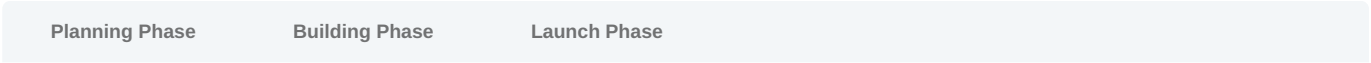
Process for standard content updates:

1. DX Team member updates content via Drupal admin
2. Brian performs QA on the changes
3. Changes are published after verification

Key Contact: Brian (QA Lead)

2.3.2 Experiments

Workflow for creating and deploying experiments:



1. Define test hypothesis
2. Determine success metrics
3. Identify target pages and audience
4. Create Asana ticket using experiment template

1. Anthony creates test in Optimizely
2. Omar sets up tracking events
3. Brian performs QA on test implementation
4. Fix any identified issues

1. Get approval from Thomas
2. Activate experiment
3. Monitor performance
4. Document results in changelog

Key Contacts: Anthony (Implementation), Omar (Tracking)

2.3.3 SEO Sweeps

Monthly SEO maintenance process:

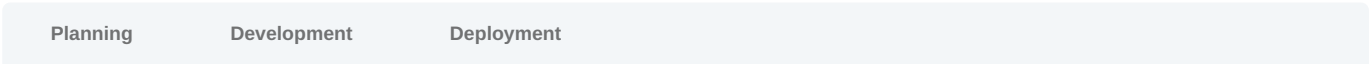
1. Omar runs a monthly audit
2. Tasks for fixes/redirects are logged in Asana
3. Team addresses issues based on priority

SEO audit includes checking: - List of redirects - Schema markup (JSON) errors - Broken links

Key Contact: Omar (SEO & Tracking Manager)

2.3.4 Releases

Release schedule and process:



- Identify content/features for release
- Create release schedule
- Assign priorities to items
- Document scope in Asana

- Code changes implemented
- Features built and tested
- Release notes drafted
- QA performed in dev/staging

- Release cutoff is Monday of deployment week
- Jason handles backend deployment
- Brian verifies frontend
- Thomas signs off on all releases

Key Contacts: Jason (Backend), Thomas (Approval)

2.3.5 Common Support Tasks

Frequently performed support activities:

Content Updates

- Text changes
- Image updates
- Link corrections

Process: Submit via Asana → Brian QA → Publish

Form Adjustments

- Field modifications
- Validation updates
- Tracking changes

Process: Submit via Asana → Anthony implements → Omar verifies tracking → Publish

SEO Fixes

- Meta tag updates
- Redirect implementation
- Schema.org markup fixes

Process: Omar identifies → Task created → Developer implements → Omar verifies

April 28, 2025

April 28, 2025

3. Tools & Processes

3.1 Asana

This guide explains how to use Asana for managing DX tasks and projects.

3.1.1 Choosing a Template

We use specific templates in Asana to standardize our workflow. Select the appropriate template when creating a new task:

- **Page Update** - For changes to existing pages
- **Delete/Redirect** - For removing pages or setting up redirects
- **New Feature** - For adding new functionality
- **Bug** - For fixing issues
- **Documentation** - For updating this documentation

3.1.2 Filling Out the Basics

Every task should include these essential fields:

Field	Description
Title	Clear description of what needs to be done
URL(s)	Affected page(s)
What-success-looks-like	Expected outcome
Priority	Urgency level
Due date	When it needs to be completed

3.1.3 Auto-assignment

Asana rules automatically: - Drop the task into the right board column - Ping Thomas/Brandy for triage

3.1.4 Task Flow

Tasks move through the following stages:

1. **Backlog** - Tasks waiting to be worked on
2. **In Progress** - Currently being worked on
3. **Peer Review** - Ready for team review
4. **Staging QA** - Ready for quality assurance
5. **Done** - Task complete and ready for deployment

When a task is deployed, a deploy note is posted in the #groupchat channel.

3.1.5 Asana Best Practices

- Keep task descriptions concise but complete
- Include all relevant URLs and assets
- Tag appropriate team members
- Update status regularly

April 28, 2025

April 28, 2025

3.2 "Request Information" Form (Quick-Guide)

This guide provides an overview of how the "Request Information" (RFI) forms operate and how data passes to the LEAD_API.

3.2.1 Why It Matters

The RFI form is the **biggest source of inquiries** for UAGC, directly contributing to: - Lead generation - Enrollments - Revenue

3.2.2 Key Owners

- **Anthony**: Front-end implementation
- **Omar**: Tracking and SEO

3.2.3 Change Request Checklist

To modify a Request Information form:

1. Log an Asana "RF Change" task
2. Describe the specific tweak needed:
3. Label changes
4. Field additions/removals
5. Validation rules
6. Tracking modifications

3.2.4 Implementation Process

The standard workflow for RFI form changes:

1. Anthony creates a prototype of the form changes
2. Brian performs QA on the implementation
3. Omar verifies the data layer is capturing correct information
4. Brandy schedules the publish date
5. Post-launch: Verify leads and GA events in dashboard

3.2.5 Common Form Elements

Standard components in our RFI forms:

- Name fields (First/Last)
- Email field
- Phone field
- Program interest selection
- Military affiliation options
- Privacy policy acceptance
- Submit button

3.2.6 Data Flow

1. User submits form
2. Form data is validated client-side
3. Data is sent to LEAD_API
4. API processes the information
5. Lead is created in CRM
6. Thank you message is displayed to user
7. GA event is triggered

3.2.7 Deep-Dive Documentation

For detailed technical specifications and API documentation, refer to the RFI Technical Documentation in the "Dev Reference" folder.

April 28, 2025

April 28, 2025

4. Documentation Guides

4.1 Getting Started with Documentation

This guide explains how to access, use, and contribute to the DX documentation platform.

> **Before You Begin**
>
> Make sure you have access to Asana and the DX documentation repository before proceeding. Contact Brandy if you need access.

4.1.1 Accessing the Documentation

1. Navigate to the documentation URL: UAGC DX Documentation
2. Log in with your UAGC credentials (if required)
3. Use the navigation menu to find the documentation you need

> **Pro Tip**
>
> Use the search feature (`/` key) to quickly find what you're looking for!

4.1.2 Using Templates

When creating new documentation:

1. Review existing documentation for style and structure
2. Choose the appropriate template for your documentation type
3. Follow the template structure to ensure consistency

> **Example Template Usage**

```
# Document Title

## Purpose
Explain what this document is for.

## Process
1. Step one
2. Step two
3. Step three

## Related Resources
- [Link to related document]
```

4.1.3 Opening a Documentation Request

When documentation needs to be created or updated:

1. Log in to Asana
2. Create a new task using the **Documentation** template
3. Fill in the required information:
4. Title: Clear description of the documentation needed
5. Priority
6. Documentation area/section
7. Due date
8. Assign the task to the appropriate owner
9. Provide any additional context or requirements

> **Important**
>
> All documentation changes should be tracked in Asana to maintain version control and accountability.

4.1.4 Documentation Best Practices

- **Be concise**: Focus on clear instructions rather than lengthy explanations
- **Use visuals**: Include screenshots or diagrams when helpful
- **Link related docs**: Connect related pieces of documentation
- **Keep it updated**: Review and update docs as processes change
- **Standardize format**: Follow established formatting guidelines

> **Quarterly Reviews**
>
> Remember that each section owner should review their documentation quarterly as part of our "Docs Day" initiative.

4.1.5 Next Steps

- Learn how to add, remove, or redirect pages
- Explore QA processes
- Understand SEO hygiene

April 28, 2025

April 28, 2025

4.2 Add, Remove, or Redirect Pages

This guide provides a step-by-step process for adding new pages, removing existing pages, or setting up redirects on the UAGC website.

4.2.1 Adding a New Page

1. **Step 1**: Create Asana task using "Page Update" template
2. **Step 2**: Gather required content
3. **Step 3**: Build page in Drupal
4. **Step 4**: QA process
5. **Step 5**: Publish page

4.2.2 Removing a Page

1. **Step 1**: Create Asana task using "Delete/Redirect" template
2. **Step 2**: Verify there are no inbound links
3. **Step 3**: Determine if a redirect is needed
4. **Step 4**: Implement deletion
5. **Step 5**: Verify removal

4.2.3 Setting Up Redirects

1. **Step 1**: Create Asana task using "Delete/Redirect" template
2. **Step 2**: Identify source and destination URLs
3. **Step 3**: Determine redirect type (301 permanent vs 302 temporary)
4. **Step 4**: Implement redirect
5. **Step 5**: Test redirect functionality

4.2.4 Best Practices

- Always check analytics before removing pages
- Document all redirects for future reference
- Test all links after changes are published
- Update sitemaps after major page changes

4.2.5 Related Resources

- SEO Hygiene Guide
- QA Smoke Test

April 28, 2025

April 28, 2025

4.3 Optimizely Tests

This guide explains the process for creating and managing A/B tests using Optimizely.

4.3.1 Test Creation Process

The complete workflow from strategy to results:

1. **Strategy Brief** - Document test hypothesis and goals
2. **Asana Ticket** - Create task using experiment template
3. **Build** - Anthony creates the test variation
4. **Track** - Omar sets up event tracking
5. **QA** - Brian verifies functionality across devices
6. **Launch** - Start the experiment after approval

4.3.2 Creating a Test Brief

Every experiment should start with a clear brief that includes:

- **Hypothesis** - What we believe will improve performance
- **Success Metrics** - How we'll measure improvement
- **Target Pages** - Where the experiment will run
- **Audience** - Who will see the experiment
- **Variations** - What changes we're testing

4.3.3 Implementation Guidelines

When building Optimizely tests:

- Use clean, reusable code
- Follow responsive design principles
- Keep variations focused on testing specific elements
- Maintain brand guidelines in all variations
- Consider load speed impact

4.3.4 QA Checklist

Before launching any test:

- Verify variation appears correctly on all target pages
- Test on multiple devices and browsers
- Confirm tracking events fire correctly
- Check for any JavaScript errors or conflicts
- Validate content for typos and accuracy

4.3.5 Analyzing Results

After a test has run:

- Wait for statistical significance
- Document results in the experiment log
- Present findings to the team
- Implement winning variations as permanent changes
- Use insights to inform future test ideas

4.3.6 Reference Materials

- Optimizely Developer Documentation
- Analytics Tagging Standards

April 28, 2025

April 28, 2025

4.4 SEO Hygiene

This guide outlines the essential SEO practices and maintenance tasks for keeping the UAGC website optimized for search engines.

4.4.1 On-Page SEO Best Practices

Ensure these elements are properly optimized on every page:

- **Page Titles**
- Include primary keyword at the beginning when possible
- Keep each title unique across the website
- Limit to under 60 characters to avoid truncation in search results
- Use Drupal tokens for consistency: `[node:title] | [site:name]`
- For program pages, follow format: `[Program Name] Degree | UAGC`
- Avoid keyword stuffing or excessive punctuation
- **Meta Descriptions** - Compelling summary, include keywords, under 160 characters
- **H1 Headings** - One per page, include primary keyword
- **URL Structure** - Readable, keyword-rich, avoid parameters
- **Content Quality** - Relevant, comprehensive, well-structured
- **Internal Linking** - Strategic links to related content
- **Image Optimization** - Descriptive filenames, alt text, appropriate file size

Page Title Configuration in Drupal

To set up proper page titles in Drupal:

1. Navigate to **Configuration > Search and metadata > Metatag**
2. Select the appropriate content type
3. Under "Page title," configure the token pattern: `[node:title] | [site:name]`
4. For custom landing pages, you may specify: `[node:field_custom_title] | [site:name]`

Example Page Titles

- Homepage: `UAGC | University of Arizona Global Campus`
- Program page: `Bachelor of Science in Computer Science | UAGC`
- Blog post: `5 Tips for Student Success | UAGC`

Important

Always review how titles display in search results using Google Search Console. Titles that are too long will be truncated, potentially cutting off important information.

4.4.2 Technical SEO

Regular checks for these technical elements:

- **Schema Markup** - Verify JSON-LD implementation
- **XML Sitemaps** - Keep updated, submit in Search Console
- **Robots.txt** - Ensure proper configuration
- **Canonicalization** - Check for correct canonical tags
- **Mobile Responsiveness** - Test on multiple devices
- **Page Speed** - Monitor Core Web Vitals
- **SSL Certificate** - Verify security implementation

4.4.3 Monthly SEO Audit Process

Omar conducts a monthly audit that includes:

1. Crawling the site for technical issues
2. Identifying broken links and 404 errors
3. Checking redirect implementations
4. Verifying schema markup for errors
5. Analyzing search performance in Google Search Console
6. Reviewing keyword rankings
7. Creating Asana tasks for needed fixes

4.4.4 Redirect Management

Guidelines for implementing redirects:

- Use 301 (permanent) redirects for content that has moved permanently
- Use 302 (temporary) redirects for temporary changes
- Maintain a central redirect log
- Avoid redirect chains (redirects pointing to redirects)
- Regularly audit redirects for relevance

4.4.5 SEO Tools

Key tools used in our SEO workflow:

- Google Search Console
- Google Analytics
- Screaming Frog
- SEMrush
- Ahrefs

4.4.6 Contact

For SEO-related questions and issues, contact Omar (SEO & Tracking Manager).

April 28, 2025

April 28, 2025

4.5 QA Smoke Test

This guide provides a quick checklist for performing basic quality assurance testing on new or updated pages.

4.5.1 Pre-Launch Checklist

Before any page goes live, verify these essential elements:

Functionality

- [ ] All links work correctly
- [ ] Forms submit properly
- [ ] Required form validation works
- [ ] Buttons and CTAs function as expected
- [ ] Interactive elements (dropdowns, tabs, etc.) work

Visual Quality

- [ ] Content displays correctly on desktop, tablet, and mobile
- [ ] Images load properly and aren't distorted
- [ ] Fonts are consistent with brand guidelines
- [ ] Spacing and alignment follow design standards
- [ ] No overlapping elements on any screen size

Technical Health

- [ ] No JavaScript errors in console
- [ ] No 404 resources
- [ ] Page load time is acceptable
- [ ] Proper metadata is in place
- [ ] Analytics tracking is working

4.5.2 Device Testing

Test pages on these key device types:

Device Type	Examples
Desktop	Windows PC, Mac
Tablet	iPad, Samsung Galaxy Tab
Mobile	iPhone, Android phones

For critical pages, use our device farm to test on a wider range of devices.

4.5.3 Browser Compatibility

Verify functionality in these browsers:

- Chrome (latest)
- Firefox (latest)
- Safari (latest)
- Edge (latest)

4.5.4 Accessibility Checks

Verify these basic accessibility elements:

- [ ] Proper heading structure (H1, H2, etc.)
- [ ] All images have alt text
- [ ] Sufficient color contrast
- [ ] Keyboard navigation works
- [ ] Form fields have labels

For deeper accessibility testing, use the WCAG 2.2 AA Quick Reference.

4.5.5 Bug Reporting

When reporting bugs:

1. Create an Asana task using the "Bug" template
2. Include the URL where the issue occurs
3. Add screenshots or screen recordings
4. Describe expected vs. actual behavior
5. Note browser/device information

4.5.6 Contact

For QA-related questions, contact Brian (Front-End Dev & QA Lead).

April 28, 2025

April 28, 2025

5. Advanced Guides

Owner: Brian (Front-End & QA Lead) **Last Updated:** 2023-11-15 **Difficulty:** Intermediate

This guide ensures UAGC web properties meet WCAG 2.2 AA accessibility standards, helping create inclusive digital experiences for all users regardless of ability.

5.1 Accessibility & Inclusive Design

This guide provides best practices for ensuring our digital experiences meet WCAG 2.2 AA accessibility standards.

Why Accessibility Matters

Accessibility isn't just a compliance requirement—it ensures our content and services are available to everyone, regardless of ability or disability.

5.1.1 WCAG 2.2 AA Quick Reference

The Web Content Accessibility Guidelines (WCAG) 2.2 AA standards are organized around four principles:

Perceivable	Operable	Understandable	Robust

Content must be presentable in ways all users can perceive.

- Provide text alternatives for non-text content
- Provide captions and alternatives for multimedia
- Create content that can be presented in different ways
- Make it easier for users to see and hear content

User interface components must be operable by all users.

- Make all functionality available from keyboard
- Give users enough time to read and use content
- Do not use content that causes seizures
- Provide ways to help users navigate and find content

Information and operation must be understandable.

- Make text readable and understandable
- Make content appear and operate in predictable ways
- Help users avoid and correct mistakes

Content must be robust enough to work with current and future technologies.

- Maximize compatibility with current and future tools

5.1.2 Website Brand Guidelines

Our digital brand must maintain accessibility while staying true to our visual identity:

1. **Color Usage**
2. Always maintain a 4.5:1 contrast ratio for text
3. Don't rely solely on color to convey meaning
4. Use our accessible color palette from the design system
5. **Typography**
6. Use our brand fonts (Roboto)
7. Minimum text size: 16px for body text
8. Maintain proper heading hierarchy (H1 → H6)
9. **Imagery**
10. All images require meaningful alt text
11. Decorative images should have empty alt attributes (alt="")
12. Avoid text in images; if necessary, duplicate text in the alt attribute

5.1.3 Website Design Blocks

When using our design system components:

Button Components

- Must be keyboard accessible
- Need clear focus states
- Require sufficient color contrast
- Should have descriptive text

Card Components

- Must have proper heading structure
- Need keyboard-accessible interactive elements
- Should maintain spacing for readability
- Require sufficient color contrast

Navigation Components

- Must be keyboard navigable
- Need clear focus indicators
- Should have proper ARIA attributes
- Must be usable at different screen sizes

5.1.4 Form Blocks

Forms require special attention for accessibility:

1. **Labels**
2. Every input must have a properly associated label
3. Use explicit labels with the `for` attribute
4. Don't rely on placeholder text as the only label
5. **Error Handling**
6. Error messages must be clearly associated with inputs
7. Use both color and text/icons to indicate errors
8. Provide clear instructions for correction
9. **Keyboard Accessibility**
10. Tab order must be logical
11. Form can be completed using keyboard alone
12. Custom controls require keyboard event handlers
13. **Screen Reader Support**
14. Use appropriate ARIA attributes when needed
15. Test with screen readers
16. Complex components need additional ARIA roles/states

5.1.5 Testing For Accessibility

Before deploying any new feature:

1. **Automated Testing**
2. Run axe DevTools or Lighthouse accessibility audit
3. Fix all critical and serious issues
4. **Keyboard Testing**
5. Navigate using Tab, Shift+Tab, Enter, Space, Arrow keys
6. Ensure all interactive elements can be accessed and activated
7. **Screen Reader Testing**
8. Test with VoiceOver (Mac) or NVDA (Windows)
9. Verify all content is announced properly
10. Check form inputs, buttons, and navigation
11. **Visual Testing**
12. Test at 200% zoom
13. Check color contrast with WebAIM Contrast Checker
14. Verify the site works with high contrast mode

5.1.6 Resources

- WebAIM Contrast Checker

- WCAG 2.2 Quick Reference
- Accessible Rich Internet Applications (ARIA)
- The A11Y Project

April 28, 2025

April 28, 2025

Related Resources

- QA Smoke Test
- Performance & Core Web Vitals
- Drupal Coding Standards

This guide was last updated on 2023-11-15. If you notice any issues, please let us know.

5.2 Drupal Coding Standards & Snippet Library

This guide provides coding standards, best practices, and reusable code snippets for Drupal development at UAGC.

> **Important**
>
> Following these standards is mandatory for all Drupal development to ensure code quality, maintainability, and security.

5.2.1 Coding Standards

Our Drupal development follows the official Drupal Coding Standards with some additional team-specific guidelines.

PHP Standards

Formatting Documentation Naming Conventions

- Use 2 spaces for indentation (not tabs)
- Line length should not exceed 80 characters
- Opening braces for classes and functions on the same line
- Use meaningful variable and function names

```
function example_module_get_content($node_id, $options = []) {
  $result = [];
  // Implementation here
  return $result;
}
```

- All functions require docblock comments
- Comments should explain "why", not "what"
- Use `@param`, `@return`, `@throws` tags properly

```
/**
 * Retrieves formatted content for a specific node.
 *
 * @param int $node_id
 *   The node ID to fetch content for.
 * @param array $options
 *   (optional) Additional options for content retrieval.
 *
 * @return array
 *   Structured array of node content.
 *
 * @throws \Drupal\Core\Entity\EntityMalformedException
 *   When the node cannot be loaded or is invalid.
 */
function example_module_get_content($node_id, $options = []) {
  // Implementation here
}
```

- Module names: lowercase, underscores for spaces
- Function names: [module_name]_[description]
- Classes: CamelCase
- Variables: snake_case
- Constants: ALL_CAPS_WITH_UNDERSCORES

JavaScript Standards

- Follow Drupal JavaScript coding standards
- Use ES6 features with Babel transpilation
- Document with JSDoc comments
- Maintain proper namespacing for behaviors

CSS/SCSS Standards

- Use BEM (Block Element Modifier) methodology
- Maintain proper component namespacing
- Structure SCSS files by component
- Use variables for colors, fonts, and spacing

5.2.2 Security Best Practices

1. **Input Validation & Sanitization**
2. Always validate and sanitize user input
3. Use Drupal's sanitization functions
4. Implement proper access checks
5. **Database Queries**
6. Use parameterized queries with `db_query()`
7. Implement entity queries instead of raw SQL when possible
8. Never use user input directly in queries
9. **Form API Usage**
10. Implement proper form validation
11. Use CSRF tokens with `#token`
12. Check user permissions

5.2.3 Performance Considerations

- Implement appropriate caching strategies
- Use lazy loading where appropriate
- Optimize database queries
- Implement efficient render arrays

5.2.4 Snippet Library

Entity Loading

```
// Load a node by ID
$node = \Drupal\node\Entity\Node::load($node_id);

// Load multiple nodes
$nodes = \Drupal\node\Entity\Node::loadMultiple($node_ids);

// Load by properties
$query = \Drupal::entityQuery('node')
  ->condition('type', 'article')
  ->condition('status', 1)
  ->range(0, 10);
$nids = $query->execute();
$nodes = \Drupal\node\Entity\Node::loadMultiple($nids);
```

Custom Block Creation

```
use Drupal\block_content\Entity\BlockContent;
```

```
// Create a custom block
$block = BlockContent::create([
  'type' => 'basic',
  'info' => 'Block title',
  'body' => [
    'value' => 'Block content here',
    'format' => 'full_html',
  ],
]);
$block->save();
```

Form API Example

```
/**
 * Implements hook_form().
 */
function example_module_form($form, &$form_state) {
  $form['name'] = [
    '#type' => 'textfield',
    '#title' => t('Name'),
    '#required' => TRUE,
    '#maxlength' => 64,
  ];

  $form['email'] = [
    '#type' => 'email',
    '#title' => t('Email'),
    '#required' => TRUE,
  ];

  $form['submit'] = [
    '#type' => 'submit',
    '#value' => t('Submit'),
  ];

  return $form;
}

/**
 * Form validation handler.
 */
function example_module_form_validate($form, &$form_state) {
  // Validation logic here
}

/**
 * Form submission handler.
 */
function example_module_form_submit($form, &$form_state) {
  // Submission handling here
}
```

Custom AJAX Callback

```
/**
 * Implements form with AJAX callback.
 */
function example_module_ajax_form($form, &$form_state) {
  $form['selection'] = [
    '#type' => 'select',
    '#title' => t('Choose an option'),
    '#options' => [
      '1' => t('Option 1'),
      '2' => t('Option 2'),
    ],
    '#ajax' => [
      'callback' => 'example_module_ajax_callback',
      'wrapper' => 'ajax-container',
      'method' => 'replace',
      'effect' => 'fade',
    ],
  ];

  $form['container'] = [
    '#type' => 'container',
    '#attributes' => ['id' => 'ajax-container'],
  ];

  return $form;
```

```
}

/**
 * AJAX callback function.
 */
function example_module_ajax_callback($form, &$form_state) {
  // Process and return the container element
  return $form['container'];
}
```

5.2.5 Module Structure

Follow this structure for custom modules:

```
example_module/
├── example_module.info.yml
├── example_module.module
├── example_module.install
├── example_module.libraries.yml
├── example_module.routing.yml
├── example_module.services.yml
├── example_module.permissions.yml
├── src/
│   ├── Controller/
│   ├── Form/
│   ├── Plugin/
│   └── Service/
├── templates/
├── css/
├── js/
└── tests/
```

5.2.6 Code Review Checklist

Before submitting code for review:

- [ ] Code follows Drupal coding standards
- [ ] All functions have proper documentation
- [ ] Security best practices are followed
- [ ] Performance considerations addressed
- [ ] Tests have been written and pass
- [ ] No PHP errors or warnings
- [ ] Module dependencies properly defined
- [ ] No hardcoded values that should be configurable

Key Contact: Jason (Senior Backend Drupal Engineer)

April 28, 2025

April 28, 2025

5.3 SEO Redirect Decision Tree

This guide helps determine when and how to implement redirects to maintain SEO value and user experience.

> **Critical for SEO**
>
> Proper redirect implementation is critical for maintaining search rankings when URLs change. Always follow these guidelines for any URL modifications.

5.3.1 When to Use Redirects

Redirects are necessary in the following situations:

1. **Page URL changes** - When a page's URL structure changes
2. **Content consolidation** - When merging multiple pages into one
3. **Site restructuring** - When reorganizing site architecture
4. **Domain changes** - When moving to a new domain
5. **Campaign tracking** - When creating vanity URLs for marketing

5.3.2 Decision Tree

Use this decision tree to determine the appropriate redirect strategy:

```
graph TD
    A[Is the page moving permanently?] -->|Yes| B[Is the content similar?]
    A -->|No| C[Is this for a campaign?]
    B -->|Yes| D[Use 301 Redirect]
    B -->|No| E[Is there a relevant replacement?]
    E -->|Yes| D
    E -->|No| F[Return 404 or Custom Page]
    C -->|Yes| G[Use 302 Redirect]
    C -->|No| H[Is this temporary maintenance?]
    H -->|Yes| G
    H -->|No| F
```

5.3.3 301 vs. 302 vs. Canonical

301 Permanent Redirect 302 Temporary Redirect Canonical Tag

- **When to use**: Content has permanently moved to a new URL
- **SEO impact**: Passes ~90-99% of link equity to the new URL
- **Browser behavior**: Browser caches the redirect
- **Example scenario**: Changing URL structure, site migration

- **When to use**: Content is temporarily moved
- **SEO impact**: No long-term link equity transfer
- **Browser behavior**: Browser does not cache the redirect
- **Example scenario**: A/B testing, temporary promotions, maintenance

- **When to use**: Similar content exists at multiple URLs
- **SEO impact**: Indicates preferred URL for indexing
- **Browser behavior**: No effect on user experience
- **Example scenario**: Product pages with filters, pagination, tracking parameters

5.3.4 Implementation Guide

301 Redirect Implementation

1. **In Drupal**:
2. Use the Redirect module
3. Navigate to Configuration → Search and metadata → URL redirects
4. Add a new redirect with "Permanent (301)" status code
5. **In .htaccess** (Apache):

```
Redirect 301 /old-path /new-path
```

6. **In Nginx**:

```
location /old-path {
  return 301 /new-path;
}
```

302 Redirect Implementation

1. **In Drupal**:
2. Use the Redirect module
3. Set status code to "Found (302)"
4. **In .htaccess** (Apache):

```
Redirect 302 /old-path /new-path
```

5. **In Nginx**:

```
location /old-path {
  return 302 /new-path;
}
```

Canonical Tag Implementation

```
<link rel="canonical" href="https://www.uagc.edu/preferred-path" />
```

In Drupal, use the Metatag module to set canonical URLs for specific paths.

5.3.5 Best Practices

1. **Redirect Chain Prevention**
2. Always redirect to the final destination URL
3. Regularly audit and fix redirect chains
4. Keep a record of all implemented redirects
5. **URL Structure Maintenance**
6. Preserve URL structure when possible
7. Plan URL changes carefully
8. Implement bulk redirects for pattern changes
9. **Monitoring & Verification**
10. Check Search Console for crawl errors
11. Verify redirects work properly after implementation
12. Monitor organic traffic to redirected pages

5.3.6 Redirect Documentation

Document all redirects in the following format:

Original URL	Destination URL	Redirect Type	Implementation Date	Reason
/old-url	/new-url	301	YYYY-MM-DD	URL structure change
/promo	/campaign	302	YYYY-MM-DD	Temporary campaign

5.3.7 Common Issues & Solutions

> ⚠️ **Redirect Loops**
>
> A redirect loop occurs when page A redirects to page B, which redirects back to page A (or through a longer chain that eventually returns to the origin).
>
> **Solution**: Map out redirect paths before implementation and test thoroughly.

> ⚠️ **Redirect Chains**
>
> Multiple redirects in sequence slow down page load and dilute link equity.
>
> **Solution**: Always redirect to the final destination URL. Regularly audit and fix redirect chains.

> ⚠️ **404 After Redirect**
>
> Redirecting to a page that doesn't exist.
>
> **Solution**: Always verify destination URLs are valid before implementing redirects.

5.3.8 Tools for Testing Redirects

- Redirect Path (Chrome extension)
- Screaming Frog SEO Spider
- Redirect Checker

Key Contact: Omar (SEO & Tracking Manager)

April 28, 2025

April 28, 2025

5.4 Performance & Core Web Vitals Playbook

This guide provides strategies for monitoring and improving website performance with a focus on Core Web Vitals.

Core Web Vitals Impact

Core Web Vitals are a set of specific factors that Google considers important for user experience, and they directly impact search rankings.

5.4.1 Core Web Vitals Explained

Largest Contentful Paint (LCP) First Input Delay (FID) Cumulative Layout Shift (CLS)

What it measures: Loading performance - how quickly the main content of a page loads.

Good score: 2.5 seconds or faster

Key factors affecting LCP: - Server response time - Render-blocking JavaScript and CSS - Resource load time - Client-side rendering

What it measures: Interactivity - how quickly the page responds to user interactions.

Good score: 100 milliseconds or less

Key factors affecting FID: - Heavy JavaScript execution - Long tasks (over 50ms) - Large JavaScript bundles - Third-party code impact

What it measures: Visual stability - how much the page layout shifts unexpectedly.

Good score: 0.1 or less

Key factors affecting CLS: - Images without dimensions - Ads, embeds, and iframes without dimensions - Dynamically injected content - Web fonts causing FOIT/FOUT

5.4.2 Monitoring Performance

Tools

1. **Google PageSpeed Insights**
2. Provides Core Web Vitals scores for both mobile and desktop
3. Offers actionable recommendations
4. Link: PageSpeed Insights
5. **Google Search Console**
6. Shows Core Web Vitals performance across your site
7. Identifies pages that need improvement
8. Groups similar issues together

9. **Lighthouse**
10. Built into Chrome DevTools
11. Run directly from your browser
12. Provides detailed performance audit
13. **Web Vitals Chrome Extension**
14. Real-time monitoring of Core Web Vitals
15. Link: Web Vitals Extension

Monitoring Schedule

Task	Frequency	Owner
PageSpeed Insights check on key pages	Weekly	Brian
Search Console Core Web Vitals review	Monthly	Omar
Full Lighthouse audit	Quarterly	Brian
Performance regression testing	Before major releases	Jason/Brian

5.4.3 Improvement Strategies

LCP Optimization

1. **Server Optimization**
2. Implement proper caching headers
3. Use a CDN for static assets
4. Optimize application code
5. Implement database query optimization
6. **Resource Optimization**
7. Compress and properly size images
8. Implement proper image formats (WebP, AVIF)
9. Use responsive images with srcset
10. Defer non-critical CSS and JavaScript
11. **Critical Rendering Path**
12. Inline critical CSS
13. Defer non-critical JavaScript
14. Preload important resources
15. Reduce third-party JavaScript impact

FID Optimization

1. **JavaScript Optimization**
2. Break up long tasks

3. Use web workers for heavy operations
4. Implement code splitting
5. Defer or lazy-load non-critical JavaScript
6. **Third-Party Script Management**
7. Audit and reduce third-party scripts
8. Use async or defer attributes
9. Lazy-load third-party resources
10. Consider self-hosting critical third-party scripts
11. **Resource Prioritization**
12. Use `<link rel="preconnect">` for important third-party domains
13. Implement `<link rel="preload">` for critical assets
14. Use the Priority Hints API where appropriate

CLS Optimization

1. **Image Handling**
2. Always specify width and height attributes
3. Use aspect ratio boxes
4. Implement proper image loading strategies
5. **Font Loading Optimization**
6. Preload web fonts
7. Use `font-display: swap` or `font-display: optional`
8. Consider system font fallbacks
9. **Layout Stability**
10. Reserve space for ads and embeds
11. Avoid inserting content above existing content
12. Implement content placeholders

5.4.4 Implementation Guide

Quick Wins

These changes typically have a high impact-to-effort ratio:

1. **Image Optimization**

```
<img src="image.jpg" width="800" height="600" alt="Description"
     loading="lazy" decoding="async">
```

2. **Font Loading**

```
<link rel="preload" href="fonts/roboto.woff2" as="font" type="font/woff2" crossorigin>
```

3. **JavaScript Deferring**

```
<script src="non-critical.js" defer></script>
```

Drupal-Specific Optimizations

1. **Advanced CSS/JS Aggregation**
2. Install and configure the Advanced CSS/JS Aggregation module
3. Enable CSS/JS minification and compression
4. **Image Styles & Responsive Images**
5. Configure appropriate image styles
6. Use responsive image styles with breakpoints
7. **BigPipe**
8. Enable the core BigPipe module
9. Implement placeholders for personalized content
10. **Caching Optimization**
11. Configure Drupal's internal page cache
12. Implement a reverse proxy cache (Varnish/Fastly)
13. Set up proper cache tags and contexts

5.4.5 Performance Testing Process

Follow this workflow for comprehensive performance testing:

```
graph TD
    A[Identify Target Pages] --> B[Run Baseline Tests]
    B --> C[Identify Performance Issues]
    C --> D[Implement Fixes]
    D --> E[Run Comparative Tests]
    E --> F[Document Improvements]
    F --> G[Monitor for Regressions]
```

Before Deployment

1. Run Lighthouse tests on staging environment
2. Compare results with production baseline
3. Ensure no performance regressions
4. Document test results in deployment ticket

5.4.6 Handling Common Issues

Slow Server Response Time

- **Problem**: Server takes too long to respond to requests, impacting LCP
- **Solutions**:
- Optimize application code
- Implement proper caching
- Consider server upgrades if necessary
- Use a CDN for static assets

Layout Shifts from Ads

- **Problem**: Advertisements cause layout shifts when they load
- **Solutions**:
- Reserve space for ad slots using fixed dimensions
- Use CSS aspect ratio boxes
- Implement sticky ads that don't push content

Heavy JavaScript

- **Problem**: JavaScript execution blocks the main thread
- **Solutions**:
- Break up long tasks into smaller chunks
- Use code splitting and lazy loading
- Defer non-critical scripts
- Consider using web workers for heavy computation

5.4.7 References & Resources

- Web Vitals
- Optimize LCP
- Optimize FID
- Optimize CLS
- Drupal Performance Documentation

Key Contacts: - Brian (Front-End & QA Lead) - Front-end performance - Jason (Senior Backend Engineer) - Server-side performance

April 28, 2025

April 28, 2025

5.5 Privacy, Consent & Cookie Governance

This guide outlines our approach to user privacy, consent management, and cookie usage in compliance with global privacy regulations.

> **Legal Compliance**
>
> Proper privacy practices and consent management are legal requirements in many jurisdictions. Always follow these guidelines to ensure compliance.

5.5.1 Privacy Regulations Overview

GDPR (EU) CCPA/CPRA (California) PIPEDA (Canada)

- **Key requirements**: Explicit consent, right to access/delete data, data breach notification
- **Territorial scope**: Applies to EU users regardless of company location
- **Penalties**: Up to €20 million or 4% of global revenue
- **Cookie implications**: Requires explicit consent for non-essential cookies

- **Key requirements**: Right to know, delete, opt-out of data sales
- **Territorial scope**: Applies to businesses serving California residents
- **Penalties**: $2,500-$7,500 per violation
- **Cookie implications**: Must allow opt-out of "sale" of personal information

- **Key requirements**: Consent for collection/use/disclosure, right to access/correct data
- **Territorial scope**: Applies to organizations collecting data in Canada
- **Cookie implications**: Requires meaningful consent for tracking

5.5.2 Consent Management System

We use a consent management platform (CMP) to capture and store user privacy preferences:

1. **Initial presentation**: Cookie banner presented on first visit
2. **Consent capture**: User selection recorded and stored
3. **Preference center**: Accessible via "Privacy Settings" link in footer
4. **Consent renewal**: Prompted after 6 months or when privacy policy changes

Banner Configuration

Our cookie consent banner includes:

- Clear explanation of cookie usage
- Granular consent options (categories)
- Accept/Reject all buttons
- Link to privacy policy
- Link to detailed cookie settings

Implementation

```
<!-- Cookie consent banner implementation -->
<script>
  window.cookieConsentConfig = {
    categories: [
      { id: 'necessary', label: 'Necessary', isRequired: true },
      { id: 'functional', label: 'Functional' },
      { id: 'analytics', label: 'Analytics' },
      { id: 'marketing', label: 'Marketing' }
    ],
    consentExpiry: 180, // days
    onConsentCapture: function(consent) {
      // Set data layer variables for GTM
      window.dataLayer = window.dataLayer || [];
      window.dataLayer.push({
        'event': 'cookieConsentUpdate',
        'cookieConsent': consent
      });
    }
  };
</script>
<script src="/path/to/consent-manager.js" async></script>
```

5.5.3 Cookie Categorization

All cookies must be categorized and documented:

Necessary Functional Analytics Marketing

- **Description**: Essential for site functionality
- **Consent required**: No (always set)
- **Examples**: Session cookies, CSRF tokens, authentication
- **Retention**: Session or short-term (typically <30 days)

- **Description**: Enhance user experience but not essential
- **Consent required**: Yes
- **Examples**: Language preference, form pre-fill, user preferences
- **Retention**: Up to 1 year

- **Description**: Measure site usage and performance
- **Consent required**: Yes
- **Examples**: Google Analytics, Hotjar, internal analytics
- **Retention**: Varies (typically 1-2 years)

- **Description**: Used for advertising and personalization
- **Consent required**: Yes
- **Examples**: Ad tracking, remarketing pixels, A/B testing
- **Retention**: Varies by vendor (typically 1-2 years)

5.5.4 Cookie Inventory Management

Maintain a complete inventory of all cookies in the following format:

Cookie Name	Category	Purpose	Provider	Duration	Personal Data
JSESSIONID	Necessary	Session management	UAGC	Session	No
_ga	Analytics	User distinction	Google Analytics	2 years	Yes (pseudonymous)

Regular Audit Process

1. Scan website for cookies quarterly (using Cookie Scanner tool)
2. Update cookie inventory documentation
3. Ensure all cookies have proper categorization
4. Verify technical implementation respects consent

5.5.5 Technical Implementation

Google Tag Manager Setup

Our GTM implementation respects consent settings:

1. **Variable setup**:
2. Create cookie consent variables for each category
3. Use Data Layer variables to read consent status
4. **Trigger configuration**:
5. Create triggers that check consent status
6. Use these triggers for all tags requiring consent
7. **Tag management**:
8. Group tags by consent category
9. Set appropriate consent checks before firing

Code Example

```
// Example GTM tag firing logic
function canLoadTag(category) {
  return (
    dataLayer.find(item =>
      item.cookieConsent &&
      item.cookieConsent[category] === true
    ) !== undefined
  );
}

// Only load analytics if consent given
if (canLoadTag('analytics')) {
  // Load analytics code
}
```

5.5.6 Privacy Policy Management

Our privacy policy must be:

1. **Comprehensive**: Cover all data collection, processing, sharing
2. **Accessible**: Easy to find, read, and understand
3. **Current**: Updated when practices change
4. **Accurate**: Reflect actual practices

Update Workflow

```
graph TD
    A[Identify Need for Update] --> B[Legal Team Review]
    B --> C[Draft Changes]
    C --> D[Stakeholder Review]
    D --> E[Final Legal Approval]
    E --> F[Update Live Policy]
    F --> G[Update Version/Date]
    G --> H[Notify Users if Significant]
```

5.5.7 Data Subject Rights Handling

Process for handling user data requests:

1. **Request intake**: Form on website or email to privacy@uagc.edu
2. **Identity verification**: Confirm requestor's identity
3. **Request processing**: Gather requested information
4. **Response**: Deliver within required timeframe (30 days for GDPR)
5. **Documentation**: Record all requests and responses

Request Types

- **Access**: Provide copy of all personal data
- **Deletion**: Remove user's personal data
- **Correction**: Update inaccurate data
- **Portability**: Provide data in machine-readable format
- **Restriction**: Limit processing of user's data
- **Objection**: Stop processing based on legitimate interests

5.5.8 Vendor Management

For third-party services processing personal data:

1. **Assessment**: Evaluate privacy practices before implementation
2. **Data Processing Agreement**: Execute with all data processors
3. **Consent check**: Ensure vendors respect user consent choices
4. **Regular audit**: Review vendor compliance annually

5.5.9 Training & Compliance

All team members involved with the website must:

1. Complete privacy training annually
2. Understand which features require consent
3. Know how to handle data subject requests
4. Report potential privacy issues immediately

5.5.10 Resources

- GDPR Official Text
- CCPA Information
- Cookie Scanner Tool
- UAGC Privacy Policy

Key Contacts: - Omar (SEO & Tracking Manager) - Consent implementation - Legal Department - Policy questions & data requests

April 28, 2025

April 28, 2025

5.6 Release Checklist & Incident Rollback Procedure

This guide provides a structured approach for planning releases and responding to incidents that require rollbacks.

> **Critical Procedures**
>
> These procedures are critical to maintaining site stability. Follow them precisely for all production changes.

5.6.1 Release Planning

Types of Releases

Standard Release Hotfix Emergency Fix

- Scheduled on a regular cadence
- Contains non-urgent feature updates and bug fixes
- Follows complete QA and testing process
- Deployed during scheduled maintenance windows
- Requires approval from Thomas (DX Director)

- Addresses critical bugs affecting user experience
- Follows expedited but thorough QA
- May be deployed outside scheduled windows
- Requires approval from Thomas or Jason

- Resolves security vulnerabilities or site-breaking issues
- Minimal but focused QA focused on the specific issue
- Can be deployed immediately when necessary
- Requires post-implementation review

Release Schedule

Standard maintenance windows:

Day	Time	Purpose
Monday	5:00-7:00 AM PT	Backend deployments
Wednesday	5:00-7:00 AM PT	Frontend deployments
First Sunday of month	11:00 PM-3:00 AM PT	Major system updates

5.6.2 Pre-Release Checklist

1. Planning & Documentation

- [ ] Release clearly defined in Asana with all included tasks
- [ ] Dependencies between tasks identified and ordered
- [ ] Release notes prepared with all changes documented
- [ ] Stakeholders notified of upcoming changes
- [ ] Deployment window confirmed

2. Code Review

- [ ] All code changes have been peer-reviewed
- [ ] Security review completed for sensitive changes
- [ ] Performance implications assessed
- [ ] Coding standards followed
- [ ] Documentation updated

3. Testing

- [ ] Unit tests pass
- [ ] Integration tests pass
- [ ] End-to-end tests pass
- [ ] Manual QA completed on staging
- [ ] Tested across required browsers/devices
- [ ] Accessibility testing completed
- [ ] SEO impact verified

4. Deployment Preparation

- [ ] Database backup scheduled
- [ ] Rollback plan documented
- [ ] Configuration changes documented
- [ ] Feature flags configured (if applicable)
- [ ] Load balancer strategy confirmed

5. Final Approval

- [ ] Final sign-off from Thomas
- [ ] QA sign-off from Brian
- [ ] Technical sign-off from Jason
- [ ] Communication plan approved
- [ ] Deployment team assigned and briefed

5.6.3 Deployment Process

```
graph TD
    A[Pre-Deployment Meeting] --> B[Verify Checklist Complete]
    B --> C[Create Full System Backup]
    C --> D[Begin Deployment Window]
    D --> E[Enable Maintenance Mode]
    E --> F[Deploy Code Changes]
    F --> G[Apply Configuration Changes]
    G --> H[Run Update Scripts]
    H --> I[Clear Caches]
    I --> J[Verify Deployment]
    J --> K[Disable Maintenance Mode]
    K --> L[Monitor System]
```

Deployment Verification

1. **Smoke Tests**:
2. Verify key pages load correctly
3. Check critical user flows
4. Confirm functionality of primary actions
5. Verify search functionality
6. Test form submissions
7. **Monitoring**:
8. Check error logs for new issues
9. Monitor application performance
10. Verify database performance
11. Watch for unusual traffic patterns
12. Check third-party integrations

5.6.4 Incident Response

Severity Levels

Severity 1 (Critical) Severity 2 (High) Severity 3 (Medium) Severity 4 (Low)

- Site completely down or unusable
- Data breach in progress
- Payment processing broken
- Response time: **Immediate**
- Notification: All stakeholders

- Major functionality broken
- Significant performance degradation
- Unable to complete enrollment
- Response time: **< 1 hour**
- Notification: Technical team and management

- Minor feature broken
- Non-critical path affected
- Cosmetic issues affecting experience
- Response time: **Same business day**
- Notification: Technical team

- Trivial bugs
- Minor visual glitches
- Non-critical improvements
- Response time: **Next release**
- Notification: Added to backlog

Incident Detection

Incidents may be detected through:

- Automated monitoring alerts
- User reports
- Social media mentions
- Manual testing
- Support tickets

Incident Response Workflow

```
graph TD
    A[Incident Detected] --> B[Assess Severity]
    B --> C[Assemble Response Team]
    C --> D[Diagnose Root Cause]
    D --> E{Resolution Viable?}
    E -->|Yes| F[Implement Fix]
    E -->|No| G[Implement Rollback]
    F --> H[Verify Resolution]
```

```
    G --> H
    H --> I[Post-Incident Review]
```

5.6.5 Rollback Procedures

Decision Criteria

Consider rollback when:

- Critical functionality is broken
- Security vulnerability is exposed
- Data integrity is at risk
- The fix is complex or uncertain
- Resolution time exceeds acceptable downtime

Drupal Rollback Process

1. **Codebase Rollback**:

```
cd /var/www/uagc
git checkout [previous-stable-tag]
```

2. **Database Rollback**:

```
# Stop web services
systemctl stop apache2

# Restore database
mysql -u [user] -p [database] < /path/to/backup/[backup-file].sql

# Restart web services
systemctl start apache2
```

3. **Configuration Rollback**:

```
# Import previous configuration
drush config:import --source=/path/to/config-backup
```

4. **Cache Clearing**:

```
drush cache:rebuild
```

Content Deployment Rollback

For content-only changes:

1. **Revert individual nodes**:

```
drush entity:delete node [nid]
```

2. **Restore from revisions**:

```
drush php:eval 'node_revision_revert([nid], [vid])'
```

5.6.6 Post-Incident Analysis

After every major incident or rollback:

1. **Document the incident**:
2. Timeline of events
3. Systems affected
4. User impact
5. Resolution steps taken
6. **Conduct blameless post-mortem**:
7. What happened
8. Why it happened
9. How we detected it
10. How we resolved it
11. What we learned
12. **Create preventative actions**:
13. Process improvements
14. Monitoring enhancements
15. Testing improvements
16. Training needed

5.6.7 Communication Templates

Planned Maintenance Notice

```
SCHEDULED MAINTENANCE NOTICE

We will be performing scheduled maintenance on the UAGC website on
[DATE] from [START TIME] to [END TIME] [TIMEZONE].

During this time, the website may be unavailable or experience intermittent issues.
We apologize for any inconvenience and appreciate your patience.

If you have any questions, please contact support@uagc.edu.
```

Incident Communication

```
SYSTEM STATUS UPDATE

We are currently experiencing [BRIEF DESCRIPTION OF ISSUE] affecting [AFFECTED SYSTEMS].
Our team is actively working to resolve this issue.

IMPACT: [DESCRIBE USER IMPACT]
WORKAROUND: [IF AVAILABLE]
ESTIMATED RESOLUTION: [TIME IF KNOWN]

We will provide updates as more information becomes available.
```

5.6.8 Emergency Contacts

Role	Name	Primary Contact	Secondary Contact
DX Director	Thomas	thomas@uagc.edu	(555) 123-4567
Senior Drupal Engineer	Jason	jason@uagc.edu	(555) 123-4568
Frontend Lead	Brian	brian@uagc.edu	(555) 123-4569
SEO & Tracking	Omar	omar@uagc.edu	(555) 123-4570
Hosting Provider	Acquia	support@acquia.com	(555) 123-0000

Key Contacts: Jason (Deployments), Thomas (Approval)

April 28, 2025

April 28, 2025

6. Reference

6.1 Analytics Standards

This document outlines the standards and best practices for analytics implementation across the UAGC website.

6.1.1 GA4 Implementation

Event Naming Conventions

We follow these standard naming conventions for GA4 events:

- Use snake_case for event names
- Keep names descriptive but concise
- Follow a consistent verb_noun pattern where possible

Examples of standardized event names: - `page_view` - `form_start` - `form_submit` - `form_error` - `click_cta` - `download_document`

Parameters and Values

Standard parameters to include with events:

Parameter	Description	Example
`page_location`	Full URL of the page	`https://www.uagc.edu/admissions`
`page_title`	Title of the page	`Admissions - UAGC`
`event_category`	Category of the event	`form`, `navigation`, `engagement`
`event_label`	Descriptive label	`RFI Form`, `Hero CTA`
`item_id`	ID of the item interacted with	`rfi-form-homepage`

6.1.2 DataLayer Implementation

DataLayer Structure

Our dataLayer follows this structure:

```
window.dataLayer = window.dataLayer || [];
window.dataLayer.push({
  'event': 'event_name',
  'eventCategory': 'category',
  'eventAction': 'action',
  'eventLabel': 'label',
  'additionalData1': 'value',
  'additionalData2': 'value'
});
```

Common DataLayer Events

Events that should always be tracked:

- Page views
- Form interactions (start, complete, error)
- CTA clicks
- Video interactions (play, pause, complete)
- File downloads
- Optimizely experiment views and conversions

6.1.3 Experiment Tracking

Optimizely Integration

When setting up experiments, ensure these tracking elements are in place:

1. Experiment view event when variation is shown
2. Conversion events for primary and secondary metrics
3. Custom dimension for experiment name and variation

Example GA4 event for experiment view:

```
window.dataLayer.push({
  'event': 'experiment_view',
  'experimentName': 'homepage_hero_test',
  'experimentVariation': 'variation_1',
  'experimentId': 'EXP123'
});
```

6.1.4 Implementation Verification

Before launch, verify that:

1. All required events fire correctly
2. Custom dimensions pass the right values
3. Events show up in GA4 DebugView
4. DataLayer contains all required information
5. No duplicate events are firing

6.1.5 Governance

- Omar oversees all analytics implementation and standards
- Any changes to tracking structure must be approved before implementation
- Documentation must be updated when new tracking patterns are introduced

April 28, 2025

April 28, 2025

6.2 Glossary of Internal Acronyms & Naming Conventions

This glossary provides definitions for commonly used acronyms, terms, and naming conventions used within the UAGC Digital Experience team.

Living Document

This glossary is a living document. If you encounter a term that isn't included, please add it following the existing format.

6.2.1 Acronyms & Abbreviations

General

Acronym	Full Term	Definition
UAGC	University of Arizona Global Campus	Our institution name
DX	Digital Experience	Our team responsible for digital properties
RFI	Request for Information	Form filled out by prospects to get more information
CMS	Content Management System	Software for creating/managing digital content (Drupal)
SEO	Search Engine Optimization	Practices to improve search visibility
SERP	Search Engine Results Page	Results page after a search query
CTA	Call to Action	Prompt encouraging user to take specific action
KPI	Key Performance Indicator	Metric used to evaluate success
ROI	Return on Investment	Value gained relative to cost
SLA	Service Level Agreement	Commitment between service provider and client
SOW	Statement of Work	Document outlining project deliverables
UAT	User Acceptance Testing	Final testing phase before release
UX	User Experience	Overall experience of a user with a product
UI	User Interface	Visual elements users interact with

Technical

Acronym	Full Term	Definition
API	Application Programming Interface	Protocol for building and integrating software
CDN	Content Delivery Network	Distributed servers for content delivery
CI/CD	Continuous Integration/Continuous Deployment	Automated methods for software delivery
CWV	Core Web Vitals	Google's page experience metrics
DNS	Domain Name System	System for converting domain names to IP addresses
GA4	Google Analytics 4	Google's latest analytics platform
GTM	Google Tag Manager	Tag management system for analytics and marketing
HTML	HyperText Markup Language	Standard markup language for web pages
HTTP	HyperText Transfer Protocol	Protocol for transmitting hypermedia documents
HTTPS	HTTP Secure	Encrypted version of HTTP
JS	JavaScript	Programming language for web development
LCP	Largest Contentful Paint	Core Web Vital measuring loading performance
CLS	Cumulative Layout Shift	Core Web Vital measuring visual stability
FID	First Input Delay	Core Web Vital measuring interactivity
PHP	PHP: Hypertext Preprocessor	Server-side scripting language
REST	Representational State Transfer	Architectural style for API design
SSL	Secure Sockets Layer	Technology for securing internet connections
WCAG		

Acronym	Full Term	Definition
	Web Content Accessibility Guidelines	Accessibility standards for web content

Project-Specific

Acronym	Full Term	Definition
ECM	Enrollment Contact Method	System for tracking enrollment inquiries
FPP	Full Program Page	Pages detailing full academic programs
BPP	Brief Program Page	Shortened program information pages
EPP	Enrollment Processing Platform	System for processing student enrollments
LSP	Lead Submission Process	Workflow for processing prospective student leads
PSO	Program-Specific Outcomes	Academic outcomes for specific programs
MCE	Multi-Channel Experience	Cross-channel marketing and engagement
PCR	Page Change Request	Process for requesting content changes
TMA	Traffic-Monitoring Algorithm	System for tracking website traffic patterns

6.2.2 Naming Conventions

File Naming

- **Images**: `[section]-[descriptor]-[dimension].[extension]`
- Example: `hero-nursing-1200x600.jpg`
- **Documents**: `[type]-[topic]-[version].[extension]`
- Example: `policy-privacy-v2.pdf`
- **Code Files**: `[functionality].[language extension]`
- Example: `lead-form-validation.js`

URL Structure

- **Program Pages**: `/programs/[degree-level]/[program-name]/`
- Example: `/programs/bachelors/business-administration/`
- **Information Pages**: `/[section]/[topic]/`
- Example: `/admissions/transfer-credits/`

- **Landing Pages**: `/[campaign-source]/[campaign-name]/`
- Example: `/google/nursing-2023/`

Database Naming

- **Tables**: `uagc_[entity]_[function]`
- Example: `uagc_user_preferences`
- **Fields**: `field_[entity]_[attribute]`
- Example: `field_student_phone`

Component Naming

- **UI Components**: `uagc-[component-type]--[variant]`
- Example: `uagc-button--primary`
- **Blocks**: `block-[section]-[function]`
- Example: `block-header-navigation`

6.2.3 Common Terms

Business Terms

Term	Definition
Conversion	When a user completes a desired action (form submission, enrollment)
Funnel	The journey a user takes from initial awareness to conversion
Lead	A potential student who has shown interest by submitting information
Prospect	An individual who may become a student but hasn't applied
Applicant	Someone who has submitted an application but isn't enrolled
Enrolled Student	Individual who has completed enrollment and is taking courses
Academic Program	A course of study leading to a degree or certificate
Modality	Method of delivering education (online, hybrid, in-person)

Technical Terms

Term	Definition
Cache	Temporary storage of web pages and assets to improve performance
Dependency	External code/library that a project relies on
Endpoint	URL where an API can be accessed
Environment	Server setup where code runs (development, staging, production)
Middleware	Software that connects applications or components
Migration	Process of moving data or code from one system to another
Refactoring	Restructuring code without changing functionality
Regression	When new changes break existing functionality
Rendering	Process of generating visual output from code
Responsive	Design approach that adapts to different screen sizes
Schema	Structure that defines how data is organized
Sandbox	Isolated testing environment
Staging	Pre-production environment for testing
Version Control	System for tracking and managing code changes

6.2.4 Department Codes

Code	Department
ADMN	Administration
ADMS	Admissions
ADVS	Advising
FINS	Financial Services
MKTG	Marketing
REGR	Registrar
TECH	Technology Services
ACAD	Academic Affairs

6.2.5 Status Codes

Code	Status	Definition
ACT	Active	Currently in use or ongoing
ARCH	Archived	No longer in use but preserved
DPRC	Deprecated	Still available but being phased out
DRAF	Draft	In development, not yet approved
PEND	Pending	Awaiting approval or implementation
REV	In Review	Being evaluated for approval
SUSP	Suspended	Temporarily unavailable

6.2.6 Resources

- UAGC Brand Guidelines
- Web Development Standards
- Editorial Style Guide

Key Contact: Brandy (Digital Marketing & Web Operations Manager)

April 28, 2025

April 28, 2025

6.3 Documentation Workflow

This page outlines the process for creating, publishing, and maintaining documentation in our DX documentation system.

6.3.1 Workflow

When adding or updating documentation:

```
graph TD
  A[Identify Need] --> B[Create Asana Task]
  B --> C[Draft Content]
  C --> D[Technical Review]
  D --> E[Edit and Format]
  E --> F[Publish]
  F --> G[Announce]
  G --> H[Maintain]
```

Step by Step Process

1. Identify the need

Determine what documentation is missing or needs updating. Look for recurring questions, process changes, or new features.

2. Create Asana task

Use the Documentation template to log the task. Include the documentation topic, priority, and due date.

3. Draft content

Write the documentation following our style guidelines. Use the existing templates and structure.

4. Technical review

Have relevant subject matter experts review for accuracy. Make sure all technical details are correct.

5. Edit and format

Ensure the documentation follows our formatting standards. Apply Markdown formatting, add images if needed.

6. Publish

Add the content to the documentation platform using our GitHub-based workflow.

7. Announce

Let the team know about the new/updated documentation via Slack in the #dx-team channel.

8. Maintain

Schedule regular reviews to keep content fresh. Each section has a designated owner.

6.3.2 Quality Guardrails

We maintain high-quality documentation through these standards:

Clarity Completeness Consistency Currency Conciseness

Instructions should be clear and unambiguous. Use simple language and avoid jargon when possible. If technical terms are necessary, consider linking to definitions or providing explanations.

Cover all necessary steps and edge cases. Don't leave readers wondering "what if" or "what next." Include troubleshooting steps for common issues.

Follow established formatting and style guidelines. Use consistent terminology, capitalization, and structure across all documentation.

Keep information up-to-date with current practices. Outdated documentation is often worse than no documentation at all. Schedule regular reviews.

Be thorough but efficient with words. Focus on what readers need to know, not on everything that could be said about a topic.

6.3.3 Quarterly "Docs Day"

To ensure our documentation remains relevant and accurate:

1. Block 2 hours every quarter for a team "Docs Day"
2. Each owner reviews their documentation sections
3. Update any outdated information
4. Remove or archive stale content
5. Identify gaps requiring new documentation

Docs Day Best Practices

- Schedule the event in advance and make it a priority
- Use a checklist to track reviews and updates
- Take notes on improvement ideas
- Celebrate progress and improvements

The goal is to maintain living documentation that grows and evolves with our processes.

6.3.4 Documentation Ownership

Section	Owner	Review Frequency
Asana	Brandy	Quarterly
Request Information Form	Anthony/Omar	Quarterly
Day-to-Day Ops	Thomas	Quarterly
Documentation Guides	All	Quarterly

Ownership Responsibilities

As an owner, you are responsible for:

- Accuracy of content
- Timely updates when processes change
- Quarterly review completion
- Responding to feedback about your section

April 28, 2025

April 28, 2025

6.4 Growth Roadmap

This page outlines our phased approach to developing and improving the DX documentation platform.

6.4.1 Phases Overview

Phase	Goal	Timeline
1: MVP	Home page, templates, top-5 guides	Weeks 1-2
2: Team Onboard	Asana form live, everyone writes one doc	Weeks 3-4
3: Automation	Linting, search, Slack bots	Month 2
4: Insights	GA dashboard for doc usage, feedback widget	Month 3+

6.4.2 Phase 1: MVP

The Minimum Viable Product phase focuses on establishing the foundation:

- Create the documentation platform home page
- Develop standardized templates for different document types
- Write the five most critical guides:
- Getting Started
- Add/Remove/Redirect Pages
- Optimizely Tests
- Basic SEO Hygiene
- QA Smoke Test

6.4.3 Phase 2: Team Onboard

The Team Onboarding phase gets everyone engaged with documentation:

- Integrate Asana form for documentation requests
- Have each team member create at least one document
- Conduct training session on documentation standards
- Get feedback on initial documentation structure

6.4.4 Phase 3: Automation

The Automation phase streamlines documentation processes:

- Implement automated linting for documentation formatting
- Enhance search functionality for faster information retrieval
- Develop Slack integrations for documentation queries
- Create automation for documentation maintenance tasks

6.4.5 Phase 4: Insights

The Insights phase focuses on measuring and improving documentation value:

- Build Google Analytics dashboard for documentation usage
- Add feedback widget to gather user input
- Analyze documentation effectiveness metrics
- Implement improvements based on user feedback and analytics

6.4.6 Beyond Phase 4

Future enhancements may include:

- Knowledge base integration
- AI-powered content suggestions
- Automated documentation testing
- Advanced visualization tools

April 28, 2025

April 28, 2025

6.5 Site Map

This page provides a complete overview of all documentation available on the UAGC DX Documentation Server.

6.5.1 Core Documents

- Home
- Why This Documentation Exists
- Who Does What
- Asana
- Request Information Form
- Day-to-Day Operations
- Documentation Workflow
- Growth Roadmap

6.5.2 Documentation Guides

Basic Guides

- Getting Started with Documentation
- Add/Remove/Redirect Pages
- Optimizely Tests
- SEO Hygiene
- QA Smoke Test

Advanced Guides

- Accessibility & Inclusive Design
- Drupal Coding Standards & Snippet Library
- SEO Redirect Decision Tree
- Performance & Core Web Vitals Playbook
- Privacy, Consent & Cookie Governance
- Release Checklist & Incident Rollback Procedure

6.5.3 Reference Materials

- Analytics Standards
- Glossary of Internal Acronyms & Naming Conventions

6.5.4 By Task

Content Management

- Add/Remove/Redirect Pages

- SEO Redirect Decision Tree

Experimentation

- Optimizely Tests

SEO & Analytics

- SEO Hygiene
- Analytics Standards
- SEO Redirect Decision Tree
- Privacy, Consent & Cookie Governance

Quality Assurance

- QA Smoke Test
- Performance & Core Web Vitals Playbook
- Accessibility & Inclusive Design

Development

- Drupal Coding Standards & Snippet Library
- Release Checklist & Incident Rollback Procedure

Documentation

- Getting Started with Documentation
- Documentation Workflow
- Glossary of Internal Acronyms & Naming Conventions

6.5.5 By Team Member

Leadership

- Growth Roadmap
- Who Does What
- Release Checklist & Incident Rollback Procedure

Frontend & QA

- QA Smoke Test
- Accessibility & Inclusive Design
- Performance & Core Web Vitals Playbook

Backend & Drupal

- Add/Remove/Redirect Pages
- Drupal Coding Standards & Snippet Library

- Release Checklist & Incident Rollback Procedure

SEO & Analytics

- SEO Hygiene
- Analytics Standards
- SEO Redirect Decision Tree
- Privacy, Consent & Cookie Governance
- Request Information Form

April 28, 2025

April 28, 2025