VA.gov Content Editing Options

Content editing options in support of a successful merger break down toward three primary implementation paths, that result in either a build-based integration (content triggers a publishing process) or a realtime content consumption (content API / headless). Each path requires consideration of a content management system (CMS), which is defined, for the purposes of this discussion, a system of managing content editing workflows and providing an interface to the product of those workflows.

We believe that each approach outlined here satisfies the short term requirements, while providing the ability to evaluate and tune workflows and implementations toward a more finished final product. This final product may be a standalone CMS (TeamSite/Drupal/Wordpress/etc) that provides publishing hooks or APIs for realtime site consumption. Solutions here can be considered part of a spectrum of approaches, which can be utilized independently or together. Of note, each solution provides a dramatic improvement in publishing time and a separate, enhanced workflow for content management.

Option 1: Utilize TeamSite with Additional Workflows

Build a form(s) in TeamSite that content editors can use via the TeamSite GUI. Define workflows within TeamSite that allow for rapid deployment and assign the appropriate content editors / approvers to those flows. This generates content files that are consumed by the existing vets-website frontend. Multiple paths forward exist for this approach:

1) Build integration: Expose new content in a machine-readable format that triggers a new build and deployment in vets-website. This build would not be tied to application level testing, which reduces the time required to make changes. Additionally, workflow for content approval is performed outside of the existing process, which allows for additional flexibility.
2) Real-time content consumption: Expose new content in a machine-readable format along with an associated content index that vets-website consumes dynamically. This option provides flexibility with new pages and allows for nearly instantaneous updates on vets-website. Integration options may prove initially problematic with SEO or latency, though technical solutions may optimize this behavior.

We suggest an iterative approach to this migration, transitioning content to the new system and workflows, while maintaining the existing content management workflow. Eventually, all current Vets.gov content will be migrated to TeamSite.

Developer LOE: Heavy
End User LOE: Light

Risks	**Rewards**
● Developers will need to learn how to build TeamSite forms and templates as well as how to develop interfaces to expose this content to the website. ● Developers will need a dedicated OIT	● Content editing workflows maintained within asystem they're already comfortable with ● Edits are published quickly/immediately ● All content managed in one system, doesn't require duplicating auth rules in another

resource that can give us their focus. ● Developers will need full access to TeamSite systems. ● If the team cannot implement hooks within the TeamSite system to trigger new builds, we'll rely on dynamic content indexes and generation, locking us in to implementation route #2. ● Additional TeamSite licenses are prohibitively expensive.	system

Option 2: Provide Alternative Content Workflows with Separate Repository

Spin up a separate repo and create a build process that doesn't require approvals to merge. Upon merge to master, it would expose data that we can pull into Vets.gov's codebase without going through the testing and approval process on vets-website. Workflow modifications would take advantage of the existing GitHub contribution rules, and would be managed by repository settings and teams. Some content edits would require pull requests with associated review before merge to master, while some members would have the ability to write directly to master.
Initially, content editors would be required to use many of the same systems and structures that are currently configured for static content editing on vets-website. As content is migrated to this system, it becomes a content database. It can then become more easily replaced by a more full featured content management system, such as Drupal, Wordpress, or a modified TeamSite. The primary advantage is separation of content editing workflows from the application development workflow. This will dramatically simplify the delivery of and maintenance of content for VA.gov compared to the current Vets.gov system.

We recommend the first iteration be directly tied to an optimized build process for vets-website (skipping unnecessary testing and build stages). If this model is well received by content editors, the next step would be to trigger the generation of machine readable content indexes and pages (following the example of successful industry services such as contentful), allowing for dynamic and instantaneous consumption by the website. If build pipeline efficiency for content edits is successful, this step may not be necessary, though we anticipate additional enhancements to the content APIs may be useful.

Option #3 represents an incremental step toward this solution, since this work would be equally applicable (though perhaps not as necessary) to this approach.

Developer LOE: Medium
End User LOE: Heavy

Risks	**Rewards**
● Content editing remains difficult since we're still using GitHub. User training and process	● Edits are published quickly/immediately ● Team would not have to learn ins and outs of

documentation is critical.	TeamSite to build (less delay) ● We would have total control over *how* the user edits the content ● Minimized build process for updates

Option 3: Improve Content Editing Workflow in Existing Configuration; ; Train Users

We would train content editors on editing markdown files in GitHub, submitting pull requests, and tagging reviewers. VFW team would manage giving users access to the GitHub repo. The build pipeline would be updated to remove unnecessary testing (accessibility, security, unit, and integration tests) for content-specific changes. We could further optimize by preventing a rebuild of stylesheets and javascript.

Developer LOE: Medium
End User LOE:

Risks	**Rewards**
● Content editing remains difficult since we're still using GitHub. User training and process documentation is critical. ● ● VFT team may end up spending more time approving or fixing end user-submitted PRs than anticipated ● Can't be used for immediate updates	● Little developer effort, would only need to set up approval workflow and participate in training ● Edits are published quickly