MHV-on-VA.gov Tech Overview

Patrick Vinograd
2/8/2022

Alignment

In my mind the key technical outcome we can accomplish in the next couple months is to identify any potential alignment issues and start mitigating them. This may require us to write code to prove out a concept or evaluate an API, but a large part of the work will be talking to people and documenting things. The kinds of mis-alignment I'm thinking about:

* Mis-alignment between the functionality expressed in designs, and the currently available MHV APIs.
* Mis-alignment between the expressed designs and the VA design system.
* Technical mis-alignment between the expected architecture of the MHV-on-VA features and the VA.gov platform
* Organizational mis-alignment between the MHV team's expected workflow and the VA.gov platform collaboration cycle
* Mis-alignment between VistA and Cerner APIs that have impacts on desired functionality.

In other words the main challenge is that we are fitting together a legacy web application, an organically-grown development platform, a legacy EHR system, and a commercial EHR system, and multiple disparate organizations, and trying to come out the other end with a cohesive web application.

Past and Current State

Simple View

* VA.gov as we refer to it comprises the content and applications available at the www.va.gov domain and all sub-paths. Operationally it also refers to the platform that supports those applications/content, and the infrastructure that hosts all of it.
* Other sub-domains (e.g. myhealth.va.gov, benefits.va.gov, www.accesstocare.va.gov, etc are implemented as entirely separate web properties in their own hosting environments, operated by separate teams.
* The interactive parts of VA.gov are implemented as a React frontend and a Ruby on Rails backend.

* The frontend resources (HTML/CSS/JS) - everything available at www.va.gov - are served out of S3.
* The backend API is available at api.va.gov and is currently hosted on EC2 instances.
* The backend API (known as vets-api) is largely an API facade. In order to service most requests, vets-api needs to reach out to other VA systems of record to submit or retrieve data. (There are on the order of 20-30 other systems that vets-api connects to). Vets-api abstracts some of the complexity of these upstream systems, and provides a cohesive REST-ful API surface area with a unified authentication mechanism, consistent error behavior, and other conveniences that make data easier to consume by frontend applications.

* MHV has a similar frontend/backend web application design. The frontend is an (unknown) JS framework. The backend is Java-based. MHV's hosting environment is unknown.
* MHV does own some of its own data. But it also integrates with other VA systems, notably VistA (VA's home-grown electronic health record system).

* Having two separate web properties introduces friction for users
  * Single sign on exists across the sites, as of fairly recently, but it is somewhat limited and kludgy.
  * MHV has adopted the VA design system to some degree, but there are still obvious look and feel differences between the sites.
  * There's plenty of linking between various areas of the two sites, but doing so cohesively is made more difficult by the single sign in process, and the fact that the two properties are operated completely independently, using different agile release processes.

Caveats

* There are some legacy sub-paths under www.va.gov that are proxied through the VA.gov infrastructure but hosted elsewhere.
* Vets-api does persist and cache *some* data - either data specific to the site operations (like user preferences) or it will retain things like form submissions to allow for retries or batch submission to upstream systems. But by and large VA.gov is not considered the "system of record" for any benefits or health data.

More Detailed View - VA.gov

Platform vs. apps

* VA.gov is separated into a platform, and many veteran facing service (VFS) applications that ride atop that platform.
* However it's important to note that the platform/app split grew organically out of a single team building a web application. OCTO decided to institute the platform-application split starting from this working application.
* So, the platform is not as well-isolated or self-service as you'd expect from something like Heroku.
  * VFS application teams build their frontend and backend modules in a monolithic codebase alongside the other application teams and platform teams.
  * The entire frontend and backend are each deployed once per day - teams have no autonomy over deploying more or less frequently without an escalation process.
  * A lot of the steps along the path to deployment still require human interaction (mainly via the `#vfs-platform-support` slack channel). VFS teams need to request PR reviews from the platform before merging code; they need manual assistance for configuration management and other devops tasks.
  * The "collaboration cycle" is a set of gating review steps intended to maintain quality by having the platform team review VFS teams' applications for things like design, accessibility, security, and launch readiness.
* Generally this system works okay in that it has resulted in a usable, reasonably consistent VA.gov, but it's high-touch for both VFS and platform personnel, and ultimately doesn't scale well.
* And, it assumes that teams build in a certain way - in React/Rails, managing issues in va.gov-team, following the collaboration cycle, participating in DSVA slack. *Whether these assumptions will hold for the MHV team is an open question.*
* The platform is moving towards a more self-service model with better application isolation and autonomy. See "Possible Future States / Platform flux" below for more details about what will be possible.

Frontend details

* Static content is managed in a CMS
* React applications live in [vets-website](https://github.com/department-of-veterans-affairs/vets-website/)

  * At this point there's fairly clean organizational separation between applications and shared frontend platform code (`src/applications` vs. `src/platform`). Frontend teams are working to disentangle some cross-application dependencies that have crept in.
* Design System
* Frontend build process does a bunch of stuff to build/test/webpack. Content-build is separate step. Upshot of frontend deployment is that all frontend assets (HTML/JS/CSS/images) land in an S3 bucket and are served from there.
* [TK] Frontend request flow diagram

Backend details

* Rails APIs live in[vets-api](https://github.com/department-of-veterans-affairs/vets-api/)
* Application vs. platform ownership is a lot less clear vs. the frontend.
  * Typically for a given application there are one or more application-specific API endpoints, built as Rails controllers and models. They implement some of the application business logic and do things like request validation and orchestration. These are built and maintained by the appropriate VFS application team and are typically built hand in hand with that team's corresponding frontend code.
  * Then when those APIs need to interact with an upstream VA system, there might be a service integration module that handles the details of interacting with that system. These are sometimes built by VFS teams, but are frequently shared across teams. And "proper" ownership is unclear.
  * There's also common functionality like authentication, error handling, etc, that is logically owned by the platform. But VFS teams will sometimes contribute code to these libraries.
* [TK] Backend request flow diagram

Current VA.gov-MHV Intersection Points

Links

There's a high degree of linking between VA.gov and MHV - both in static content and in some navigation elements (like the "My Health" link in the top nav).
Linking between web properties is a-okay. However:
* The MHV approach to implementing SSO (described below) means that linking is more difficult.
* Because the teams are working in relative isolation from one another, there's more potential for broken links.

API/Feature Integrations

MHV is one of the many upstream systems that VA.gov is already integrated with.
* The VA.gov dashboard fetches summary information about unread secure messages and available prescription refills using MHV APIs.
* The flagship mobile app uses MHV APIs to power its secure messaging feature.

There are a couple of historical integrations that are not currently used:
* Vets.gov used to have full user-facing implementations of Secure messaging, Rx Refill, and Health records, all powered by MHV APIs. These were mothballed during the Vets.gov - VA.gov transition.
* There is a set of MHV account creation APIs. These were used to transparently provision an MHV account for eligible users to let them access the above features. These were removed in favor of simply redirecting users to MHV and letting account provisioning happen there.

Design System

The VA design system is intended for use across public-facing VA properties. That said, it originated in the VA.gov ecosystem and is managed by the VA.gov platform team. MHV has adopted it to some degree but generally there's a delta between the two sites' designs.

Single Sign On

VA.gov and MHV have implemented single sign on capability. They use a service called SSOe, provided by the VA Identity and Access Management (IAM) team.

The two sites have implemented SSOe sign-on using different approaches:
* VA.gov has a SAML integration, so during authentication the user is redirected to SSOe and then to a federated credential provider, but then eventually lands back at VA.gov in a signed in state. SSOe establishes a session cookie, and VA.gov uses that to establish its own API session cookie .
* MHV uses an SSOe proxy service - meaning that once a user signs in using SSOe, all traffic is proxied through an SSOe-owned subdomain. You'll notice this after signing in, that the base URL has switched to from `myhealth.va.gov` to `eauth.va.gov/mhv-portal-web/`
  * The upshot for the MHV team is that they don't have to do any session management, and they receive information about the authenticated user as HTTP headers injected by that SSOe proxy server.
  * But it means that linking is brittle - in many cases a link to a specific MHV page will not work via that proxy URL, and will land the user at the MVH home page.

  * Frequently, links even within MHV break when the user is logged in, because of the proxied domain.

* Single sign on technically works, in that a user who navigates from one site to another arrives in a logged in state.
* Accomplishing that auto-login behavior requires at minimum one redirect round trip to an SSOe domain, which is annoying for users.
* Session termination is kludgy because of the different approaches used by the two sites.

Credentials

As mentioned above, VA uses a number of federated credentials across its web properties. One of the federated options is an MHV credential.
* Originally this was the means to log in to the MHV site - MHV acted as its own sole credential provider.
* Later support was added to use those MHV credentials on VA.gov, so MHV has a role as both a website and a federated credential provider.
* MHV's credential has a number of wrinkles - there are different account tiers with varying levels of access to features on both sites - these correspond roughly but not exactly to NIST level of assurance.
* MHV has decided to deprecate its own credential because it does not meet current NIST guidance and would require too much investment to get it up to par.
  * Timeline for deprecation/retirement is unknown.

How Cerner fits in

* Cerner's Electronic Health Record (EHR) system is slated to replace VistA, VA's home-grown EHR.
* VA is rolling out Cerner medical center-by-medical center.
  * This means that any system that displays health information to a user needs to be able to pull data from both EHR systems, or else be capable of rerouting users to a different view.
  * Note that veterans frequently receive care at multiple medical centers and so may have information about ongoing care located in both systems.
* Most of MHV's features are integrated specifically with VistA
  * Health records for sure, also Rx Refill.
  * Secure messaging is believed to be a relatively standalone capability but does have some connection to VistA for user provisioning and management.
  * Appointments, aka VA Online Scheduling (VAOS) may already have a means to provide a unified view of Cerner + VistA data. At the least, a proposal for such a design has been discussed.

* Rather than having user-facing health portals tied to EHR systems (as we have today), build one portal that is EHR-independent.
  * Likely means a backend service is capable of brokering between the EHR systems, insulating the portal from that complexity.
  * See Cerner FHIR API references and analysis
  * Learn more about MHV's VistA integration

Possible Future States

Platform flux

The VA.gov platform is currently working toward providing more self-service capabilities:
* Separating the application and platform code in separate repositories and separate runtimes.
* Isolating each frontend and backend application into its own repository.
* Allowing autonomous deployment of individual applications on their own release cadence instead of an all-or-nothing daily deploy.
* Allowing teams to access their own observability information like application metrics and logs
* Additional development and operations tasks

All of the above is intended to be surfaced via a platform developer console with a unified login mechanism.

The exact timing of these capabilities being available is not known, so a key thing to determine is whether there will be a race condition between the MHV teams needing these capabilities to have an effective workflow, and them being generally available for use.

MHV-on-VA.gov variations

Frontend migration only

It's feasible to build the relevant frontend applications on VA.gov, and power them using the same APIs that power MHV's frontend today.
* We know this is generally feasible - in fact MVP versions of several of these features were live on Vets.gov in 2017.
* Provides some immediate benefits
  * Single sign on/authentication delta is no longer a problem.
  * MHV features written as React apps become more consistent with VA.gov design, content, IA standards.

    * Linking gets easier
* There's likely some impedance mismatch between the APIs and intended designs that would need resolving.
* Lowest lift, but leaves MHV team in a fundamentally different position relative to the platform compared to other teams - different workflow, different backend architecture, different monitoring needs.
* How does API authentication work across two disparate backend stacks?
    * Generally we've solved this with using the vets-api facade model, but that is potentially a lot of proxy code to write and maintain.

Backend lift and shift

Take MHV's current backend applications and migrate them into the VA.gov platform hosting environment.
* Platform goal is to support autonomous backend deployments and "bring your own stack". But this is quite a ways off from being a reality.
    * Meantime, current collaboration cycle and other processes are not set up for it. Situation will be "all other VFS teams" and "MHV team" as platform customers. Do we care?
* Believe MHV backend is running in some kind of Java application server. How well does that fit in with Platform devops practices?
*

Backend modernization

Consider rewriting any MHV backend services that warrant it.
* Avoid building into current vets-api monolith (since it's being decomposed as we speak), but instead refactor to head in the intended direction of VA.gov platform.
* This shouldn't be done only for the sake of language/platform consistency.
* Future-looking use cases, including but not limited to the requirements around Cerner integration, should drive this.
* Risks/costs of rewriting a significant backend application should be appreciated.

Aspirational ideas

* Further linking between secure messaging and appointment or rx refill - e.g. if a prescription is mentioned in a secure message, make it a clickable link to prescription refill /[Discussed during mobile secure messaging discovery]/

* Provide an API for other tools to interact with secure messaging. Use case - allow provider to send after visit summary (AVS) with a single click from AVS tool, rather than having to print to PDF, open secure messaging, and attach a file. /[This was suggested at one point by Dr. Evans]/

Implications for platform capabilities

[TODO] What do we anticipate the initial needs of the MHV development teams will be, relative to the current and future platform capabilities.

Single sign on deep dive

[TODO]

Credential deep dive

[TODO]

Open Questions

* How coupled/monolithic is the MHV application (frontend and backend?)
* Establish SWAGs for the relative timing of:
  * VA.gov platform being capable of autonomous frontend deployment
  * VA.gov platform being capable of autonomous backend deployment
  * MHV credential being retired
  * Cerner API availability
  * Phased migration of MHV features onto VA.gov