Play	Checklist	Action
Play 1 Understand what people need	1. Early in the project, spend time with current and prospective users of the service	We identified key groups of potential end users and interviewed several users before design and development began.
	2. Use a range of qualitative and quantitative research methods to determine people's goals, needs, and behaviors; be thoughtful about the time spent	We created a user interview guide focused on understanding user' top tasks and overall needs with regards to drug recall information.
	3. Test prototypes of solutions with real people, in the field if possible	We conducted multiple rounds of task-based usability testing with representative end users of our service.
	4. Document the findings about user goals, needs, behaviors, and preferences	Based on data collected during user interviews, we created a primary persona, which documented user goals, needs, behaviors, and preferences.
	5. Share findings with the team and agency leadership	All user research and usability findings were presented to the Scrum team. Without agency leadership, we also presented to our Product Manager, who represented the agency customer's needs.
	6. Create a prioritized list of tasks the user is trying to accomplish, also known as "user stories"	We created Epics and User Stories in JIRA to document user tasks and drive all design and development.
	7. As the digital service is being built, regularly test it with potential users to ensure it meets people's needs	We conducted multiple rounds of task-based usability testing with representative end users of our service.
Play 2 Address the whole experience, from start to finish	1. Understand the different points at which people will interact with the service – both online and in person	All users will interact with our service online. We ensured that our service works and is usable on all major platforms (desktop, tablet, and mobile.)
	2. Identify pain points in the current way users interact with the service, and prioritize these according to user needs	Tools that are available on the open market today are often complicated and will not work for a mobile experience. Our users wanted a simple "Google-like" experience that would work "in the checkout line at CVS."
	3. Design the digital parts of the service so that they are integrated with the offline touch points people use to interact with the service	Our service works and is usable on all major platforms (desktop, tablet, and mobile.)
	4. Develop metrics that will measure how well the service is meeting user needs at each step of the service	We have a page-level user feedback form User Story in our Backlog.
Play 3 Make it simple and intuitive	1. Create or use an existing, simple, and flexible design style guide for the service	We created a style guide to maintain consistency for current and future design work.
	2. Use the design style guide consistently for related digital services	We created a style guide to maintain consistency for current and future design work.
	3. Give users clear information about where they are in each step of the process	We wrote page-level text and instructions using "plain-language" best practices.
	4. Follow accessibility best practices to ensure all people can use the service	Our Visual Designer and Front-End Developer are both well-versed in accessible design for Federal Websites. They collaborated early and often to ensure designs and code are Section 508 compliant and fully accessible. This was validated during QA testing using WAVE.
	5. Provide users with a way to exit and return later to complete the process	Does not apply to our service.
	6. Use language that is familiar to the user and easy to understand	We wrote page-level text and instructions using "plain-language" best practices.
	7. Use language and design consistently throughout the service, including online and offline touch points	We wrote page-level text and instructions using "plain-language" best practices. Our Content Strategist ensured consistency across the service.
Play 4 Build the service using agile and iterative practices	1. Ship a functioning "minimum viable product" (MVP) that solves a core user need as soon as possible, no longer than three months from the beginning of the project, using a "beta" or "test" period if needed	We shipped our Minimum Viable Product (MVP) in 6 days and collected user feedback by conducting usability testing on the MVP (Sprint 1.)
	2. Run usability tests frequently to see how well the service works and identify improvements that should be made	We ran 2 sets of formal usability test with representative users. Test #1 was to get initial feedback on the MVP (Sprint 1) and guide enhancements for Sprint 2. Test #2 was to validate usability enhancements made in Sprint 2.
	3. Ensure the individuals building the service communicate closely using techniques such as launch meetings, war rooms, daily standups, and team chat tools	We met for daily 15-minute stand-up Scrums at 9 a.m., followed by a 1 p.m. open meeting for problem solving. We used collaborative tools such as Jira and Skype for Business and had a shared conference for ad hoc meetings.
	4. Keep delivery teams small and focused; limit organizational layers that separate these teams from the business owners	We had a flat organizational structure with a small and focused multi-functional team reporting up to one Product Manager.
	5. Release features and improvements multiple times each month	We had 2 short sprints with a major release at the end of each that included new features and functionality.
	6. Create a prioritized list of features and bugs, also known as the "feature backlog" and "bug backlog"	We maintained our backlog in Jira and designated issues as either bugs, usability recommendations, enhancements, or features.
	7. Use a source code version control system	We used GitHub for source code version control.
	8. Give the entire project team access to the issue tracker and version control system	Our entire team had access to Jira and GitHub.
	9. Use code reviews to ensure quality	We conducted code reviews for each sprint concurrent with functional testing.
Play 5 Structure budgets and contracts to support delivery	1. Budget includes research, discovery, and prototyping activities	As part of our budget process, our Business Analyst and Technical Architect worked with the Product Manager to develop a comprehensive estimate to complete all phases of the prototype. At the end of each sprint we assessed and adjusted the budget as needed.
	2. Contract is structured to request frequent deliverables, not multi-month milestones	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	3. Contract is structured to hold vendors accountable to deliverables	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	4. Contract gives the government delivery team enough flexibility to adjust feature prioritization and delivery schedule as the project evolves	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	5. Contract ensures open source solutions are evaluated when technology choices are made	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	6. Contract specifies that software and data generated by third parties remains under our control, and can be reused and released to the public as appropriate and in accordance with the law	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	7. Contract allows us to use tools, services, and hosting from vendors with a variety of pricing models, including fixed fees and variable models like "pay-for-what-you-use" services	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	8. Contract specifies a warranty period where defects uncovered by the public are addressed by the vendor at no additional cost to the government	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
	9. Contract includes a transition of services period and transition-out plan	Aquilent has experience providing Agile services to the Government and will work with GSA to establish a flexible contract that aligns with Agile methodology.
Play 6 Assign one leader and hold that person accountable	1. A product owner has been identified	Aquilent established a Product Owner/Manager at the beginning of the project. The Product Owner has full authority to operate on behalf the company, has accountability for the end product, and has a strong relationship with GSA 18F.
	2. All stakeholders agree that the product owner has the authority to assign tasks and make decisions about features and technical implementation details	There was complete consensus among all stakeholders that the Product Owner/Manager has full authority to manage the team, assigning tasks and making final decision on product features.
	3. The product owner has a product management background with technical experience to assess alternatives and weigh tradeoffs	Our Product Owner/Manager has extensive expertise leading the development of innovative technical solutions for the Government.
	4. The product owner has a work plan that includes budget estimates and identifies funding sources	The Product Manager established a budget and received corporate approval for the budget. Throughout the process he tracked status against the budget and made adjustments as necessary.
	5. The product owner has a strong relationship with the contracting officer	Aquilent's Product Owner will establish and maintain a positive relationship with the Contracting Officer for the BPA.
Play 7 Bring in experienced teams	1. Member(s) of the team have experience building popular, high-traffic digital services	Our team is made up of experienced professionals that have worked on high-traffic Websites such as Healthcare.gov and USPS.com.
	2. Member(s) of the team have experience designing mobile and web applications	Our team has experience designing and developing mobile and web applications to support citizens. Examples of digital applications include Healthcare.gov, USPS.com, SeaPort-e, VA's Healthy Living Assessment (HLA), and MyHealthe Vet.
	3. Member(s) of the team have experience using automated testing frameworks	Our team has experience using automated testing tools such as Selenium and PHPUnit.
	4. Member(s) of the team have experience with modern development and operations (DevOps) techniques like continuous integration and continuous deployment	Our team has experience with modern DevOps techniques across multiple environments using tools such as GitHub, Jenkins, Selenium, Chef, CruiseControl and AWS tools (AWS CloudWatch, Auto Scaling, and CloudFormation)
	5. Member(s) of the team have experience securing digital services	All of Aquilent's developers have experience securing digital services. We follow Federal Government security guidelines on all projects and members of our team have worked on applications that range from FISMA Low to FISMA High. For example, VA's Healthy Living Assessment project includes personally identifiable information with FISMA High requirements.

	6. A Federal contracting officer is on the internal team if a third party will be used for development work	Not applicable to this effort.
	7. A Federal budget officer is on the internal team or is a partner	Not applicable to this effort.
	8. The appropriate privacy, civil liberties, and/or legal advisor for the department or agency is a partner	Not applicable to this effort.
Play 8 Choose a modern technology stack	1. Choose software frameworks that are commonly used by private-sector companies creating similar services	For this prototype we used Bootstrap and Laravel software frameworks.
	2. Whenever possible, ensure that software can be deployed on a variety of commodity hardware types	Because we used Docker container our product can be deployed on a variety of commodity hardware types.
	3. Ensure that each project has clear, understandable instructions for setting up a local development environment, and that team members can be quickly added or removed from projects	Our instructions documented how to set up a development environment using GitHub clone.
	4. Consider open source software solutions at every layer of the stack	We chose proven, open source languages and tools such as PHP, Laravel, Twitter Bootstrap, JQuery, Google Fonts, Linux, Docker, Jenkins, Vagrant, VirtualBox, and Chef .
Play 9 Deploy in a flexible hosting environment	1. Resources are provisioned on demand	We use CloudFormation to provision resources on demand in AWS.
	2. Resources scale based on real-time user demand	We use AWS Auto Scaling to scale resources in real time based on load and user demand.
	3. Resources are provisioned through an API	We use CloudFormation to provision resources in AWS.
	4. Resources are available in multiple regions	For this prototype, we used AWS East only. Using a load balancer, we can easily switch to multiple regions (or multiple availability zones) if required.
	5. We only pay for resources we use	We hosted our service in the Cloud (AWS) and only pay for what is used.
	6. Static assets are served through a content delivery network	Our service is very lightweight with regards to static assets, but would leverage a CDN such as Akamai or AWS CloudFront if needed.
	7. Application is hosted on commodity hardware	We host our service (application) in the Cloud (AWS).
Play 10 Automate testing and deployments	1. Create automated tests that verify all user-facing functionality	We created a suite of automated test scripts in Selenium.
	2. Create unit and integration tests to verify modules and components	We developed unit tests in PHPUnit to automatically run in Jenkins. For integration testing we created Selenium tests using the Firefox Selenium IDE plugin and exported them to Python Unit Tests that were run through Jenkins.
	3. Run tests automatically as part of the build process	We exported test scripts as Python scripts into our GitHub repository. They were scheduled to run headless automatically when a build was deployed to each of the environments. The automated scripts were also set up to run on demand or by using the IDE built in to Firefox.
	4. Perform deployments automatically with deployment scripts, continuous delivery services, or similar techniques	We used Jenkins to trigger download checkout from GitHub and deploy to the various environments.
	5. Conduct load and performance tests at regular intervals, including before public launch	Not applicable to our service at this time.
Play 11 Manage security and privacy through reusable processes	1. Contact the appropriate privacy or legal officer of the department or agency to determine whether a System of Records Notice (SORN), Privacy Impact Assessment, or other review should be conducted	Not applicable to our service.
	2. Determine, in consultation with a records officer, what data is collected and why, how it is used or shared, how it is stored and secured, and how long it is kept	Not applicable to our service.
	3. Determine, in consultation with a privacy specialist, whether and how users are notified about how personal information is collected and used, including whether a privacy policy is needed and where it should appear, and how users will be notified in the event of a security breach	We have a User Story in our Backlog to implement a Privacy Policy on the service.
	4. Consider whether the user should be able to access, delete, or remove their information from the service	Not applicable to our service.
	5. “Pre-certify” the hosting infrastructure used for the project using FedRAMP	We used AWS East to host our service, which is FedRAMP compliant.
	6. Use deployment scripts to ensure configuration of production environment remains consistent and controllable	We wrote deployment scripts, which are used by Jenkins and are part of the Chef provisioning process.
Play 12 Use data to drive decisions	1. Monitor system-level resource utilization in real time	We used AWS CloudWatch Detail Monitoring to monitor system-level resource utilization in real time.
	2. Monitor system performance in real-time (e.g. response time, latency, throughput, and error rates)	We used AWS CloudWatch Detail Monitoring to monitor system performance utilization in real time.
	3. Ensure monitoring can measure median, 95th percentile, and 98th percentile performance	We used AWS CloudWatch Detail Monitoring, which can measure median, 95th percentile, and 98th percentile performance.
	4. Create automated alerts based on this monitoring	We set up automated monitoring alerts in AWS CloudWatch Monitoring.
	5. Track concurrent users in real-time, and monitor user behaviors in the aggregate to determine how well the service meets user needs	We use AWS CloudWatch Detail Monitoring to collect user data such as timestamp, IP address, and URL. We will use this to ensure the service meets user needs.
	6. Publish metrics internally	We run metrics reports internally from AWS CloudWatch Monitoring and publish to the team.
	7. Publish metrics externally	We will integrate with GSA DAP and openly publish metrics.
	8. Use an experimentation tool that supports multivariate testing in production	We used GitHub which allows for branching of A/B versions of the application.
Play 13 Default to open	1. Offer users a mechanism to report bugs and issues, and be responsive to these reports	We use JIRA to log, document, track, and close all bugs and issues.
	2. Provide datasets to the public, in their entirety, through bulk downloads and APIs (application programming interfaces)	Not applicable to our service.
	3. Ensure that data from the service is explicitly in the public domain, and that rights are waived globally via an international public domain dedication, such as the “Creative Commons Zero” waiver	Dataset is from openFDA and is explicitly in the public domain.
	4. Catalog data in the agency’s enterprise data inventory and add any public datasets to the agency’s public data listing	Not applicable to our service.
	5. Ensure that we maintain the rights to all data developed by third parties in a manner that is releasable and reusable at no cost to the public	Dataset is from openFDA and is explicitly in the public domain. The Government maintains all rights to the dataset.
	6. Ensure that we maintain contractual rights to all custom software developed by third parties in a manner that is publishable and reusable at no cost	All software we used for this service is open source and is publishable and reusable at no cost.
	7. When appropriate, create an API for third parties and internal users to interact with the service directly	Not applicable to our service.
	8. When appropriate, publish source code of projects or components online	Our source code for this service is published openly on GitHub.
	9. When appropriate, share your development process and progress publicly	Our development process and progress on this service is published openly on GitHub.