Service Plays Table

The following table describes our approach for checklist items that support each play in the *US Digital Service: Digital Services Playbook.*

Play Number	Name	Checklist Item	Our Approach
1	**Understand what people need**	Early in the project, spend time with current and prospective users of the service	We interviewed users via role play to improve our understanding and identify needs of users.
		Use a range of qualitative and quantitative research methods to determine people's goals, needs, and behaviors; be thoughtful about the time spent	We collated needs of users and gauged their perspectives by selecting and interviewing users with experience in purchasing hardware and software.
		Test prototypes of solutions with real people, in the field if possible	We tested our prototype with e-commerce subject matter experts including authorized users and authorized administrators.
		Document the findings about user goals, needs, behaviors, and preferences	We documented the requirements we gathered in the user stories. Our focus was to develop an application that makes acquiring hardware, software, and services from preselected vendors easy and user friendly.
		Share findings with the team and agency leadership	Since the agency leadership was not available for conversations, we conducted weekly meetings and worked with experienced leaders to review our work, as well as gather valuable inputs.
		Create a prioritized list of tasks the user is trying to accomplish, also known as "user stories"	Based on information gathered during interviews and role playing, a list of user tasks and objectives were created and prioritized. This list was then refined into user stories and entered into the product backlog.
		As the digital service is being built, regularly test it with potential users to ensure it meets people's needs	We involved all stakeholders in a series of testing sessions starting from substantiating the concepts to comprehensive user testing.
2	**Address the whole experience, from start to finish**	Understand the different points at which people will interact with the service – both online and in person	We gained understanding of the points of interaction that a user has with the service by engaging users, including subject matter experts, to understand all possibilities.
		Identify pain points in the current way users interact with the service, and prioritize these according to user needs	We identified the current issue as being the absence of a centralized system with nothing to enforce use of pre-selected vendors. This puts the State at risk of missing high volume discounts and leads to inconsistent product purchases across teams and departments.

Play Number	Name	Checklist Item	Our Approach
		Design the digital parts of the service so that they are integrated with the offline touch points people use to interact with the service	We identified offline and online moments: • Offline - Users use printed vendor catalogs to search for and compare items. • Online - Users use search engines to search for and compare items. Users may also use vendor websites to order (e.g., Amazon, HP.com, NewEgg.com, etc.)
		Develop metrics that will measure how well the service is meeting user needs at each step of the service	We identified our metrics based on usability testing and in our user centric design. We chose the following metrics: • Ease in navigation by minimizing the number of steps to accomplish defined tasks. • Consolidation of product information into categories to allow the user to more quickly focus on a desired purchase. • Allow feature comparison of three or less products.
3	**Make it simple and intuitive**	Use a simple and flexible design style guide for the service. Use the U.S. Web Design Standards as a default	We adopted rules from the U.S. Web Standard UI component library and strictly followed brand guidelines to define color, typography, form fields, spacing and click-through. The look and feel of the pages in our application is innovative, unique and simple.
		Use the design style guide consistently for related digital services	We implemented the style guidelines and applied our standards in a uniform manner throughout the application.
		Give users clear information about where they are in each step of the process	We included a breadcrumb trail to help users understand their location within the application, as well as a graphical order status indicator to provide a visual of user progress through steps of the order process.
		Follow accessibility best practices to ensure all people can use the service	We diligently followed the World Wide Web Consortium's Web Content Accessibility Guidelines (W3C WCAG) as well as the Americans with Disabilities Act (ADA) section 508 standards and guidelines for accessibility.
		Provide users with a way to exit and return later to complete the process	Our design allows users to log in and out at their discretion. The logout option appears on every screen.
		Use language that is familiar to the user and easy to understand	Content and language within the application exhibits a true understanding of the intended users and is written in concise plain language.
		Use language and design consistently throughout the service, including online and offline touch points	Content on pages and panels in the application is consistent, uniform, and straightforward.

Play Number	Name	Checklist Item	Our Approach
		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	Upon receipt of the RFI, our team began working with the product manager to initiate Sprint 0 with the goal of defining the MVP. The team was challenged by the timeline and scope of creating the prototype. Despite the challenges, we were able to prioritize and revise user stories to support an MVP product on an initial release. Additional features were moved to a backlog for future sprint iterations.
		Run usability tests frequently to see how well the service works and identify improvements that should be made	We tested our design concept basic wireframes, as well as HTML mock-ups for usability to provide the best features and functionality to perspective users.
		Ensure the individuals building the service communicate closely using techniques such as launch meetings, war rooms, daily standups, and team chat tools	We held daily stand-up meetings and created a plan to facilitate constant communication to the entire team. Our daily stand-up meetings were a maximum duration of 15 minutes. These sessions helped us remove impediments to development and promoted quick decision making as we all had a clear understanding of the project’s status. We monitored and controlled sprint backlogs using JIRA burndown charts and a Scrum Board.
4	**Build the service using agile and iterative practices**	Keep delivery teams small and focused; limit organizational layers that separate these teams from the business owners	We formed an experienced scrum team of 13 people and collaborated with others when needed.
		Release features and improvements multiple times each month	We operated in a high-speed iterative development mode to facilitate delivery of improvements within weekly sprints.
		Create a prioritized list of features and bugs, also known as the “feature backlog” and “bug backlog”	We created epic stories during our design and definition phases from which user stories were developed. With this information, we used JIRA to build our feature backlog and bug backlog.
		Use a source code version control system	We leveraged GitHub's version control and the standard GitFlow model for all source code version control during development.
		Give the entire project team access to the issue tracker and version control system	For issue tracking and version control, team members were given access to JIRA, Apache Subversion (SVN), Microsoft SharePoint, and GitHub.
		Use code reviews to ensure quality	We conduct reviews of each pull request and spot-checked our code during development. Each team worked collaboratively throughout the project to ensure architectural and technical decisions were understood, reviewed, and clearly communicated.
5	**Structure budgets and contracts to support delivery**	Budget includes research, discovery, and prototyping activities	We set up a budget covering all project activities and tracked to it to ensure that we maintained a lean approach.

Play Number	Name	Checklist Item	Our Approach
		Contract is structured to request frequent deliverables, not multi-month milestones	For this effort, we estimated and planned sprints based on short term iterations.
		Contract is structured to hold vendors accountable to deliverables	As requested by the State, we worked to deliver our vendor profile and prototype, along with the required README.md file on time.
		Contract gives the government delivery team enough flexibility to adjust feature prioritization and delivery schedule as the project evolves	We accounted for time to review and reprioritize or add user stories between sprints. This allowed us to respond to user feedback, testing outcomes, and burn rate evaluation.
		Contract ensures open source solutions are evaluated when technology choices are made	We gave preference to use of open source tools throughout our design and development process.
		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	All code developed for this prototype was released as open source on GitHub.
		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	We used a variety of models, including pay-for-what-you-use hosting from Azure, and free open source licensed tools.
		Contract specifies a warranty period where defects uncovered by the public are addressed by the vendor at no additional cost to the government	N/A
		Contract includes a transition of services period and transition-out plan	N/A
6	**Assign one leader and hold that person accountable**	A product owner has been identified	Derek was selected as the product manager with authority and responsibility for providing the business case.
		All stakeholders agree that the product owner has the authority to assign tasks and make decisions about features and technical implementation details	During Sprint 0, the team agreed that Derek would be the single owner with final authority on all aspects of the prototype development.
		The product owner has a product management background with technical experience to assess alternatives and weigh tradeoffs	Derek has many years of experience managing large and small scale IT deliveries. He has also lead technical development teams.
		The product owner has a work plan that includes budget estimates and identifies funding sources	Our work plan and budget were set during Sprint 0. The plan and budget were tracked throughout the prototype development.
		The product owner has a strong relationship with the contracting officer	We coordinated with the State’s procurement office as needed and when permissible.
7	**Bring in experienced teams**	Member(s) of the team have experience building popular, high-traffic digital services	Our development team had extensive experience in high-end digital services. For example, Ashvin has nine years of experience in this area.

Play Number	Name	Checklist Item	Our Approach
		Member(s) of the team have experience designing mobile and web applications	Our design and development team has significant experience in the web-application. For example, Ashvin also has significant experience in this area.
		Member(s) of the team have experience using automated testing frameworks	Our projects used CI technologies, such as Jenkins, and standard unit testing libraries, like Jacoco and Junit, to conduct automated testing in addition to our manual test plans.
		Member(s) of the team have experience with modern development and operations (DevOps) techniques like continuous integration and continuous deployment	We developed using docker for production and configuration, Jenkins for deployment, and HPE SiteScope and Azure for monitoring.
		Member(s) of the team have experience securing digital services	Our development team has been involved with secured digital services while supporting projects for client in the health and human services, banking and financial, and telecommunication industries.
		A Federal contracting officer is on the internal team if a third party will be used for development work	N/A
		A Federal budget officer is on the internal team or is a partner	N/A
		The appropriate privacy, civil liberties, and/or legal advisor for the department or agency is a partner	We reviewed the prototype and process with our legal team to ensure there were no privacy, civil liberty, or other legal issues.
8	**Choose a modern technology stack**	Choose software frameworks that are commonly used by private-sector companies creating similar services	We leveraged knowledge from our experience with existing clients in the public and private sectors to guide our technology choices.
		Whenever possible, ensure that software can be deployed on a variety of commodity hardware types	All deployments were targeted at Red Hat Enterprise Linux (RHEL) running on virtualized hardware.
		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	We used a combination of docker files to ensure that developers and QA team members could quickly replicate development environments on their systems.
		Consider open source software solutions at every layer of the stack	All of our tools at every level of the stack are open source.
9	**Deploy in flexible hosting environment**	Resources are provisioned on demand	All resource are provisioned on demand.
		Resources scale based on real-time user demand	Our design allows for few to many users at any time.
		Resources are provisioned through an API	We used a RESTful API to source data displayed within the application.
		Resources are available in multiple regions	Resources are available in multiple regions.
		We only pay for resources we use	Our design's use of Azure allows expansion and payment of resources as necessary.

Play Number	Name	Checklist Item	Our Approach
		Static assets are served through a content delivery network	Our use of Azure gives us the ability to leverage their Cloud Delivery Network to move content closer to users.
		Application is hosted on commodity hardware	Our solution is virtualized, so is compatible and can function on a plug and play basis on a wide variety of hardware.
10	**Automate testing and deployments**	Create automated tests that verify all user-facing functionality	We automated our functionality tests with Selenium.
		Create unit and integration tests to verify modules and components	We built unit tests into the server code.
		Run tests automatically as part of the build process	Unit tests were run during the build process with test results captured through Jacoco.
		Perform deployments automatically with deployment scripts, continuous delivery services, or similar techniques	Deployments were automatically created by Jenkins and Ansible. These artifacts were called from Nexus and deployment was performed in two steps—1) build a new image of the .war file which is run and uploaded in Nexus; 2) Ansible retrieves the image from the docker registry which is deployed on targeted nodes.
		Conduct load and performance tests at regular intervals, including before public launch	We used Jmeter to conduct load and performance tests at regular intervals.
11	**Manage security and privacy through reusable processes**	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	We conducted a review of our plans and final prototype with our Legal and Privacy team. Since the source of the data was provided by the State, there were no legal or privacy actions identified.
		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	We only collected and stored test data entered by the user. This data displayed only during the user's active session.
		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	Based on our legal and privacy review, no privacy policy was identified as being needed for this prototype.
		Consider whether the user should be able to access, delete, or remove their information from the service	For purposes of the prototype, users are not permitted to access, delete, or remove their information. Users are also not permitted to see other user's data.
		"Pre-certify" the hosting infrastructure used for the project using FedRAMP	Enterprise Services is an Azure consulting partner in the GovCloud region. Azure is the preferred public cloud partner for Enterprise Services.
		Use deployment scripts to ensure configuration of production environment remains consistent and controllable	Docker files and Azure resources were used to ensure consistency and control.
12	**Use data to drive decisions**	Monitor system-level resource utilization in real time	We monitor resource utilization of the system in real-time mode through SiteScope.

Play Number	Name	Checklist Item	Our Approach
		Monitor system performance in real-time (e.g. response time, latency, throughput, and error rates)	We use SiteScope to collect different types of performance and availability information to monitor the system.
		Ensure monitoring can measure median, 95th percentile, and 98th percentile performance	N/A
		Create automated alerts based on this monitoring	We used SiteScope to configure an alert to automatically trigger an email message informing team members of the system's status.
		Track concurrent users in real-time, and monitor user behaviors in the aggregate to determine how well the service meets user needs	We used SiteScope to manage end user status information requests, configure change requests, and access control.
		Publish metrics internally	We shared collected metrics within our development team.
		Publish metrics externally	Metrics we collect are available to authorized parties upon request.
		Use an experimentation tool that supports multivariate testing in production	N/A
13	**Default to open**	Offer users a mechanism to report bugs and issues, and be responsive to these reports	We assumed that users would report bugs and issues using existing State procedures.
		Provide datasets to the public, in their entirety, through bulk downloads and APIs (application programming interfaces)	The data we used was provided by the State in the RFI.
		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	The non-test data used for the service was from a source provided by the State and originated in the public domain.
		Catalog data in the agency's enterprise data inventory and add any public datasets to the agency's public data listing	We added no data to any agency's enterprise data inventory.
		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	We used only the free data provided by the State for this prototype.
		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 the code developed for this prototype will be released as open source on GitHub.
		When appropriate, create an API for third parties and internal users to interact with the service directly	We used only data provided by the State in the RFI resource file.
		When appropriate, publish source code of projects or components online	All code developed for this prototype will be released as open source on GitHub.
		When appropriate, share your development process and progress publicly	Artifacts outlining our development process and progress will be deployed to our public GitHub repository.