U.S. DIGITAL SERVICE

Digital Services Playbook

The American people expect to interact with government through digital channels such as websites, email, and mobile applications. By building digital services that meet their needs, we can make the delivery of our policy and programs more effective.

Today, too many of our digital services projects do not work well, are delivered late, or are over budget. To increase the success rate of these projects, the U.S. Government needs a new approach. We created a playbook of 13 key "plays" drawn from successful practices from the private sector and government that, if followed together, will help government build effective digital services.

See the plays **Help improve this content**

Digital Service Plays

1. Understand what people need
2. Address the whole experience, from start to finish
3. Make it simple and intuitive
4. Build the service using agile and iterative practices
5. Structure budgets and contracts to support delivery
6. Assign one leader and hold that person accountable
7. Bring in experienced teams
8. Choose a modern technology stack
9. Deploy in a flexible hosting environment
10. Automate testing and deployments
11. Manage security and privacy through reusable processes
12. Use data to drive decisions
13. Default to open

In detail

PLAY 1

Understand what people need

We must begin digital projects by exploring and pinpointing the needs of the people who will use the service, and the ways the service will fit into their lives. Whether the users are members of the public or government employees, policy makers must include real people in their design process from the beginning. The needs of people — not constraints of government structures or silos — should inform technical and design decisions. We need to continually test the products we build with real people to keep us honest about what is important.

Checklist

- ☐ Early in the project, spend time with current and prospective users of the service
- ☐ Use a range of qualitative and quantitative research methods to determine people’s goals, needs, and behaviors; be thoughtful about the time spent
- ☐ Test prototypes of solutions with real people, in the field if possible
- ☐ Document the findings about user goals, needs, behaviors, and preferences
- ☐ Share findings with the team and agency leadership
- ☐ Create a prioritized list of tasks the user is trying to accomplish, also known as “user stories”
- ☐ As the digital service is being built, regularly test it with potential users to ensure it meets people’s needs

Key Questions

- ○ Who are your primary users?
- ○ What user needs will this service address?
- ○ Why does the user want or need this service?
- ○ Which people will have the most difficulty with the service?
- ○ Which research methods were used?
- ○ What were the key findings?
- ○ How were the findings documented? Where can future team members access the documentation?
- ○ How often are you testing with real people?

PLAY 2

Address the whole experience, from start to finish

We need to understand the different ways people will interact with our services, including the actions they take online, through a mobile application, on a phone, or in person. Every encounter — whether it’s online or offline — should move the user closer towards their goal.

Checklist

- ☐ Understand the different points at which people will interact with the service – both online and in person
- ☐ Identify pain points in the current way users interact with the service, and prioritize these according to user needs
- ☐ Design the digital parts of the service so that they are integrated with the offline touch points people use to interact with the service
- ☐ Develop metrics that will measure how well the service is meeting user needs at each step of the service

Key Questions

- ○ What are the different ways (both online and offline) that people currently accomplish the task the digital service is designed to help with?
- ○ Where are user pain points in the current way people accomplish the task?
- ○ Where does this specific project fit into the larger way people currently obtain the service being offered?
- ○ What metrics will best indicate how well the service is working for its users?

PLAY 3

Make it simple and intuitive

Using a government service shouldn’t be stressful, confusing, or daunting. It’s our job to build services that are simple and intuitive enough that users succeed the first time, unaided.

Checklist

- ☐ Use a simple and flexible design style guide for the service. Use the U.S. Web Design System as a default
- ☐ Use the design style guide consistently for related digital services
- ☐ Give users clear information about where they are in each step of the process
- ☐ Follow accessibility best practices to ensure all people can use the service
- ☐ Provide users with a way to exit and return later to complete the process
- ☐ Use language that is familiar to the user and easy to understand
- ☐ Use language and design consistently throughout the service, including online and offline touch points

Key Questions

- ○ What primary tasks are the user trying to accomplish?
- ○ Is the language as plain and universal as possible?
- ○ What languages is your service offered in?
- ○ If a user needs help while using the service, how do they go about getting it?
- ○ How does the service’s design visually relate to other government services?

PLAY 4

Build the service using agile and iterative practices

We should use an incremental, fast-paced style of software development to reduce the risk of failure. We want to get working software into users’ hands as early as possible to give the design and development team opportunities to adjust based on user feedback about the service. A critical capability is being able to automatically test and deploy the service so that new features can be added often and be put into production easily.

Checklist

- ☐ 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
- ☐ Run usability tests frequently to see how well the service works and identify improvements that should be made
- ☐ Ensure the individuals building the service communicate closely using techniques such as launch meetings, strategy rooms, daily standups, and team chat tools
- ☐ Keep delivery teams small and focused; limit organizational layers that separate these teams from the business owners
- ☐ Release features and improvements multiple times each month
- ☐ Create a prioritized list of features and bugs, also known as the “feature backlog” and “bug backlog”
- ☐ Use a source code version control system
- ☐ Give the entire project team access to the issue tracker and version control system
- ☐ Use code reviews to ensure quality

Key Questions

○ How long did it take to ship the MVP? If it hasn’t shipped yet, when will it?

○ How long does it take for a production deployment?

○ How many days or weeks are in each iteration/sprint?

○ Which version control system is being used?

○ How are bugs tracked and tickets issued? What tool is used?

○ How is the feature backlog managed? What tool is used?

○ How often do you review and reprioritize the feature and bug backlog?

○ How do you collect user feedback during development? How is that feedback used to improve the service?

○ At each stage of usability testing, which gaps were identified in addressing user needs?

PLAY 5

Structure budgets and contracts to support delivery

To improve our chances of success when contracting out development work, we need to work with experienced budgeting and contracting officers. In cases where we use third parties to help build a service, a well-defined contract can facilitate good development practices like conducting a research and prototyping phase, refining product requirements as the service is built, evaluating open source alternatives, ensuring frequent delivery milestones, and allowing the flexibility to purchase cloud computing resources.

The TechFAR Handbook provides a detailed explanation of the flexibilities in the Federal Acquisition Regulation (FAR) that can help agencies implement this play.

Checklist

☐ Budget includes research, discovery, and prototyping activities

☐ Contract is structured to request frequent deliverables, not multi-month milestones

☐ Contract is structured to hold vendors accountable to deliverables

☐ Contract gives the government delivery team enough flexibility to adjust feature prioritization and delivery schedule as the project evolves

☐ Contract ensures open source solutions are evaluated when technology choices are made

☐ 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

☐ 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

☐ Contract specifies a warranty period where defects uncovered by the public are addressed by the vendor at no additional cost to the government

☐ Contract includes a transition of services period and transition-out plan

Key Questions

○ What is the scope of the project? What are the key deliverables?

○ What are the milestones? How frequent are they?

○ What are the performance metrics defined in the contract (e.g., response time, system uptime, time period to address priority issues)?

PLAY 6

Assign one leader and hold that person accountable

There must be a single product owner who has the authority and responsibility to assign tasks and work elements; make business, product, and technical decisions; and be accountable for the success or failure of the overall service. This product owner is ultimately responsible for how well the service meets the needs of its users, which is how a service should be evaluated. The product owner is responsible for ensuring that features are built and managing the feature and bug backlogs.

Checklist

- ☐ A product owner has been identified
- ☐ All stakeholders agree that the product owner has the authority to assign tasks and make decisions about features and technical implementation details
- ☐ The product owner has a product management background with technical experience to assess alternatives and weigh tradeoffs
- ☐ The product owner has a work plan that includes budget estimates and identifies funding sources
- ☐ The product owner has a strong relationship with the contracting officer

Key Questions

- ○ Who is the product owner?
- ○ What organizational changes have been made to ensure the product owner has sufficient authority over and support for the project?
- ○ What does it take for the product owner to add or remove a feature from the service?

PLAY 7

Bring in experienced teams

We need talented people working in government who have experience creating modern digital services. This includes bringing in seasoned product managers, engineers, and designers. When outside help is needed, our teams should work with contracting officers who understand how to evaluate third-party technical competency so our teams can be paired with contractors who are good at both building and delivering effective digital services. The makeup and experience requirements of the team will vary depending on the scope of the project.

Checklist

- ☐ Member(s) of the team have experience building popular, high-traffic digital services
- ☐ Member(s) of the team have experience designing mobile and web applications
- ☐ Member(s) of the team have experience using automated testing frameworks
- ☐ Member(s) of the team have experience with modern development and operations (DevOps) techniques like continuous integration and continuous deployment
- ☐ Member(s) of the team have experience securing digital services
- ☐ A Federal contracting officer is on the internal team if a third party will be used for development work
- ☐ A Federal budget officer is on the internal team or is a partner
- ☐ The appropriate privacy, civil liberties, and/or legal advisor for the department or agency is a partner

PLAY 8

Choose a modern technology stack

The technology decisions we make need to enable development teams to work efficiently and enable services to scale easily and cost-effectively. Our choices for hosting infrastructure, databases, software frameworks, programming languages and the rest of the technology stack should seek to avoid vendor lock-in and match what successful modern consumer and enterprise software companies would choose today. In particular, digital services teams should consider using open source, cloud-based, and commodity solutions across the technology stack, because of their widespread adoption and support by successful consumer and enterprise technology companies in the private sector.

Checklist

- ☐ Choose software frameworks that are commonly used by private-sector companies creating similar services
- ☐ Whenever possible, ensure that software can be deployed on a variety of commodity hardware types
- ☐ 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
- ☐ Consider open source software solutions at every layer of the stack

Key Questions

- ○ What is your development stack and why did you choose it?
- ○ Which databases are you using and why did you choose them?
- ○ How long does it take for a new team member to start contributing?

PLAY 9

Deploy in a flexible hosting environment

Our services should be deployed on flexible infrastructure, where resources can be provisioned in real-time to meet spikes in traffic and user demand. Our digital services are crippled when we host them in data centers that market themselves as “cloud hosting” but require us to manage and maintain hardware directly. This outdated practice wastes time, weakens our disaster recovery plans, and results in significantly higher costs.

Checklist

- ☐ Resources are provisioned on demand
- ☐ Resources scale based on real-time user demand
- ☐ Resources are provisioned through an API
- ☐ Resources are available in multiple regions
- ☐ We only pay for resources we use
- ☐ Static assets are served through a content delivery network
- ☐ Application is hosted on commodity hardware

Key Questions

- ○ Where is your service hosted?
- ○ What hardware does your service use to run?
- ○ What is the demand or usage pattern for your service?
- ○ What happens to your service when it experiences a surge in traffic or load?
- ○ How much capacity is available in your hosting environment?
- ○ How long does it take you to provision a new resource, like an application server?
- ○ How have you designed your service to scale based on demand?
- ○ How are you paying for your hosting infrastructure (e.g., by the minute, hourly, daily, monthly, fixed)?
- ○ Is your service hosted in multiple regions, availability zones, or data centers?

- ○ In the event of a catastrophic disaster to a datacenter, how long will it take to have the service operational?
- ○ What would be the impact of a prolonged downtime window?
- ○ What data redundancy do you have built into the system, and what would be the impact of a catastrophic data loss?
- ○ How often do you need to contact a person from your hosting provider to get resources or to fix an issue?

PLAY 10

Automate testing and deployments

Today, developers write automated scripts that can verify thousands of scenarios in minutes and then deploy updated code into production environments multiple times a day. They use automated performance tests which simulate surges in traffic to identify performance bottlenecks. While manual tests and quality assurance are still necessary, automated tests provide consistent and reliable protection against unintentional regressions, and make it possible for developers to confidently release frequent updates to the service.

Checklist

- ☐ Create automated tests that verify all user-facing functionality
- ☐ Create unit and integration tests to verify modules and components
- ☐ Run tests automatically as part of the build process
- ☐ Perform deployments automatically with deployment scripts, continuous delivery services, or similar techniques
- ☐ Conduct load and performance tests at regular intervals, including before public launch

Key Questions

- ○ What percentage of the code base is covered by automated tests?
- ○ How long does it take to build, test, and deploy a typical bug fix?
- ○ How long does it take to build, test, and deploy a new feature into production?
- ○ How frequently are builds created?
- ○ What test tools are used?
- ○ Which deployment automation or continuous integration tools are used?
- ○ What is the estimated maximum number of concurrent users who will want to use the system?
- ○ How many simultaneous users could the system handle, according to the most recent capacity test?
- ○ How does the service perform when you exceed the expected target usage volume? Does it degrade gracefully or catastrophically?
- ○ What is your scaling strategy when demand increases suddenly?

PLAY 11

Manage security and privacy through reusable processes

Our digital services have to protect sensitive information and keep systems secure. This is typically a process of continuous review and improvement which should be built into the development and maintenance of the service. At the start of designing a new service or feature, the team lead should engage the appropriate privacy, security, and legal officer(s) to discuss the type of information collected, how it should be secured, how long it is kept, and how it may be used and shared.

The sustained engagement of a privacy specialist helps ensure that personal data is properly managed. In addition, a key process to building a secure service is comprehensively testing and certifying the components in each layer of the technology stack for security vulnerabilities, and then to re-use these same pre-certified components for multiple services.

The following checklist provides a starting point, but teams should work closely with their privacy specialist and security engineer to meet the needs of the specific service.

Checklist

- ☐ 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
- ☐ 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
- ☐ 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
- ☐ Consider whether the user should be able to access, delete, or remove their information from the service
- ☐ “Pre-certify” the hosting infrastructure used for the project using FedRAMP
- ☐ Use deployment scripts to ensure configuration of production environment remains consistent and controllable

Key Questions

- ○ Does the service collect personal information from the user? How is the user notified of this collection?
- ○ Does it collect more information than necessary? Could the data be used in ways an average user wouldn’t expect?
- ○ How does a user access, correct, delete, or remove personal information?
- ○ Will any of the personal information stored in the system be shared with other services, people, or partners?
- ○ How and how often is the service tested for security vulnerabilities?
- ○ How can someone from the public report a security issue?

PLAY 12

Use data to drive decisions

At every stage of a project, we should measure how well our service is working for our users. This includes measuring how well a system performs and how people are interacting with it in real-time. Our teams and agency leadership should carefully watch these metrics to find issues and identify which bug fixes and improvements should be prioritized. Along with monitoring tools, a feedback mechanism should be in place for people to report issues directly.

Checklist

- ☐ Monitor system-level resource utilization in real time
- ☐ Monitor system performance in real-time (e.g. response time, latency, throughput, and error rates)
- ☐ Ensure monitoring can measure median, 95th percentile, and 98th percentile performance
- ☐ Create automated alerts based on this monitoring
- ☐ Track concurrent users in real-time, and monitor user behaviors in the aggregate to determine how well the service meets user needs
- ☐ Publish metrics internally
- ☐ Publish metrics externally
- ☐ Use an experimentation tool that supports multivariate testing in production

Key Questions

- ○ What are the key metrics for the service?
- ○ How have these metrics performed over the life of the service?
- ○ Which system monitoring tools are in place?
- ○ What is the targeted average response time for your service? What percent of requests take more than 1 second, 2 seconds, 4 seconds, and 8 seconds?
- ○ What is the average response time and percentile breakdown (percent of requests taking more than 1s, 2s, 4s, and 8s) for the top 10 transactions?
- ○ What is the volume of each of your service's top 10 transactions? What is the percentage of transactions started vs. completed?
- ○ What is your service's monthly uptime target?
- ○ What is your service's monthly uptime percentage, including scheduled maintenance? Excluding scheduled maintenance?
- ○ How does your team receive automated alerts when incidents occur?
- ○ How does your team respond to incidents? What is your post-mortem process?
- ○ Which tools are in place to measure user behavior?
- ○ What tools or technologies are used for A/B testing?
- ○ How do you measure customer satisfaction?

PLAY 13

Default to open

When we collaborate in the open and publish our data publicly, we can improve Government together. By building services more openly and publishing open data, we simplify the public's access to government services and information, allow the public to contribute easily, and enable reuse by entrepreneurs, nonprofits, other agencies, and the public.

Checklist

- ☐ Offer users a mechanism to report bugs and issues, and be responsive to these reports
- ☐ Provide datasets to the public, in their entirety, through bulk downloads and APIs (application programming interfaces)
- ☐ 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
- ☐ Catalog data in the agency's enterprise data inventory and add any public datasets to the agency's public data listing
- ☐ 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
- ☐ 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
- ☐ When appropriate, create a versioned API for third parties and internal users to interact with the service directly
- ☐ When appropriate, publish source code of projects or components online
- ☐ When appropriate, share your development process and progress publicly

Key Questions

- ○ How are you collecting user feedback for bugs and issues?
- ○ If there is an API, what capabilities does it provide? Is it versioned? Who uses it? How is it documented?
- ○ If the codebase has not been released under an open source license, explain why.
- ○ What components are made available to the public as open source?
- ○ What datasets are made available to the public?