De-risking Government Technology Guide

September 2024

GSA 18F

CONTENTS

01 Introduction

Only 13 percent of large government technology projects succeed.[1] The majority of these projects fail to deliver working software that meets the needs of agency staff or the public who must use it.

Why? Agencies at all levels of government face common challenges at every phase of a project. These include, for instance, the difficulty of choosing a software approach to serve needs that are often complex and unique, tension between bureaucratic processes and modern software development practices, and lacking in-house knowledge to assess the quality of code.

This guide was written to give government tools to lower the high risk of failure for technology projects. It addresses two main challenges — **how to choose a software solution and how to work with a vendor to build quality custom software quickly** — in four main sections. Building one upon the other, this guidance can help agency staff make and explain choices that can improve a project's chances of success.

What's in this guide

1. Understanding and choosing a software solution, which explains:
   a. The trade-offs between buying commercially available off-the-shelf (COTS) software and investing in building custom software
   b. When to use COTS or custom software, and why most projects will require both
   c. Why customizing COTS software to serve unique agency needs increases the risk that a project will fail

1 The Standish Group's Haze Report, 2015.

2. Four key principles for effective custom software development:
    a. Understand and commit to modern software development practices
    b. Use performance-based services contracting
    c. Identify and empower a full-time, in-house product owner to lead the project
    d. Set the team up for success

3. Buying custom software development services, including:
    a. How to write a solicitation for a performance-based services contract
    b. Things to keep in mind when budgeting for custom software development

4. Working with a vendor development team, including:
    a. Approaching vendor management as a partnership
    b. Leading product direction
    c. Setting up the relationship
    d. Reviewing work
    e. Maintaining the relationship

What's not in this guide

This guide can't address every question or situation that comes up in a government technology project, or every law or regulation that may apply to a specific project. The guide offers guidelines, models, and good practices to lower the risk of project failure by helping you understand what goes into building working software, and how to keep a custom software project on track.

We've tried to make the recommendations broadly applicable and useful for people working at any level of government. As laws and regulations differ across levels and across states and localities, we get into specifics only when they are relevant to procurement at a particular level of government.

The guide is focused on the full life cycle of acquisition activities from writing a solicitation to evaluating bids, building a positive relationship with a vendor to managing conflict. But it doesn't touch on every aspect of these stages. Where we can link to a resource for further information, we do.

Who's the guide for?

- Government staff at the federal or state level who are directly involved in the procurement and/or post-award phases of technology acquisition
- Government product owners and program or technical staff who want to understand how the buying process affects product delivery
- Government employees involved in planning, reviewing, budgeting for, and approving technology projects

Who are we? Why trust us?

We're federal employees who work for 18F, a group within Technology Transformation Services at the General Services Administration. Since 2014, 18F teams have partnered with federal and state agencies to help them acquire human-centered technology systems and services, as well as build and update systems, processes, and culture. We're contracting officers, technologists, researchers, designers, engineers, and product managers.

History and authors

The De-risking Government Technology Guide 2.0 consolidates content from the two parts of the original De-risking Guide: the "State Software Budgeting Handbook," released in August 2019, and the "Federal Agency Field Guide," released in September 2020. In addition, the new section "Working with a software development team" includes detailed guidance about vendor management.

These updates were made possible by the Office of Management and Budget's Facing a Financial Shock initiative. From May 2023 to June 2024, this funding enabled an 18F team to assess the feasibility of solutions to common challenges within state government IT acquisition, such as knowledge and skill gaps, focus on planning over results, and bureaucracy. After identifying vendor management to be a major area of concern among agency staff, the team took this opportunity to revise the De-risking Guide with relevant content.

The current guide reflects the contributions of many current and former 18F staff:

- Authors of the State Software Budgeting Handbook
  - Robin Carnahan, Randy Hart, and Waldo Jaquith
- Authors of the Federal Agency Field Guide
  - Mark Hopson, Victoria McFadden, Rebecca Refoy, and Alicia Rouault
- Contributors to the Federal Agency Field Guide
  - Alan Atlas, Heather Battaglia, T. Carter Baxter, Kelsey Foley, Waldo Jaquith, Ryan Johnson, Brandon Kirby, Ian Lee, Miatta Myers, Steven Reilly, Stephanie Rivera, Peter Rowland, and Greg Walker

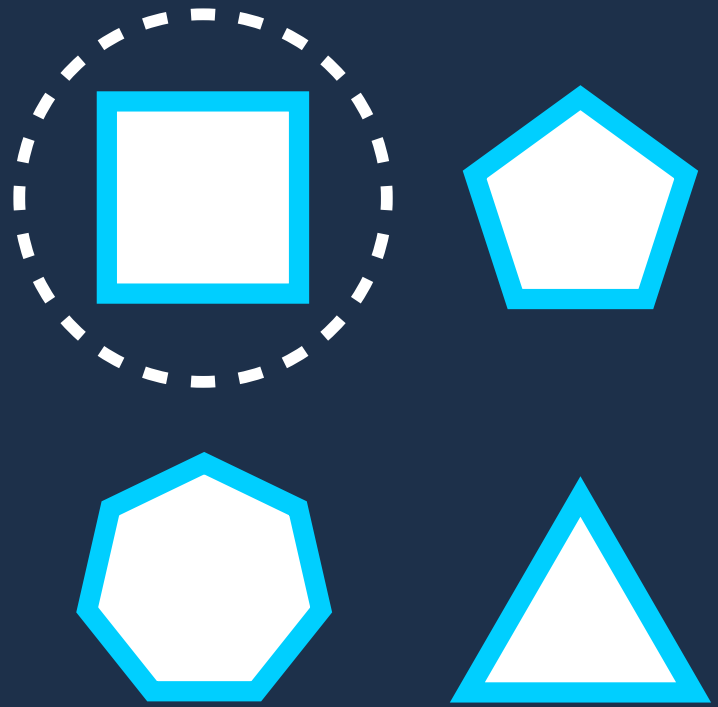
- Authors and editors of the De-risking Guide 2.0
  - Alan Atlas, Elizabeth Ayer, Brian Burns, Stacy Dion, Randy Hart, Mark Hopson, Selena Juneau-Vogel, Miatta Myers, Laura Poncé, Peter Rowland, Amelia Wong, and Lindsay Young

With thanks to:

- All who shared feedback on the guide's content:
  - From 18F: Claire Blaustein, Lalitha Jonnalagadda, Amanda Kennedy, Jason Nakai, Allison Norman, Cale Rubenstein
  - From Technology Transformation Services: Davida Marion
  - The several state employees who participated in user research
- Igor Korenfeld for designing the guide's PDF template
- Mel Choyce for designing new visual elements, updating the layout, and refining the PDF template
- Nate Borrebach for building the new online version
- Komal Rasheed for her invaluable guidance and support throughout the entirety of the Facing Financial Shock project

If you have feedback on this guide, please let us know through the 18F Guides GitHub repo.

02 Understanding and choosing a software solution



SUMMARY

Understanding the benefits and risks of commercially available off-the-shelf (COTS) and custom software will help government agencies choose a solution appropriate to their needs.

A major reason that government technology projects fail or struggle is that government agencies often approach obtaining software as a matter of building or buying it. The reality is more complex.

Many custom-built software systems are composed partly of commercially available products and services (cloud hosting, for instance). Meanwhile, agencies frequently buy commercial software products and then spend additional funds customizing them to suit their specific needs.

A government technology system is almost always a mixture of commercial and custom parts. This complexity requires *building thoughtfully and buying differently* than has been done in the past. It's critical to consider the costs and benefits of commercial products and custom development. One should ask: "Can we buy this piece of the system without having to customize it?" And: "If we build a custom piece of software, how do we ensure it is delivered on time, on budget, and satisfies our users?"

Understanding the benefits and risks of the options, and when each is appropriate, is necessary to answering those questions and setting your project up for success.

Commercially available off-the-shelf (COTS) software

In plain language, **commercially available off-the-shelf (COTS)** means software that is sold to the government in the same form it's sold to any other consumer.

COTS software is designed to do specific things and is *configured* to meet your organization's needs and preferences. **Configuration** of COTS software means making changes to the product's available "out of the box" settings. For example, types of configuration include changing the layout of your email inbox or deciding if a field appears on a form by turning options on or off. Configuration changes don't require **customization**, which is when *a developer modifies the product's code base to meet your needs*.

SOME BACKGROUND ON COTS SOFTWARE AND ITS BENEFITS

COTS software, like other commercial items, products, and services, is promoted and mandated for use at the federal level to the "maximum extent practicable" thanks to two laws: the Federal Acquisition Streamlining Act of 1994 (Public Law No. 103–259) and the Clinger Cohen Act of 1996 (Public Law No. 104–106). The federal government's shift towards commercial offerings influenced many state and local governments as well. Along with the rise of the internet, these laws changed how the government could buy information technology. They made commercial items like COTS software exempt from the more rigorous procurement processes the government uses to evaluate products and services.

The rationale for these changes was that the pressures of a competitive market were expected to keep costs low and quality high for consumers. That's worked well for physical objects, but software is an inherently different kind of product with a different life cycle. This essential difference is the source of the government's risk when acquiring software from the commercial market.

WHEN TO CHOOSE COTS

COTS is the right choice for meeting a need that *many* other buyers have, like email. An agency could develop its own email system, but it would be a waste of time, money, and effort since existing COTS email systems come with a wide array of features and functions that any buyer can use to meet its needs.

Custom software

Custom software refers to software code written specifically for a buyer's needs. Rather than being a commercially available *item*, custom software is built by a development team working in-house or through a vendor that builds the product and works with your IT department to put it into production online for its intended users.

WHEN TO CHOOSE CUSTOM SOFTWARE

If your agency has a unique need that is currently not served by a large marketplace — something other than email or video conferencing, for example — you should invest in building custom software to meet that need. This is a likely scenario for government agencies, which often have unique requirements and specifications, as well as laws and policies they must follow.

"Unrecognizably modified off-the-shelf" (UMOTS) software

If you buy a commercially available off-the-shelf (COTS) product and then modify it to meet your needs, you are licensing COTS and paying for custom software development. If those changes are more than minor (use our test to find out), you may end up with what some call **"unrecognizably modified off-the-shelf software" (UMOTS)**.

WHY TO AVOID UMOTS — OR BEWARE CUSTOMIZING COTS

UMOTS describes the frequent and risky tendency of government agencies to choose a COTS product and then modify it to such an extent that it is no longer compatible with updates to the core COTS product. It's responsible for the failure and struggles of many government technology modernization projects. Unfortunately, some vendors often "sell" UMOTS during the solicitation process with inaccurate or incomplete explanations of a COTS product's functionality.

We advise you to avoid UMOTS. It increases the risk of project failure and eliminates the primary benefit of COTS, which is to not reinvent the wheel. When you modify a COTS product, it becomes difficult and expensive to maintain. It may function poorly or not at all. It has the least amount of transparency and control for the buyer. It typically results in your agency becoming locked into long-term reliance on a single vendor (known as "vendor lock-in").

In the federal context, another reason to avoid extensive modification of COTS software is that you aren't complying with regulations. The Federal Acquisition Regulation (FAR) states that if it is necessary to make customizations or modifications to the technology to meet

federal requirements, it isn't a commercially off-the-shelf item. By law, only minor modifications are allowed for a product to still be considered commercial. Minor modifications refer to those that *"do not significantly alter the nongovernmental function or essential physical characteristics of an item or component, or change the purpose of a process"* (FAR Part 2.101).

You can avoid the risk involved in customizing COTS software if you:

- Use our test questions for identifying UMOTS.
- Conducting thorough market research before and during the solicitation process. Along with informing you of what's available, market research should help you sort out if the agency's needs are best served by custom software or by adapting agency processes to be compatible with a COTS product.
- Use risk mitigation prototyping.

TESTING FOR UMOTS

If you're thinking about acquiring a COTS solution that would need any degree of customization to meet an agency's needs, you could end up with UMOTS. To avoid that outcome, ask this set of questions.

Will the vendor need to write *any* software code to enable the COTS product to meet your requirements and specifications?

If the answer is "yes," you'll very likely end up with UMOTS. This is because one or both of these things will happen:

- The modification will alter the nongovernmental function of what the software was originally designed to do.
- The labor costs to change the code will be higher than the base price (licensing plus sometimes hosting fees) of the product itself. (Derived from FAR 2.101)

Typically, customizing COTS software results in both things happening. Modifying a commercial product from its nongovernmental function takes a lot of developers' time and effort. As a result, the costs for that work will almost certainly exceed the base costs.

To know for sure if the new code will result in higher labor than base costs, add up the proposed cost of labor and compare the sum to the base cost of the COTS solution.

If the labor costs exceed the base cost, the solution is UMOTS.

If the vendor won't provide an itemized list of labor costs, it's also a sign you will end up with UMOTS.

Has any organization successfully implemented the COTS solution …

1. In a similar time frame to the one you're planning?
2. Within budget?
3. And to the satisfaction of its users?

When it comes to COTS software, you should expect to find — or that the vendor can supply — *at least three examples* that clearly demonstrate successful implementation according to those terms.

If you can't find those examples, the solution is UMOTS.

Other questions to ask to figure out if your custom code will result in UMOTS

- Will modifying the COTS software mean it can no longer follow the routine schedule for upgrades and patches?
- Once modified, will the vendor own modifications to the resulting product or system?
- Is the COTS vendor being unclear about the cost to customize, maintain modifications, or migrate existing data? Or about ownership of and access to government data, or how to export data when the contract ends?

If the answer is "yes" to any of these questions, you'll likely end up with UMOTS.

A government technology system is almost always a mixture of commercial and custom parts. This complexity requires building thoughtfully and buying differently than has been done in the past. It's critical to consider the costs and benefits of commercial products and custom development.

A cautionary note on no-code and low-code software

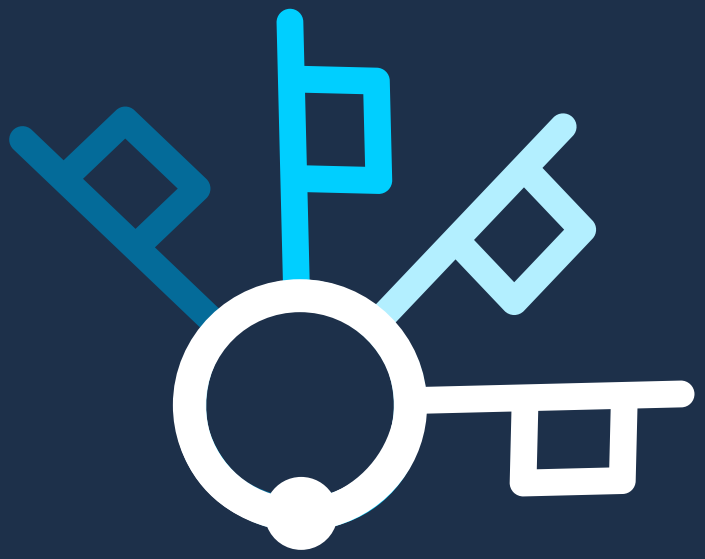
No-code and **low-code** software platforms allow you to build applications with back-end databases without writing any code or significantly less code. They are being sold aggressively to government as an alternative to developing custom software applications. This is an appealing sales pitch for agencies that don't have the resources or experience to manage custom software, but these solutions often require custom development to make them work for agencies' needs. While they may seem like easy and fast solutions compared to custom development, they can actually be more difficult and expensive, and lead to greater risk of failure.

As with any COTS product, these solutions can be an appropriate choice when your needs are straightforward and can be served by the platform's standard functionality. However, agencies often find out after they have committed to a no-code or low-code platform that its core functionality can't do something the agency needs. The agency must then pursue custom development to enable additional functionality within the limitations of the platform, which is often expensive, clunky, and makes the application more difficult to maintain. In the end, the agency often has to make compromises and accept a lower level of performance — while still paying a premium.

The following sections of this guide are specific to writing a solicitation for and managing custom development projects, so they don't apply specifically to implementing a no-code/low-code solution. Still, as these platforms don't eliminate the need for careful development practices, some of the principles for designing, building, and evaluating applications apply to developing applications regardless of the underlying technology.

As with any technology project, you may lower risk by selecting the right technologies to use to build the end product from the beginning.

03 Four key principles for effective custom software development



SUMMARY

When contracting a development team to build custom software, agencies should use performance-based services contracting and understand modern software development practices, a product owner's role, and how to set the team up for success.

If you've decided that your needs are best met with custom software, your goal is to build it in a way that maximizes cost efficiency and reduces risk through every stage of development.

Many government agencies don't have personnel who can create and maintain custom, human-centered software. They must buy the time and skills of professionals to form a development team to do that work. In other words, the agency must go through the acquisition process to procure the services of a vendor. That vendor team must also be experienced in using modern software development practices.

These key principles will enable you to contract with a development team who can build custom software successfully:

1. Understand and commit to using modern software development practices.
2. Use performance-based services contracting.
3. Identify and empower a full-time, in-house product owner to lead the project.
4. Set the team up for success.

Principle #1: Understand and commit to modern software development practices.

Government agencies have typically used the "**waterfall**" method for developing software, which involves a lot of advance planning and collecting comprehensive requirements at the beginning of a project. Unfortunately, this approach increases the risk that custom software development will fail because planning often takes years to complete and it falsely presumes that all needs can be accounted for before a project starts. By the time the contract is awarded, the gathered requirements no longer represent current agency needs, priorities, and resources.

A less risky approach to building software in government is to use the modern software development methods and practices defined below. They will help you plan appropriately, solicit and evaluate vendor proposals, and acquire professional services with the right experience and skills.

Five key modern software development methods and practices:

- User-centered design
- Iterative and incremental development
- Unified development infrastructure
- Service-oriented architecture
- Open source software

USER-CENTERED DESIGN

User-centered design is the practice of building software so that the people expected to use it can actually use it. In government, users (sometimes called "end users") may be government staff

and/or public users. (User-centered design shares many principles with related fields such as user experience design, customer experience, and service design.)

User-centered design follows repeating cycles of research with real users of the software, design, and development. **User research** includes interviews, usability testing, and other methods. These reveal users' expectations and needs for the software. They also expose points of confusion and bugs in code.

User research is integral to building working software. Hearing from end users themselves is the only way to get and understand their perspectives and ensure that you're addressing their needs. There is no substitute for direct user feedback. The perspectives of a stakeholder who has deep experience with a program or system are still not representative of a real user.

Insights from user research are often used to write "user stories." A **user story** is written with the syntax:

> "As a [**role**], I need [**this thing**], so I can [**accomplish this**]."

For example:

> As a **social worker**, I need **case notes to be cached on my phone**, so that I can **access case notes in areas without mobile phone service**.

User stories, along with technical considerations, inform the design and development of software. Ideally, user research happens continually throughout the entire project because user needs may evolve over time.

Project stakeholders and team members can only guess how users will use software. Designing with and for users is the only way to ensure the software will serve their needs.

Consult the 18F User Experience Guide for more detail on approaches.

ITERATIVE AND INCREMENTAL DEVELOPMENT

Effective software systems are built by a development team that uses iterative and incremental methods.

Today, one of the most popular versions of iterative development for building software is called “agile.” Its goal is to test working software with its intended users as soon as possible to find out if it meets their needs. And, if not, to correct it quickly so it does.

Agile is an alternative to the waterfall development process described above. It avoids the risks of using waterfall by empowering a development team to decide how it builds the product, and to use practices that enable it to work quickly and change course as needed based on new information.

There are several methods for practicing agile. The most prevalent is called “Scrum.” Its key features include a self-organizing team, customer focus, and responding to change.

A Scrum team usually includes five to nine people. Depending on the nature of the project, it may include developers, product managers, user experience (UX) researchers or designers, content strategists, and/or security experts.

Agile tools and methods support quickly building code and responding to new information. They act as “sources of truth” and guardrails for prioritizing and planning work. They include:

- **Product vision**: a short description of the product’s primary goal
- **Product roadmap**: a high-level diagram of how the team envisions building the product over time
- **Product backlog**: a list of product features and bug fixes that is usually written in the user-story format
- **Burn-down chart**: a graph that visualizes the amount of work left to be done on a project and how much time it is estimated to take
- **Burn-up chart**: a graph that visualizes completed work
- **Project risks**: a list of conditions that could affect the project’s outcomes and that the team works to mitigate

A Scrum team works in **sprints**: short, regular cycles of work that may be as brief as a week and as long as four weeks. Two weeks is the most common.

On day one of a sprint, the team plans only what it’ll do for that cycle. At sprint’s end, the team reviews its work, demonstrates the software to stakeholders, and then plans the next cycle by pulling user stories from the backlog. This process is repeated until the team has addressed all of the user stories or the budget for the project runs out, whichever happens first.

Each sprint, without exception, delivers functioning software: tested, documented, and ready for use. In this way, the team delivers value constantly and quickly develops software that is good enough to be rolled out for broad use, and continues to refine and improve it.

UNIFIED DEVELOPMENT INFRASTRUCTURE

Modern software development practice is grounded in the principle that there shouldn't be a division between developing and operating software. The team that writes the software takes responsibility for how software performs in production (as a live application or site). This approach is associated with "DevOps" practices, which also rest on this principle of a unified development infrastructure.

This is achieved by using automated testing and deployment tooling that allow the entire process of creating the environment for deployment, and incrementally updating it, to be scripted and repeatable. These tools and practices make it possible to make a change in the code and implement it in the production service almost instantaneously. They make it easy to make smaller incremental changes frequently and catch and fix errors.

SERVICE-ORIENTED ARCHITECTURE

Large and complex technology systems are made of smaller independent components that perform specific functions or services. All the parts can function together thanks to shared standards and application programming interfaces (APIs).

Each component's API contains a set of rules for how to communicate with it and call on it to perform its specific function. Systems that are built with this modular architecture are more flexible and sustainable. By standardizing and documenting the way components communicate with each other, a developer can focus on building components independently.

OPEN SOURCE SOFTWARE

Open source software is software with source code that anyone can inspect, modify, and enhance.

Developers often choose to build with open source software as it has many benefits. Building with open source technologies, and in an open code repository, often leads to a better and more secure end product than proprietary code. (This assumes developers follow best practices for open source software security.) Open source practices encourage critical evaluation and participation from contributors. These practices can lead to suggestions for improvement and identifying bugs and vulnerabilities. When an open source code base is used by a strong community of developers, everyone benefits from this active refinement as it continuously improves the code's quality and security.

Since the public funds government software projects, the government should allow the public to consult and use what it's paid for. Making government software projects open source enables the public — and other agencies — to leverage these investments for their own purposes. It also increases transparency and makes these investments publicly accessible and reusable by default.

There are other benefits to open source development for government technology projects as well:

1. Open source software makes collaboration easier among agencies, contractors, and the public because it is meant to be reused and adapted. It allows anyone that uses it to focus on using the code for their specific needs, rather than having to build and maintain common features from scratch.

2. The government retains ownership of the code, which reduces the risk of dependence on a single vendor.

3. It levels the playing field for future procurements and increases competition. New offerors can review the code to help them decide if they want to bid and what to include in their proposal.

4. Software developers contribute to open source projects to demonstrate their skills to colleagues and employers, current or future. There is a mutual benefit for the contributing developer and for the project. Public-facing government software tends to have high visibility and a built-in user base. Making that software open source cultivates a community of developers and other users around the project that is invested in making it better.

(Open source software isn't appropriate for every project, such as when an agency doesn't have the rights to reproduce and release the code. Or, when publicly releasing the item is restricted by a law or regulation, such as the Export Administration Regulations or the International Traffic in Arms Regulation.)

Principle #2: Use performance-based services contracting

Software developers as a labor category and profession qualify as "commercially available" and "professional services" under the FAR. To acquire the time and expertise of a team experienced in modern software development services, you must use performance-based acquisition methods for the solicitation, competition, and evaluation of proposals.

Performance-based services contracting (PBSC) stresses that all aspects of an acquisition must be structured around the purpose of the work to be performed, and involve a way to assess contractor performance objectively rather than dictating the manner in which the work is to be performed.

This approach to contracting professional services ensures that:

- Contractors are given freedom to determine how to meet the government's performance objectives.
- Appropriate levels of quality in performance are achieved.
- Payment is made only for services that meet those levels.

Learn about the solicitation process for performance-based services contracting.

Principle #3: Identify and empower a full-time, in-house product owner to lead the project

Modern software development is led by a **product owner**. In tech lingo, a **product** is the thing a development team builds. It may be a website, mobile app, data service, intranet application, etc. A product owner is the *individual* responsible for making sure the team builds a thing that serves the needs of its users, as confirmed by research. They receive and review a vendor team's work.

A product owner works closely with the team to ensure its work is focused on creating a product that meets its users' needs and organizational goals. Their daily work includes deciding on priorities, adjusting direction based on feedback, and communicating with stakeholders. Unlike a project manager, who focuses on planning and monitoring projects, a product owner's focus is on the product's value to users and the team's quality of work and well-being. They choose among priorities throughout the project, weighing the best response to information as it arises, and the value and impact of change versus stability.

Slow product decision-making is a common problem for development teams. To avoid it, the product owner must be available to the team and empowered by the organization. Specifically, the product owner of a government technology project must be:

- An individual, not a committee.
- Employed by the agency the product is being built for.
- Assigned at *least* half-time to the project, ideally full-time — especially for large or high-priority projects.
- Permitted to make most decisions about the product's development *without having to seek approval from stakeholders*.

A product owner doesn't need to be an expert in technology. A strong product owner understands the needs of the product's users, the goals of the organization and any legal, technical or policy constraints that need to be weighed in decisions.

While it's possible for a product owner to learn "on the job," it's better that they receive formal training in agile product ownership. Free or paid training is offered online and in person through many sources (such as the 18F Product Guide). If the product is critical to the agency, the product owner should have prior experience in modern software development practices or access to an experienced product coach.

LEADERSHIP'S ROLE IN A CUSTOM SOFTWARE DEVELOPMENT PROJECT

Agency leadership's role in custom software development is to create an environment where modern software development practice is possible.

Leadership is responsible for declaring what is important from the perspectives of policy and operations, and any concrete constraints the team must work within. They should be ready to take action when questions arise that fall outside the product owner or team's responsibilities. They are also responsible for enabling the team to work in new ways and giving them space to innovate to meet the goals.

It's essential for the project's success that leadership make it an organizational priority to support modern software methods, and work to align governance and oversight processes to permit their use. For example, if an agency has traditionally used waterfall processes to develop software, including expecting detailed documentation of

requirements, these management practices will make it impossible for the team to adjust course based on user feedback. Leadership should spearhead the effort to shift an organization's culture and policies to enable iterative software development or people will be discouraged from trying it again. This is especially important as learning new methods takes time and will involve missteps.

Other responsibilities for leadership include:

- Providing funding
- Helping overcome internal challenges
- Serving as authorizing officials in security accreditations
- Nearing launch of the software, facilitating internal and external communications

It can be helpful for leadership to be involved in helping to set the product vision, reviewing the roadmap as it relates to organizational strategy, and seeing demos of the software.

There's no one-size-fits-all approach. The degree that leadership should be involved in the project depends on the nature of the challenge, organizational culture, and work preferences of individuals.

Principle #4: Set the team up for success

Before you award a contract, there are some things to do that will make it possible for the development team to work efficiently and productively from day one of a project. These reduce the risk of delays and wasting funds. Ideally, these are addressed before onboarding a vendor development team.

HIRE TECHNICAL STAFF IN-HOUSE, IF NEEDED

If your agency doesn't have leadership, budgeting, or technical staff who have experience with modern software development practices, it's missing knowledge and skills that are crucial for budgeting for and building custom software successfully.

It may be tempting to rely on vendors to fill this gap. Or, if you're at a state agency, the state's central IT department. But, agencies are best served by in-house staff who have technical knowledge and understand the software's relationship to the agency's mission. They can both confidently represent the contract and assess the quality of the vendor team's performance.

To determine if your budget office or leadership has the necessary experience to consider software requests or lead software projects, ask around. All but the smallest agencies will have technical staff who can join project leadership. There are few budget offices who currently employ software developers.

If your agency lacks staff with the technical knowledge to pursue a custom software project successfully, you'll need to hire someone who does — even if only seasonally or on contract. Your best bet is

a developer or designer with experience building modern software, ideally for the government.

The cost of bringing in a developer or training current employees in modern development practices is tiny in comparison to the cost of a custom software project. Plus, once you have that knowledge in house, it can be drawn upon for future projects.

Software is never "done." It will always need to adapt to changing user needs, technology, policy, regulations and laws. To properly maintain it, you must have developers on staff who fully understand the program or system.

ALLOW FOR AND PROVIDE SUPPORT FOR A REMOTE TEAM

Allowing for the development team to work remotely will give you access to the best development resources in the United States. It will also likely lower costs while increasing competition. There's a significant difference in the salary of software developers in the most expensive and least expensive states, and small businesses will be able to enter the vendor pool.

Remote collaboration is easy with modern online tools. An agency product owner can communicate daily with a distributed team through any number of available tools that support video conferencing, instant messaging, task management, collaborative whiteboarding and document editing.

Government agencies often struggle with enabling remote collaboration due to network restrictions and software approval policies. Ensuring a remote vendor team can be productive from day

one of a project requires making sure they can access such tools well before work begins.

Determine which collaboration tools teams will need and make those available to them. As an interim step, agencies may want to develop a provisional Authorization to Operate (ATO) process for piloting tools that are relatively low risk. This process could inform decisions about which tools should go through the ATO process to be rolled out more broadly.

CLEAR THE "PATH TO PRODUCTION"

Unlike in the private sector, making government systems available to end users is a highly regulated and scrutinized activity. It requires technical, legal, legislative, and other approvals, along with extra layers of development.

Before publishing a solicitation, figure out the process for getting a vendor team access to the hosting and deployment environments necessary for testing and launching the software, and make sure they have access to them. In other words, clear any bureaucratic obstacles the vendor team could experience in the "path to production" and document the path clearly and comprehensively. (To find out if code can be deployed to the needed environments, you can: 1) prototype a solution or 2) talk to agency technical staff. Ideally, do both.) If this process for access is not done before a team is onboarded, it can result in wasted time and effort, as well as frustration.

Each agency will have its own set of processes, rules, and regulations around security clearances. (For instance, the Homeland Security Presidential Directive 12 Policy applies across the federal government.) To streamline work that concerns security issues, write a requirement into the contract that the vendor must delegate an

individual to act as the security clearance liaison (a point of contact for all questions and requests) for the project.

An agency that meets the following criteria can feel confident they can award an agile software development contract and that the vendor onboarding process will be relatively smooth:

- There is access to a hosting environment, administered by an employee at the agency.
- There is an organizational account on a social code repository (for example, GitHub, GitLab, or Bitbucket) for the agency, administered by one or more employees of the agency.
- There is a process by which changes made to code on the repository are automatically deployed to the hosting environment and the agency has the ability to release frequently (i.e., a unified development architecture).

PROTOTYPE TO LEARN

Prototyping is an exercise that will help you understand your agency's ability to support an agile software development project in terms of technology, human resources, and policy.

The exercise can be as simple as publishing a single "Hello, world" webpage. It should answer these questions:

- What is the administrative process to gain access to the hosting and deployment environment?
- What processes or policies does a software team need to work through to get access to services and deploy applications to them?
- Which stakeholders are required for approvals? What things do they need to approve? What form do applications for approval take?

This work will help the team understand their tool preferences and document internal processes. The prototype can also be useful as supporting documentation in an RFP or to give to the winning vendor.

ASK QUESTIONS OF AGENCY TECHNICAL STAFF

Invite relevant technical staff at your or your partner agency to a meeting to get answers to these questions:

- How are existing digital products hosted and deployed? Who is involved in those processes?
- How do we get access to the agency's deployment environment (for example, cloud.gov, Amazon Web Services, Microsoft Azure, on-premise servers)?
- Are there existing technology stacks, solutions or components that are approved for use or that are recommended? Are there preferences for any of the options?

Having clear answers to these questions is the *minimum* amount of information an agency should have going into a build. If the answer to these questions are unknown, or the answers are murky, more investigation is needed prior to publishing an RFP.

04 Buying custom software development services

SUMMARY

Learn about the elements of a solicitation for a performance-based services contract, and tips on how to budget for technology projects so they can be pursued iteratively and incrementally.

To acquire a development team to build custom software using modern software development practices, you should use a performance-based services contract. That begins by writing a solicitation in the way outlined in this section. You should also plan budgeting for technology projects so it can support an iterative and incremental approach.

Writing a solicitation for a performance-based services contract

A **solicitation** is the document that articulates an agency's need for a product or service in terms that enable businesses to submit bids or proposals within a competitive bidding process. Agencies have different terms for solicitations, including Request for Proposal (RFP), Request for Quote (RFQ), and Request for Offers (RFO), among others.

The **general solicitation process** has three steps:

1. An agency writes a solicitation to seek industry help to satisfy a need.

2. Industry reviews and responds to the solicitation with proposals for satisfying the agency's need.

3. The agency picks the proposal it has determined to be the strongest offer with the least amount of risk.

This section explains what happens in the first step in the process: market research and the writing of the solicitation. The second step occurs in the vendor context. On the third step, refer to our guidance on evaluating proposals and bids.

CONDUCTING MARKET RESEARCH TO IDENTIFY SUGGESTED SOURCES

Every solicitation begins with the agency conducting market research, which continues throughout the process of writing the solicitation.

The FAR defines **market research** as "collecting and analyzing information about capabilities within the market to satisfy agency needs." ("Needs" and "requirements" are often used interchangeably in the market research context.)

Done well, market research shapes the final product as well as a product's requirements. If market research is done poorly, the project will face issues from the beginning, including delays in the schedule, increased costs and, ultimately, unsatisfied users.

Consult our detailed guidance for conducting market research in the resources section.

WRITING THE SOLICITATION USING 18F'S AGILE CONTRACT FORMAT

Our **Agile Contract Format** has three elements:

1. A statement of objectives for performance-based services

2. A time-and-materials type contract

3. A Quality Assurance Surveillance Plan (QASP) that defines the expectations of quality that will be monitored throughout the contract using specific indicators

The solicitation also spells out a rationalized competitive award process to potential vendors and the evidence-based evaluation methods that will be used to evaluate bids and proposals.

Statement of objectives for performance-based services

FAR Part 37.601 states a performance-based solicitation may either be a performance work statement (PWS) or a statement of objectives (SOO).

We use a statement of objectives for the purpose of competition since, unlike a performance work standard, it requires industry to produce evidence that will help you evaluate proposals. A statement of objectives requires these six elements:

- Purpose
- Scope or mission
- Period and place of performance
- Background
- Performance objectives (that is, required results)
- Any operating constraints

18F has created a statement of objectives template for writing a solicitation for agile software development services. (Download the template (Microsoft Word).) To use our template to create an Agile Contract Format, you will plug agile artifacts into the sections as follows:

SOO section	Agile analog
Scope or mission	Product vision
Performance objectives (that is, required results)	Product backlog
Any operating constraints	Non-functional requirements

Time-and-materials type contract

A **time-and-materials (T&M) type contract** is used for acquiring services at fixed hourly rates for labor, and supplies or materials used to create and/or make the end product available to users.

You will use a T&M type contract with a not-to-exceed ceiling to buy the time and expertise of a development team that will use modern software development methods to build a product. (A T&M contract is preferred to a Labor-Hour (LH) type contract when acquiring professional software development services for the reason that it enables a team to buy any tools or services they need to build the software and enable its functionality. For example, the team may need to pay a service provider to support SMS messaging in an application.)

A T&M contract gives the development team the flexibility, freedom, and professional discretion it needs to develop software iteratively based on user research. It also allows them to respond to changes that arise on the program side, such as shifts in priorities or resources. For instance, if the government decides to terminate early for some reason, such as a change in priorities or the vendor is not

performing, this type of contract enables it to do so without having to go through the burdensome termination procedures that come with other types of contracts, and to still own the software that has been delivered up to that point. Work just stops being assigned to the vendor (such as through the product backlog) and they can't bill their time to the government.

For software projects, a T&M contract protects the government's financial and performance interests better than other contract types. The vendor bills on an as-needed basis and only for actual time incurred. Each submitted invoice must provide exact billing data. (Firm-fixed-price type contracts don't provide this protection as work is performed over time and submitted invoices lack detail.) A T&M contract dissuades vendors from falsifying invoices since the FAR deems false invoices a "false claim" and exposes the vendor to the risk of paying triple the initial damages for each instance.

Every T&M type contract requires a dollar ceiling and can't exceed three calendar years in duration in order to reduce the risk of the project going over budget.

We use a not-to-exceed (NTE) ceiling, provided regardless of the actual proposal price from potential bidders, to allow for the iterative nature of modern software development.

The FAR states that a T&M contract is not supposed to exceed three calendar years. Consequently, the period of performance is between one to three calendar years in total. (18F's Determinations & Findings artifact (Microsoft Word) provides a full justification for using T&M type contracts and explains this aspect in depth.)

Quality indicators defined in a Quality Assurance Surveillance Plan (QASP)

The FAR allows for a range of ways to establish and monitor contract performance. Quality indicators for software development are best declared through a well defined, objective set of criteria that serve as an assessment tool for both the government agency and the vendor. At the federal level, these criteria are put into a **Quality Assurance Surveillance Plan (QASP)**.

The QASP is the most rigorous way to oversee vendor performance. Waterfall projects often collect a long list of functional requirements before work begins and involve written status updates. In contrast, the QASP is focused on criteria that can be verified objectively and continuously throughout the performance period. The QASP sets the standard that, at the end of each sprint, all code is delivered to a government-owned repository and must be complete, tested, accessible, deployed, documented, and secure.

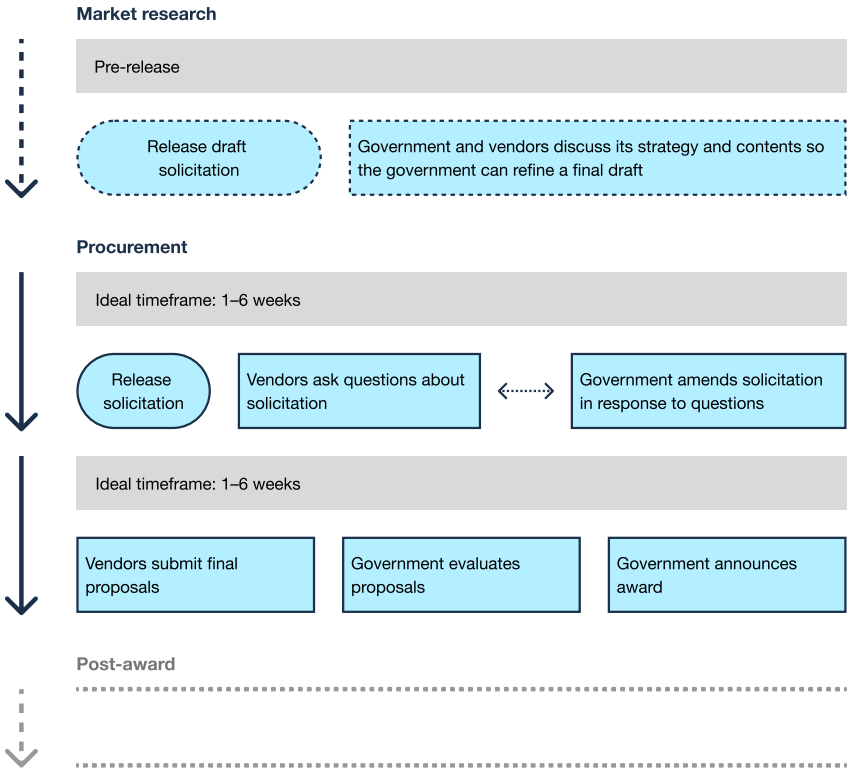
You can include this sample QASP, which covers a minimum set of quality indicators, without any changes in your solicitation. Modify it to meet your specific needs if necessary.

At the federal level, the government can allow a vendor to provide their own QASP. *We strongly advise against this*. If you allow the vendor to define its own measures of success, you give up one of the most powerful tools the government has for monitoring and ensuring quality.

Learn about the elements of a QASP and how to use them.

Rationalized, competitive award process

The competitive award process to acquire an outsourced development team varies for federal, state, or local agencies due to different acquisition laws.



At the federal level, for example, the process ideally takes four to 12 weeks from release to award. It includes these steps:

- (Optional) Release a draft solicitation as part of market research. Government and vendors discuss its strategy and contents so the government can refine a final draft.
- Release the solicitation.

- Vendors ask questions. Government amends the solicitation in response to questions.
- Vendors submit final proposals.
- Government evaluates proposals. Verbal interviews occur, if included in the process.
- Government documents award decision.
- Government announces award.

(The optional step to release a draft solicitation during market research is included because it's more likely to result in qualified proposals than a Request for Information (RFI), which is more commonly used. Further, verbal interviews are mentioned because 18F strongly recommends them to validate the approach provided in written proposals.)

The solicitation should require bidders to keep proposals under 10 pages. We recommend a hard limit of five pages, with the narrative sections kept to two to three pages each. Short proposals can:

- Increase and make competition more equitable by reducing the effort to create them. (New and small businesses don't always have the dedicated resources for writing proposals that larger companies do.)
- Reduce the likelihood of vendor protest.
- Reduce the time and work government staff spend evaluating the proposals.

Note: If vendors request more narrative space, it indicates one of two things:

- The vendor might not be experienced or comfortable working with an iterative approach to software development. A vendor experienced in responding to more traditional solicitations for waterfall development will be used to providing long, detailed explanations of how they would meet every requirement. Don't let vendors' requests for more narrative space dissuade you from keeping a page or word limit. We've found most vendors appreciate the request for brief explanations.
- The government's request was poorly written and is confusing to potential bidders.

Evidence-based evaluation methods

A solicitation explains the factors and methods the agency will use for evaluating proposals.

Agencies often include factors in solicitations that don't enable them to discern risk or meaningfully judge the content of proposals. These add significant amounts of time and effort to the evaluation process with little benefit.

Our approach uses only four key factors: technical approach, staffing approach, similar experience, and price.

Proposals are evaluated as follows:

- Technical approach, staffing approach and similar experience will be given more weight than price.
- Proposals will not be evaluated by a numeric point or color scoring scheme. Instead, each member of the evaluation team will review each proposal and list its pros and cons. The whole team will discuss the proposals' pros and cons to determine the strongest ones.
- Evaluation teams may use whatever materials are available beyond the proposals to help them decide: websites, news articles, samples of prior work, etc. (Include a statement in the solicitation that the government may use such information at its discretion.)

Technical approach

Inform vendors their proposed technical approach will be evaluated for how it describes its approach to modern software development practices. In particular, ask them to include the process they will follow to meet the solicitation's quality expectations for the software. Ask them also to identify any risks they anticipate in regard to the specific project's development effort and how it would address them.

Staffing approach

We've found that how a vendor proposes to staff a project is a strong indicator of how much experience they've had in working in iterative ways. As a result, the solicitation should inform vendors that evaluation of their proposed approach to staffing the project will focus on the stated skills and team composition.

Adding a key personnel clause

At the federal level, you may add a **"key personnel"** clause to the solicitation. It requires a vendor submitting a proposal to supply the résumés of the people proposed for certain positions and establishes that the government will approve any proposed replacements. Its purpose is to ensure that the vendor team committed throughout the contract operates at an equivalent level of experience and expertise to the key personnel included in the vendor's proposal.

For that reason, the key personnel clause is typically perceived as a quality control measure. But, requiring vendors to specify dozens of key personnel comes with some risks.

Before using a key personnel clause, consider:

- Most bidding vendors will not have enough people "on the bench" at the time of proposal to be able to commit to assigning them to the project when the contract is awarded. If a vendor that has committed key personnel gets the contract, they are then required to put those people on the project. As those staff are functionally benched until the contract is awarded, it can increase a bidding company's costs, as well as the likelihood of protests. The more a company invests or risks to make a bid, the more likely it will dispute a decision not to go with its proposal.
- The market for skilled developers is fluid. It's not possible for a company to lock down all key personnel months before the actual work *might* begin on a contract.

If you choose to use the key personnel clause, ask vendors to specify *two or three positions at most:* a project lead, a technical lead and, optionally, a design lead.

Finally, ask that all named key personnel participate in a **verbal interview**: a timed, unstructured question-and-answer session in which they will answer questions about the proposal's technical approach. Verbal interviews will allow the agency to better understand each contractor's proposed technical approach and to observe key personnel's interactions and working style. They are a critical quality control measure that confirms the information provided in the written proposal. Verbal interviews don't allow contractors to make presentations, ask questions, or change their submission in any way. (They are not FAR Part 15 oral presentations).

Verbal interviews should be tailored to each proposal. Consult our sample interview question bank in the Resources section.

Similar experience

Actual code is a far better indicator of how a vendor team is likely to perform under real-world conditions than exercises like "bake-offs" or "hackathons." Similar experience is best evaluated by reviewing concrete evidence of the vendor's work so you may assess its quality.

The solicitation should ask the contractor to submit code repositories similar in size, scope, and complexity to the work that the agency is undertaking.

Ask for links to two or three source code repositories that illustrate the work of the company or technical lead and other relevant key personnel. This may include examples from previous employment or volunteer projects, since many contractors won't have had clients willing to work in the open and thus no public code repositories to share. If a private repository is shared, the vendor must promptly provide access to the government-provided Git users.

Price

To evaluate price, you will calculate a total estimate by looking at the number of people, their allocation (part time, full time, etc.) and the hourly rate for their labor. The ideal size of a Scrum team is between four to nine people. A typical team has six. A Scrum team never has more than 10.

Before you send out a solicitation, you must also create an independent government cost estimate (IGCE). This will give you a baseline against which to judge the cost of proposals.

Use GSA's Contract-Awarded Labor Category tool to estimate the average hourly labor rate. (The average hourly labor rate will fluctuate over time based on contract awards. The contracting officer judges what is reasonable comparison pricing for labor.) Then, multiply that number by a reasonable number of hours that a person would be expected to work in a given year. For example, 1,880 working hours a year accounts for holidays and some leave.

The table below represents a sample IGCE. It assumes nine people and represents a typical mix of talent on a Scrum team. Based on average hourly labor rates as of June 2024, in this sample scenario the IGCE for one year of performance is $1.96 million.

Sample independent government cost estimate for a vendor Scrum team for one year

Position title*	# of people	Average hourly labor rate**	Estimated cost (# of people × avg. hourly labor rate × 1,880)
Senior Software Developer	1	$139	$261,320
Senior Designer	1	$119	$223,720
Software Developer	3	$120	$676,800
Designer	3	$97	$547,080
Content strategist	1	$136	$255,680
Total			**$1,964,600**

* Vendors will propose different labor categories or skill mixes based on how they typically operate and what they assume the work will entail. Expect position titles to vary based on the competition pool and the skills needed for a project. For example, "Senior UX Designer," "Senior Product Designer," "UX Designer or Researcher," etc.

** If an agency requires the development team to be on-site or have top-secret security clearances, expect the average hourly labor rates to be substantially higher and that fewer companies will be able compete for the work. The clearance process itself is an added cost. Also, hourly rates for these positions vary depending on where an individual team member lives within the United States.

Check out our in-depth guidance on how to evaluate proposals and bids and use our sample evaluator worksheet to help determine the strongest proposals.

Budgeting for custom software development

Government budget and appropriation processes and cycles typically run over one or two years. As a result, agencies must request funds for technology projects many months or even years before they can

begin. For example, a budget request made in the first year of a project may take two years to get approval. The contract may not be awarded until three years later. The software needed at the time of the budget request may not be delivered until five or more years have passed.

Because this process takes years, government agencies will often decide to pursue building or updating large, complex systems through one big contract because it seems like the most efficient way forward. However, this approach makes the project more likely to fail because it doesn't account for changes in agency needs that will occur over time.

A less risky way to build or modernize a major system is to embrace iterative and incremental approaches in budgeting as well as in software development. Start small rather than let your project snowball into something that's too big to succeed.

SMALL PROJECTS, SMALL BUDGET ALLOCATIONS

Large technology systems are made up of smaller component systems. To lower the risk that building or updating a large system will fail, carve a large project into several small ones and budget for these in small, incremental budget allocations.

A modular approach to budgeting for and building a large system insulates each small project from the others. As a rule of thumb, we recommend keeping budgets below $10 million. If one fails, it won't affect the others. Smaller projects also operate below a threshold that requires the layers of agency oversight that can delay and complicate the budget approval process involved with large projects.

BUDGET FOR A "RISK MITIGATION PROTOTYPE"

Building a prototype for a small part of a larger system prior to awarding a contract mitigates risk in several ways.

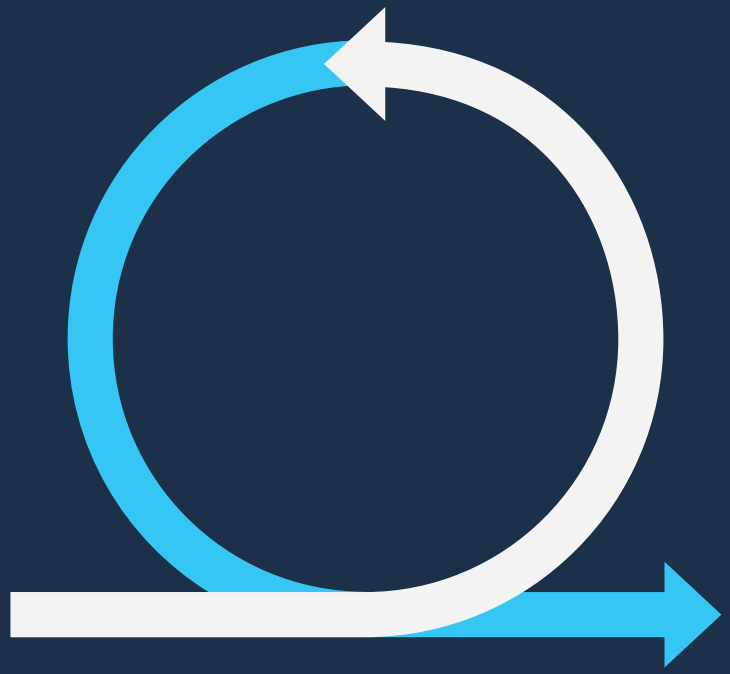
It exposes potential pitfalls and other issues that can only be identified when actually working with software code. In our experience, risk mitigation prototyping often reveals challenges in the "path to production" that should be mediated before awarding a contract to build custom software. For instance, it can prevent costly post-award expenditures that can accrue when getting ready to begin actual development.

It also results in code that can be tested by a system's intended users and used as an artifact in the solicitation process. In our experience, potential bidders find this early risk mitigation work very valuable for helping them decide if they're a good fit for an agency's needs and for how to staff a team for such an effort.

In our work with agency partners, we can often complete some form of risk mitigation prototyping with a team of three to four 18F staff in a few sprints. We document what we've learned as part of the solicitation document, so that an implementation team doesn't have to start from scratch in an unknown development environment.

(To fund this type of effort, federal agencies can apply to the Technology Modernization Fund (TMF)).

05 Working with a vendor development team



SUMMARY

The principles of agile oversight underlie every aspect of working with a vendor team to build effective custom software, from setting up and maintaining the relationship to leading product direction and reviewing the quality of work.

Introduction to vendor management

Once an agency has awarded a contract to a vendor to build custom software, its goal is to foster a healthy working relationship with the development team and make sure it delivers working software that meets the needs of its intended users. This work is typically called "vendor management."

As day-to-day management of a vendor's work on a custom software development project is the responsibility of the government product owner, this section is written primarily for people in that role. Where government technical leads and other staff should be involved, they are mentioned specifically.

In government contracting, "**vendor management**" means the activities and interactions with a vendor that occur after a contract has been awarded. That work should result in the vendor fully performing its duties under the contract and the organization achieving its desired outcomes from the contract.

In the context of a performance-based services contract, vendor management is not about enforcing the letter of contracts. It's about developing a productive relationship with the vendor, engaging closely with them as they work to identify solutions, and regularly evaluating the actual software they were hired to produce.

The contract broadly defines the outcomes you want to achieve together and sets the boundaries of how you and a vendor will work together. It doesn't dictate the exact nature of the day-to-day work of building software.

Indeed, it is a sign that you and your vendor have a healthy relationship if you rarely need to discuss the contract. If you're meeting multiple times a week and working closely together, your conversations focus on the work itself: what the team is doing, how they're doing it, and obstacles, complexities, and solutions they're discovering. There's no need to refer back to the contract.

Good collaboration and communication are the cornerstones of effective vendor management of a custom software development project. This is why the government product owner (whether or not a technical expert) must actively participate in the project team and in meetings where work is being done, and from the earliest stages of planning.

This section outlines good practices for building a healthy relationship with an outsourced development team through active collaboration and communication. It starts by explaining the principles of agile oversight, which underlie our guidance, and explains the practices in relation to the following aspects of vendor management:

- Setting up the relationship
- Reviewing the vendor's work
- Maintaining a healthy vendor relationship, including managing conflict

PRINCIPLES OF AGILE OVERSIGHT

The agile manifesto established a set of values for agile software development. They are explained here for a government context. If you followed the solicitation process for a performance-based services contract as explained in this guide, you have set up your relationship with your vendor to abide by these principles.

Individuals and interactions over processes and tools

You took the time to run a good procurement and select the best vendor. Now, create an environment that gives the vendor team space to do the work you hired them to do. Focus on building your relationship with the vendor team and communicating expectations for how you will regularly evaluate its work and not on how closely the vendor complies with established agency procedures and culture.

Working software over comprehensive documentation

Use regular demonstrations of working software rather than written reports to measure the progress of work. To evaluate the quality of demos, you'll use the quality indicators articulated in the performance-based services contract. (Some documentation is still required during the project to record details about how the software was built, security and privacy protection mechanisms, and instructions for its maintenance. Check out Reviewing the vendor's work for specifics.)

Customer collaboration over contract negotiation

As already noted, a sign of a good relationship with your vendor is that you're in constant communication. Setting up an environment

that enables professional, open dialogue between you and the vendor team is imperative to successful collaboration.

Responding to change over following a plan

A development team can build working software only if they can respond to needs as they evolve over time. The purpose of acquiring the team's services through a performance-based services contract was to enable the team to be able to work in this fashion.

Leading product direction

A government custom software project is successful only if the software delivers on the intent of the particular policy or service it was designed for. That purpose must remain front and center at all times.

The product owner is responsible for ensuring that software achieves its purpose and meets quality expectations. This work involves leadership throughout the many aspects of the project, including:

- Setting the overall product vision and goals.
- Communicating constraints.
- Translating the goals into work.
- Navigating and recording major decisions.
- Releasing and evaluating the software.

SETTING PRODUCT VISION AND GOALS

The **product vision** is the guiding statement of what a software project is trying to achieve. It should be clear and concrete about what will *change* in the world by delivering the software.

For example, the vision for a new system within a benefits program might be: "Make it easier for state agency workers to detect potential fraud and error, and take timely action to resolve discrepancies, while preserving participants' access to the full benefits they deserve." The vision for an internal purchasing platform might be: "Create a world where the federal government can work out in the open with nontraditional vendors to get quality solutions delivered quickly and cost effectively for the public." The vision can have more detail than these examples, but it should be immediately understandable and compelling for those involved in the product. (Consult the 18F Product Guide for detailed guidance on creating a vision.)

The vision should be established in the solicitation phase and stated in the contract. During the solicitation phase, the vision informs the goals the government is trying to achieve with the vendor's help.

After the contract is awarded, the vision helps the team maintain focus during the project. It provides motivation and serves as a tool for aligning the team toward around the same goal.

The vision helps the product owner and the vendor team set objectives and prioritize work by providing overarching guidance for what to pay attention to and how to weigh trade-offs. As context for all work, it is important to spend time establishing a shared understanding of the vision at the beginning of the project, communicating it frequently, and realigning around it when necessary.

The team should check their progress against the vision no less than quarterly and update its approach for delivering on it if needed.

There are a number of frameworks you can use to help you translate the vision into goals. Nonprofit organizations frequently use the framework of Impact, Outcomes, Outputs. Software development

teams more often use Objectives and Key Results; the North Star framework; or Goals, Signals, and Measures.

Choose a structure that fits how your organization discusses work and goals. Then tie the problems you're trying to solve to the outcomes the team is pursuing, for example, a 20 percent increase in digital application submissions, a 30 percent decrease in call center requests related to application submission errors, time to deploy a new feature shifts from once a quarter to every two weeks, etc. The most important thing is to have some way to explain how the day-to-day product development tasks form building blocks toward the vision.

COMMUNICATING CONSTRAINTS

It's also important to identify, communicate, and manage constraints that the development team may encounter while building the product. For example, the programming languages the agency can support or who can have access to production servers.

Even if a constraint is outside the control of the development team, the government must be transparent about any obstacles the team may face. Sharing this information allows the development team to plan for them. It also allows the development team and the government to brainstorm possible solutions and mitigations together.

TRANSLATING THE GOALS INTO WORK

It is not always easy to translate goals into work that delivers on those goals. In most cases, there are many different ways to approach a problem, and it is rarely clear how well an idea will work before it's realized and users can try it out. The product owner helps the team to

navigate this uncertainty. The product owner works with the team to prioritize work that is most likely to deliver the most progress towards the goal soonest, based on user research.

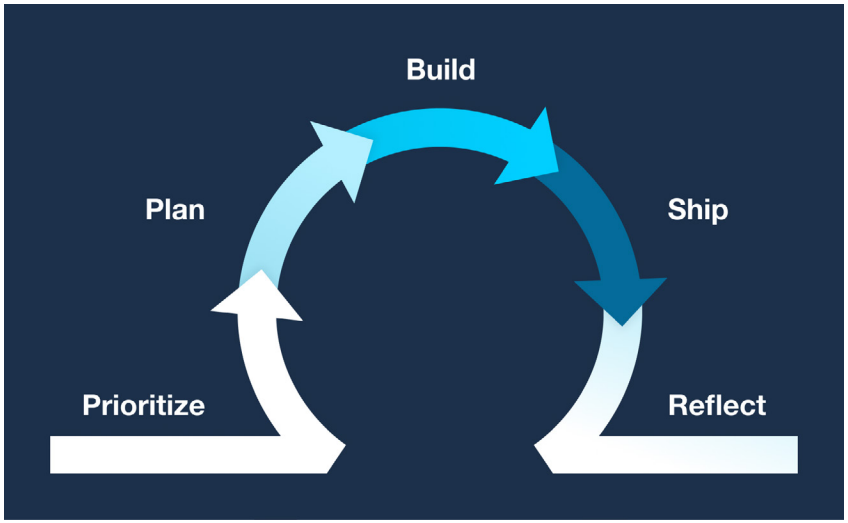
The role of user research

Once a vendor team starts, it can be tempting to dive straight into software development in order to show progress. A better place to start is for the team to become familiar with and invested in the needs of the system's intended users.

Even if user research has been done before the vendor joins the project, conducting a round of user research when the vendor starts is helpful. There are always questions about user behavior to address. It also establishes frequent research as a norm that always informs the next set of product decisions.

It may feel risky or inefficient to involve the whole team in user research, especially a new vendor team. But it helps the whole team gain critical context, understanding, and empathy. People will learn the most from direct user contact, which provides nuances and details that may not be apparent from a summary. Participating in user research helps teams make better decisions about the software and results in less rework later.

Understanding the software development cycle



Stage	Description
Prioritize	Government decides what work the vendor should focus on in a sprint based on discussion with the vendor team, user research, and stakeholder input.
Plan	Vendor and the government lead divvy up and assign tasks and agree on a "definition of done."
Build	Vendor team does the work and uses automated and other quality assurance tools to ensure code quality at deployment.
Ship	Vendor team delivers the work to the government, including all code and other artifacts. Government leads review for adherence to the quality indicators and definition of done.
Reflect	Government and vendor team review the work done in the previous sprint, discuss what worked, and what will need to be tweaked in the next sprint.

One of the most important roles the government product owner plays in a software development project is working with the team in each sprint to decide what to do next and to evaluate what's been done. This involves frequent interaction via well-structured meetings which

is one reason many teams begin work with an existing set of roles and meetings like Scrum.

To decide what to do next, product owners work with a team to define small pieces of work that it can work on independently. The "small pieces" are commonly written as "user stories," which capture what a user is trying to do and why. The process of fleshing out stories is critical to aligning the product owner and team on the work to be done and clarifying its connection to goals.

User stories should be accompanied by a "**definition of done**," the criteria for when work on a backlog item can be considered "done." This enables a vendor team to work independently and meet the product owner's expectations for completeness.

At the start of a sprint, the product owner and vendor team agree on the next stories to work on. Then, at the end of the cycle, the team should demonstrate completed work, even if it won't be directly experienced by the user.

Having the vendor team demonstrate the product regularly is an indispensable part of healthy oversight and good product leadership. Along with the quality indicators discussed below, "demos" are the only way the government can be truly confident the vendor's work is on track. They also help to avoid late surprises.

Based on the demo, the product owner can then agree if it meets the definition of done or if more work and further clarification of the work is needed. If the work doesn't meet the criteria to be considered done, it's not a moment for blame, but for discussion and improving how you communicate and collaborate with the vendor team. After all, it's impossible to anticipate all aspects of the work ahead of time. Ask questions like: "Was something missing from the definition of done?" "Is the team lacking essential context?"

MAJOR DECISIONS

Some decisions have risks or impacts beyond what a vendor is pre-authorized to decide. These decisions can include issues like whether to use a new third-party component, what data the system should store, or how a new capability should integrate with existing systems.

The government product owner or technical lead will likely be able to decide some of these on their own. But you will often need to work with the vendor about issues that involve established processes and other government stakeholders. It is the government's responsibility to consult with stakeholders and ensure that the implications of decisions are surfaced.

Whether the product owner or another agency representative makes a decision, it's good practice to document major decisions in an Architecture Decision Record (ADR), sometimes just called a "decision record." This tool helps maintain the system, as well as communicate the decision and any associated risks to stakeholders.

Major decisions come up frequently at early stages of projects and then in waves, such as during release planning. They should be expected. If a vendor team isn't flagging major decision areas, the product owner should bring it up in a meeting with the team and the contract administrator.

RELEASING THE SOFTWARE

When the software is ready for use, it's a good risk management practice to release it to a small group of users first before rolling it out to more people.

It's important to discuss a rollout strategy for release early in development. Deciding who will use the product first will inform choices about what to build first and in which order to deliver capabilities, such as a whole payment flow or household registration process. The rollout strategy may also impact decisions about how the system captures and stores data.

As you get closer to release, you may need to flesh out responsibilities for compliance, release, and operations between the vendor team and agency. Oftentimes, the vendor team will need to interact with agency teams at this point, like operations or a help desk. You may need to get involved in these discussions to ensure that all of the teams are getting the information they need for a successful launch.

Before release, many agencies require hands-on or "**user acceptance testing**," where intended users test the product's features and functionality. But, hands-on testing doesn't need to wait until all of the software is complete. It's better to test a set of capabilities of the software or process before the team moves on. The earlier functionality is tested, the simpler it is to localize and fix issues.

It is common for the product owner to help coordinate testing and work with the team to ensure that the tests adequately exercise the system. It is also likely you will test the system yourself. These tasks help de-risk the project and help you and other government stakeholders gain confidence in the software before it's released.

EVALUATING THE SOFTWARE

After release, a product owner's work isn't over. Systems don't always perform as they should. They may even cause unexpected problems

in the processes they're part of. A key part of the product owner's role is to evaluate if the system delivers on the project goals over time. Information collected after release is also essential for making ongoing development plans.

As with a rollout strategy, it's important to have an evaluation plan in place before release to ensure the information needed to evaluate the product against the goals is being collected. The vendor may be able to help with evaluating how the software is performing, but the government is ultimately responsible for doing that work.

Setting up the vendor relationship

Along with a strong understanding of product leadership, setting up the government-vendor relationship is key to effective vendor management. This begins in the project kick-off meeting and involves being ready to onboard the vendor team and establish healthy patterns for working together.

PROJECT KICK-OFF MEETING

Once the contract is awarded, the first interaction between the government and the vendor takes place at the post-award orientation meeting — the **project kick-off meeting**.

This is the first opportunity for the product owner and vendor team to talk about the product vision and desired outcomes for the project together. This is also the time to establish how the development team will operate as a unit by sharing expectations about communicating and working with each other, including discussing how particular issues and challenges will be resolved by the development team, government, or both.

Kick-off will involve a lot of new information for the vendor team. It's important to be thoughtful in designing this orientation so the team members have time to process and ask questions.

Consider spreading out orientation topics over a week rather than scheduling the typical half-day or full-day post-award orientation meeting. A multi-day format will allow more robust discussion of each topic area and allow time during and in between meetings to reflect, synthesize, craft follow-up questions, and strategize next steps. Each day should have no more than four sessions. Keep each one to 60 minutes or less to maximize information retention. (Review a sample kick-off week agenda.) Make sure to capture key orientation information in a written format, such as a README or guide, that the vendor can reference later. (Documentation will also help when onboarding new staff to the project.)

The project kick-off week should set the expectation for collaboration, so invite the vendor to lead a session of their choosing. They may want the opportunity to lead a team-building session or an open working session where they can ask questions about what they've learned.

It is important to limit attendance to the project kick-off week to only individuals that will participate in the day-to-day work. The week should serve as team-building for the newly formed team. Looky-loos or curious parties should not attend. However, it's a good idea to invite the project's executive stakeholder to the first meeting to stress its significance.

The week after kick-off, the product owner should schedule one-on-one meetings with each member of the vendor team. We encourage all team members to meet in this fashion, especially the product support and technical support on both sides so they can set up recurring meetings and establish open lines of communication early.

ONBOARDING

Onboarding a new vendor team involves getting them the access they need to systems, buildings, and government-furnished equipment. An agency typically has processes in place for a common level of access, but a custom software development project will often require a more specialized level of access. The process of getting it can be an arduous journey of paperwork, permissions, and authorizations and take a few weeks to a few months. During that time, the vendor can still bill the government for hours worked.

You can make the onboarding more efficient if you clear the "path to production" during solicitation so software development systems and deployment environments are in place by the time the contract starts. If that work has been done, onboarding is a matter of getting the team access to these environments as early as possible. If not, then that prep work needs to be done and comes with the additional issue of often enmeshing the vendor team in messy discussions about its need for access.

PATTERNS FOR WORKING TOGETHER

By the end of kick-off week, the government and the vendor team should know how they intend to work together, including:

- How often they will meet
- What will be covered at meetings
- Who will participate in each type of meeting
- Who will lead each type of meeting

As noted earlier, the government should convey the product vision during kick-off and check in about understanding of and alignment on vision and goals quarterly.

In a best-case scenario, you and the vendor team do planning and review activities together. Whether you use a Scrum-based approach or not, you'll need enough collaboration time to commit to and complete specific pieces of work in a one- or two-week cycle. You can expect to spend at least three hours per cycle on regular activities, including planning and review meetings in which the team:

- Frames work and discusses desired outcomes
- Agrees on work to be done next
- Reviews completed work together
- Interprets new information and feedback, and decides what to do about them
- Reflects on how to work better together as a joint team

Regular check-ins with the contract administrator

The product owner should set up regular meetings with the vendor's contract administrator (or whoever is the step above the vendor team lead) in case a situation requires escalation. These regular check-ins should be at least once a month and last no more than 30 minutes.

The point of a check-in is to establish a frequent and open channel of communication with contractor leadership to discuss the project's progress, successes, and challenges. The contracting officer or equivalent isn't required to attend, but it's a good practice to invite them for their awareness.

ESTABLISHING INTERNAL AGENCY COMMUNICATION AND COLLABORATION

The start of a contract is a good time for the government to set up ways to track progress, deliverables, invoices, spend rate, and project risks. Even though the designated contract administrator is contractually responsible for approving invoices and accepting vendor deliverables, the product owner and government technical lead are also critical because they will work with the vendor team on a daily basis and will have firsthand knowledge of the team's roles, responsibilities, and performance.

The contract administrator, product owner, and tech lead should be in frequent contact about the vendor's performance. Documenting the status of the contract and vendor performance in one shared document will also help make sure that all government roles are in the loop. Common data points that should be tracked during performance include:

- Invoice numbers and dollar amounts (to monitor spending levels)
- Deliverables and their respective quality indicators
- Any relevant materials or artifacts that the government and vendor team agrees are meaningful and valuable for tracking performance

Reporting the development team's progress to agency leadership is the responsibility of the product owner or contract administrator. Information that shows the vendor's actual progress, like software demos or communications that articulate the status of work toward outcomes or perceived risks, should be shared to minimize misunderstandings or miscommunication, and surface issues that may need to be addressed by all of those involved.

Reviewing the vendor's work

Good vendor management rests on the government's power and capacity to accept the vendor's work or ask for rework. This is much more than just quality assurance. Reviewing the vendor's work effectively requires understanding the software's intent and acceptable trade-offs. It also requires focusing on the software and not relying on reports about the project's progress.

A significant part of the review is checking that the software's functionality meets user needs. Review should also check the work against the project's "quality indicators," which enable the government to assess if a vendor development team's work meets the expectations for quality laid out in the contract.

QUALITY INDICATORS

In software development, "quality" is sometimes assessed narrowly as a lack of defects or bugs. A more productive way to approach assessing quality is to set clear, positive expectations upfront and monitor them continuously throughout a project. Doing so will enable you to build a high-quality product and maintain a healthy vendor relationship.

As explained earlier, quality expectations and indicators for your project should be incorporated into the solicitation and open to questions from vendors before they submit a proposal. Communicating these from the get-go lowers the risk of friction between you and the chosen vendor.

It is reasonable and consistent with private-sector tech practices to ask to see proof from your vendor that they are meeting expectations

for quality, so don't be shy about reviewing quality indicators. Think of them as the vital signs to be checked regularly during a project that help make sure the vendor is building high-quality and maintainable software.

In general, good quality indicators:

- Create a space for conversations between government and vendor to keep work on track
- Focus on necessary, user-centered work products
- Are grounded in common professional standards
- Rely on facts, not opinions
- Don't create additional work for you or the vendor
- Give the vendor freedom to meet the criteria in a variety of ways
- The vendor should be able to demonstrate they are meeting such quality indicators without additional work. Automation tools collect most of the necessary data by default.

How to monitor quality indicators

When

After the contract is awarded, the project's quality indicators should be reviewed at every sprint, usually every two weeks, as part of acceptance of vendor work.

Who

To avoid a conflict of interest, a government employee with sufficient technical knowledge should conduct the review. This may either be the agency product owner or a government technical lead assigned to the project, depending on the requirement.

Method

Indicators are evaluated using two methods: manual review and automated testing.

In a **manual review**, the government reviewer looks at the deliverable and judges whether it meets the standard set by the expectation stated in the contract. For example, the indicator "documented code" is not satisfied by the existence of documentation. The reviewer must judge if the documentation adequately explains the code.

Automated testing is done using tools that run tests every time code is submitted, or by some other trigger in the software development workflow. The vendor should set up the tests and provide the evaluator with the results from the testing tools.

After an initial evaluation of all of the quality indicators, the reviewer should talk to the vendor about the results, good or bad. This dialogue should align the reviewer's and the vendor's expectations and address early signs of problems or other concerns. Repeated failures to meet quality expectations should be documented and escalated to the appropriate contracting officials.

18F QUALITY INDICATORS

18F teams use the following quality indicators in our projects with agency partners. We recommend these as a minimum set that should be stated in a solicitation and contract for custom software development. They are presented and explained below in the form by which they're known at the federal level: Quality Assurance Surveillance Plan (QASP).

Each indicator is accompanied by a method or methods that make it easy to review and document. Expect a software vendor experienced in modern software development practices to be able to easily demonstrate they are meeting them at every sprint review.

Modify our set of indicators to meet the standards and requirements of your agency and project as needed. If, for example, your agency has more thorough requirements for testing accessibility, those are the performance standards you should use.

When writing a quality indicator, make sure it:

- States a performance standard(s) that is short and clear
- Is measurable on an ongoing basis
- Is work that lies within a vendor team's scope and capacity to control
- Is not a specific program outcome, such as reduction in processing times, payment accuracy, etc. (The government can't pass responsibility for program outcomes to a software vendor. A vendor can follow the program's assessment of how to generate outcomes, but the program is responsible for the ultimate results.)

Tested code

Performance standards	Acceptable quality level	Method of assessment
Code delivered under the order must have substantial test code coverage and a clean code base	Minimum of 90% test coverage of all code	Automated testing

Testing is an essential practice for developing functional software that performs well.

Developers write automated tests alongside their code that find flaws and/or verify that features function the way they were designed to. As code develops, tests are added so that future changes and additions run through the entire "suite" of tests. This practice ensures that revised and new code don't break features and functionality.

To meet this quality standard, a software developer must:

- Use automated testing tools
- Write automated tests for the code they develop
- Address the issues that surface in testing immediately

A developer can easily demonstrate that they're following these practices by producing summaries of the automated tests that show the code base passes all of the project's tests.

An important high-level indicator of quality in those reports is **code coverage**, or what percentage of the code base in the project is executed or touched by the automated tests. Code that isn't covered by any tests is a source of potential errors and a liability for future development. Expect a high threshold for coverage: 18F's standard is 90 percent. The 10 percent allows a buffer for how much code can be uncovered since full coverage is not always practical. For instance, the developers may have determined that a certain function should be tested manually.

Automated testing of code isn't perfect. New errors can get through even with code coverage and testing. When an error gets through automated testing, a developer must fix it and write an automated test for that particular error so that it is caught in the future.

If a developer can demonstrate they are regularly using automated testing that provides 90 percent or more code coverage and fixes

errors quickly, they have tools and practices in place for satisfying the quality indicator of tested code.

Properly styled code

Performance standards	Acceptable quality level	Method of assessment
Meeting acceptable quality level for this indicator	0 linting errors and 0 warnings	Styling standards and linters

Code style refers to established standards for writing and formatting a programming language. This practice maintains the readability and consistency of code so that it's easy to review and future developers can understand and maintain it.

Every programming language has its own code styling standards, much like there are various style guides for writing. To help them adhere to a code style, developers use code "linters," which test code against a style's rules and show code that needs to be changed to meet the chosen styling standard.

From a quality perspective, it is important that the vendor uses the chosen code style consistently and that styling errors or warnings caught by the code linter are corrected before the code is delivered and integrated into the product.

As with automated testing, a developer who is following these standards and using linters can easily and regularly produce output from the tool that shows there are currently no styling errors or warnings.

Review 18F's recommendation for linters for JavaScript and CSS.

Accessibility

Performance standards	Acceptable quality level	Method of assessment
Web Content Accessibility Guidelines 2.2 – 'AA' standards	0 errors reported using an automated scanner, and 0 errors reported in manual testing	Automated and manual testing

Note: Section 508 obligates federal agencies to make all their public-facing websites and digital services accessible. Many states have their own accessibility standards.

To meet full compliance with accessibility standards requires using automated and manual testing.

Developers of public-facing websites can check that their projects meet common accessibility standards by using an open source accessibility testing tool, like Pa11y. **Accessibility testing tools** run a series of automated tests on a site that detect accessibility issues.

Expect accessibility tests to be included in the suite of automated testing tools set up during a project and that you'll review their results every sprint. Integrating regular automated accessibility testing during development will keep the project on a path towards meeting this quality expectation.

Manual testing requires more effort than automated, which makes it impractical to do every sprint. An initial manual test should be done when the project's main functions and interactions can be tested. This initial review will set the base line for the project and often reveals a number of accessibility issues that need to be addressed.

To resolve the issues, prioritize them into categories of critical, moderate, and low priority:

- Critical issues pose serious accessibility challenges that will exclude users and should be addressed immediately.

- Moderate issues should be resolved within the next sprint.
- Low priority issues can be added to the project backlog and scheduled with other project tasks.

The development team should conduct a manual review for each major release to the project. These reviews should build on the base line and only test the portions of the project that have changed. The team should also document the review and remediation process for each accessibility issue in each phase of testing so there is an ongoing record.

Refer to the 18F Accessibility Guide for a comprehensive checklist and descriptions of accessibility issues and how to test for them.

Deployed

Performance standards	Acceptable quality level	Method of assessment
Code must successfully build and deploy into a staging environment	Successful build with a single command	Live demonstration

Modern development processes approach deployment of code through **continuous integration and continuous deployment (CI/CD)**. These tools create a development "pipeline" that automatically builds and deploys the project so it can be tested and then deployed to a production server that runs the public-facing site.

Automated CI/CD tools, which are integrated into version control systems like GitHub, make this practice possible. These tools and practices make it easier to maintain software and quickly make changes in response to user needs.

A development team can demonstrate it has set up the pipeline using CI/CD practices and tools if it is able to deploy a change to the

testing (also called "staging") or public-facing production environment at any time with just a single command. The deployment process should be comprehensively documented in plain language so it is understandable to non-technical agency staff.

Documentation

Performance standards	Acceptable quality level	Method of assessment
• All dependencies are listed and the licenses are documented • Major functionality in the software/source code is documented in plain language • Individual methods are documented in-line using comments that permit the use of documentation generation tools such as JSDoc • A system diagram is provided	Vendor provides documentation as specified in this section	Manual review

As the owner of the software created by the vendor, you need accurate and current documentation of the software so future developers can understand how it was built and why various decisions were made.

There are two types of documentation:

- In-line documentation is written into the code as comments that describe what specific pieces of code do.
- Supplementary documentation is written explanation of how the system works, its major functions, and any open source software "dependencies" required to run it.

- For maintenance, it's also important to document:
- Tools used during the project
- Software licenses for the tools
- How to get access to the tools
- Where log-ins are stored

This documentation is especially crucial if the system will be transitioned from the vendor to the agency.

The expectation for this quality indicator is that new code and documentation of it are written at the same time. It is most efficient to document new code as it is written and more likely to be accurate.

Security

Performance standards	Acceptable quality level	Method of assessment
Open Web Application Security Project (OWASP) Application Security Verification Standard 4.0.3	Code submitted must be free of medium- and high-level static and dynamic security vulnerabilities	Evidence of automated testing per OWASP

Make security testing a regular part of the sprint review process. Addressing vulnerabilities when they arise will reduce the risk that the project launches with significant security flaws. These practices should make it easy for a vendor to meet the hosting agency's security and compliance standards.

To check that applications are free from known security vulnerabilities, developers use open source, community-developed security standards like OWASP, and scanning tools that perform automated testing of applications against those standards.

Security scanning involves static and dynamic analysis. **Static scanning** refers to scanning the source code for vulnerabilities. **Dynamic scanning** refers to security tests of the application that determine if it is protected against common security vulnerabilities.

As with other automated tests, the vendor should be able to demonstrate the code in its current state doesn't have any vulnerabilities that are classified by OWASP as either medium- or high-level static or dynamic vulnerabilities.

Learn more about good practices for security in government.

User research

Performance standards	Acceptable quality level	Method of assessment
Usability testing and other user research methods must be conducted at regular intervals throughout the development process (not just at the beginning or end)	Artifacts from usability testing and/or other research methods with end users are available at the end of every applicable sprint, in accordance with the vendor's research plan	Demonstrated evidence of user research best practices

Designing human-centered software involves many decisions. A development team's decisions are better when they're informed by the perspectives of a system's intended users. This is why it's critical to know a team is conducting and making decisions based on evidence gained from user research throughout the entire project.

User research explores possibilities, tests assumptions, and reduces risk in a project by engaging frequently with end users. It includes qualitative and quantitative methods, including user interviews, usability testing, journey mapping and card sorting. It also involves

investigating tools and systems, and interacting with members of the public.

The research approach and methods used on a particular project will vary depending on the problem it's trying to solve, timeline, phase of the project, goals, and constraints.

When reviewing user research materials, processes, or deliverables, these are good signs that reflect the use of best practices:

- Recruiting from a diverse population
  - The team should be recruiting participants with a diverse range of perspectives, needs, and abilities. This helps ensure a product or service will be accessible to anyone who may use it. It's also important the team consider barriers to use and inclusion faced by various groups who may use the software and recruit people from those groups.
- Research plan(s) with clear and appropriate goals
  - Planning ensures that participants' and the team's time is respected throughout the research process. It also helps the team adapt its approach in response to real-world conditions. A research plan should include clearly stated and appropriate goals, methods, and research questions.
- The whole team is part of the research process
  - It's a good sign to see active team participation in research planning, observing research sessions, debriefing, and discussing the findings because it indicates shared investment in learning and serving the needs of users. (Every member of a team need not participate in every aspect of research.)
- Research participants' privacy is protected
  - When participants trust you, they are more likely to share full and accurate accounts of their experiences. A large part of maintaining trust with participants involves protecting their privacy. Signs that the vendor is protecting PII (personally identifiable information) include the use of pseudonyms, keeping access to raw notes limited, collecting informed consent, and de-identifying research data before synthesizing.
- Actionable research findings
  - After each round of research, the whole team should identify how the research findings change the work planned for the next sprint or for future design efforts. Articulating insights from findings involves various activities that allow the project team to work together to begin to map out larger patterns and themes.

Learn more about user research in the 18F User Experience Guide.

A note on code review

Code review refers to the common practice of developers regularly reviewing each other's code on a project. It is critical to maintaining consistency and quality on a project with many contributors. It allows reviewers to suggest improvements to the code and helps keep everyone on the team aware of what others are doing and how it may affect their own work.

Code review facilitates the manual review method of assessment required for the code-related indicators explained above. It also helps produce higher quality code with fewer defects.

While our sample QASP (included in the Resources section) doesn't include a specific quality indicator for code review, expect a vendor team to be engaging in this practice as part of its efforts to meet quality expectations.

You can ensure a vendor team engages in internal code reviews by asking a vendor how its developers approach them as part of the

proposal evaluation process. It may be done in a formal meeting. It may be done through version control systems like GitHub, in which developers review and approve new code and changes to existing code through "pull requests" before they are integrated or "merged" into the project's code.

The scope of a code review can range from addressing small issues to large ones. The only rule is that all new code or changes to existing code is being reviewed by at least one person before being merged.

It can be a challenge to establish a healthy balance of government involvement in a vendor team's code reviews. The "right" amount supports the flow of work and doesn't delay or block it. No involvement increases the risk that the project won't meet end user needs or critical design flaws won't be discovered until it is difficult to fix them. Too much can undermine the vendor's autonomy and motivation to produce quality code independently. For instance, if a government reviewer strictly dictates how something should be done and is not open to dialogue, it can lead to frustration and breakdown in communication. The level of government involvement also depends on the availability of staff with relevant technical expertise. (Consider hiring an independent contractor to act as reviewer if needed.)

Open and regular communication is the key to finding a healthy level. When government experts, or independent contractors working on behalf of the government, are able to participate in the code review process, discuss expectations about their level of involvement with the vendor at the start of work. Then, at every sprint review, proactively solicit the team's feedback about how that participation is going. Acknowledge and resolve issues before they harm the working relationship.

When technical expertise is not available on the government side to participate in code reviews, ask the vendor to confirm 1) that they are

conducting code reviews, and 2) to demonstrate, in the form of pull request discussions and approvals, that reviews are happening.

Learn about 18F Engineering's approach to code review.

Review an example of how to document the code review process in a government technology project.

Maintaining a healthy vendor relationship

There are common warning signs that your relationship with a vendor is becoming dysfunctional. These include when you and the vendor are:

- Paying more attention to processes and tools instead of individuals and interactions.
- Valuing comprehensive documentation over working software.
- Spending more time negotiating what the contract means instead of collaborating to deliver value.
- Fixating on initial plans instead of accepting change as an inevitable part of the work.

If any of these occur, don't immediately blame the vendor. These often appear when a vendor team is pressured by the agency to demonstrate they are following the rules of the contract. You can still get the relationship back on track.

HOW TO MANAGE AND RESOLVE CONFLICT

A “healthy” relationship with a vendor on a software development project will still involve conflict. When conflict does arise, it’s important to make it productive instead of destructive.

You can do that by always staying close to the work and maintaining good communication channels, which will help you detect issues early and address them before they become major problems.

Many conflicts with vendors are about performance and ultimately become conflicts over contracts. Contracts are legal documents that protect you and the vendor. To some extent they are also an attempt to predict the types of conflicts that could arise and to resolve them preemptively, or at least to provide an outline for resolution. As such, they are, essentially, relationship agreements. Contracts establish a framework for how the parties will work together. Unfortunately, they are terrible tools for managing software projects. Contract language is typically dense and hard to understand, intended to be difficult to modify rather than accommodating to changing needs, and is designed to meet the needs of someone in a legal or procurement role, not a software project team member. Consistent with the FAR’s guidance for disputes and appeals that govern all federal contracts, we recommend resolving issues without resorting to contractual claims.

We’ve found that if the government and vendor team are using the methods for communicating and demonstrating continuous progress that are outlined in this guide, contract claims almost never occur.

The product owner has the most responsibility for resolving conflict since they’re also responsible for maintaining the project’s speed. Some vendors also have an “agile coach,” or someone in a similar mediating role, on the team to help to unblock issues or deal with

conflict. The goal for government and vendor is to discuss issues as they arise directly and professionally and resolve them before formal dispute resolution is needed.

Here are some common issues we’ve seen arise in government-vendor relationships and how they’ve been addressed without resorting to contract claims.

Problems meeting the quality expectations

Monitoring quality indicators on a continuous basis is the most effective way to get ahead of issues. If the vendor misses one or more indicators, the way to resolve the issue is by discussing it with the vendor and documenting the conversations.

For example, if a vendor isn’t meeting code coverage expectations in early sprints, the technical reviewer should ask the vendor team in the next sprint review meeting what’s causing it to miss the acceptable quality level and discuss remedies together. Documenting the reasons for the problem and the actions the vendor will take in future sprints to correct it should be captured in one place, such as the code repository, for later reference.

Staffing misalignment

Avoiding staff misalignment on the vendor team begins during proposal evaluation, when the government should assess the reasonableness and rationale of vendors’ proposed staffing approach, including the team’s size and composition of roles.

Still, if the vendor team is using an iterative approach, the make-up of the team may need to change. For example, after a few sprints, the vendor team might realize there’s a gap in the skills needed to

maintain or increase its speed of work. Or user research might reveal a new priority for the project that requires new skills on the team.

Allowing for an adjustment in staffing is one reason we recommend using a time-and-materials (T&M) type contract with a maximum ceiling price for a custom software project. The T&M contract type enables the team composition and/or hours to be adjusted as long as the government and vendor agree it's necessary, and there isn't a major impact on the estimated ceiling price for the period of performance.

Staying close to the work enables the government to be able to interpret the reasonableness of proposed staffing changes. It also helps spot potential issues with staffing that should be addressed with the vendor, such as frequent turnover, which might be a sign of friction within the project team.

Turnover of key personnel

In a federal contract, the intention of a key personnel clause is to ensure a vendor staffs a team that has the necessary expertise and experience for a project.

If the vendor proposes a change to key personnel after the contract is awarded, the government should discuss the matter with an open mind. A change in leadership on the project might disrupt the flow of the vendor team's work, so it's important that the government and vendor discuss the impact of the change and work together to mitigate it.

Before agreeing to the change, the government should review the résumé(s) for the proposed replacement(s) or meet with them. *However*, the government can't participate in the vendor's hiring processes for a replacement, such as reviewing applications or sitting

in on interviews. Most agencies are restricted from "acting as an employer" to anyone on the vendor team because they lack "personal service" contract authority (refer to FAR 37.104).

Doing work outside priority order

Sometimes a team might work on issues in the backlog that are not in the order of priority set by the product owner. Whenever this occurs, the product owner should find out why.

There are many possible reasons. The team may not have understood the priorities. It may have disagreed with them. Or, it may have had a logical reason. For instance, the team may have discovered an issue that wasn't captured in the backlog but needed to be addressed before a prioritized issue.

If you find that the team didn't understand the priorities, discuss how your processes and communication can be improved. Clarifying and aligning on how priorities are communicated may be enough to address the issue.

If the vendor doesn't understand the priorities, you may need to share more context about the project or program-specific topics.

If the team doesn't agree with the priorities, it may be because they have important information you're not aware of that affects the functionality or integrity of the software.

If the vendor team understands the priorities but isn't attending to them, it may be a sign of a staffing issue that the vendor needs to address. For example, if the team repeatedly de-prioritizes an important task, it might indicate weak skills in a particular area or need for a separate workstream. As the product owner, your role is to

highlight the impact of the issue on the work and create space and motivation for the vendor to resolve it.

If the team is choosing to take on more tasks in a sprint than those selected as priorities, it's not a cause for concern as *long* as the top-priority tasks are being completed at a satisfactory rate. But, if lower priority work is drawing focus away from higher priority tasks, the product owner should address it with the team.

Making decisions outside the team's authority

Good vendor teams sometimes make decisions beyond their authority. Because every agency operates differently, it can be difficult for a new team to know which decisions they are free to make, which decisions need to be communicated along with implications, and which decisions are truly for the agency to decide. The important thing is to spot when this happens and clarify the boundaries of team decision-making.

Common areas where this issue comes up are:

- Tech re-platforming (such as introducing a new programming framework or data store)
- Reopening settled questions, especially wanting to redo user research
- Decisions that constrain launch strategy or operations

While it is the vendor team's responsibility to signal when its choices may have a wider impact, the product owner should strive to create an environment that encourages that communication by asking good questions and remaining engaged throughout the project. Although a product owner should be mindful of leaving decisions to the team that are in its purview, it's their responsibility to actively manage the

consequences of the vendor team's choices for people outside the team and to consider their long-term impacts.

Like with other challenges, the first thing a product owner should do if a team is making decisions outside its authority is to talk with the team. Escalation should only be a last resort. The product owner should work to clarify the types of decisions to be made and how they want to be involved in each. Decisions on these matters should be written into the document that captures the team's operating principles, such as a team charter.

A government custom software project is successful only if the software delivers on the intent of the particular policy or service it was designed for. That purpose must remain front and center at all times.

06 Conclusion

SUMMARY

Following the principles and practices in this guide will help agencies lower the risk that their technology projects will fail. It's not easy work, so it's important to start small and just try. And then to keep trying.

Today, complex technology systems deliver vital government functions that support people's quality of life. As the launch of HealthCare.gov and numerous other examples show, that function is easily stymied when agencies make ill-informed choices and use ineffective methods to acquire and develop software.

But, agencies can, and sometimes do, avoid those errors and deliver technology systems that serve their intended purpose. They do so by following the principles and practices collected in this guide.

These aren't new ideas. They are widespread in the private technology sector. They're not new to government either. Iterative development was used over half a century ago to deliver ambitious projects like the X-15 hypersonic jet, while many technical experts in government, like those on the Defense Innovation Board, know: "modern methods allow a project to continuously improve, adapt to evolving threats, and take advantage of rapid technology advances." Yet "modern methods" still aren't the norm in government.

Why? Because we are getting in our own way. In its interim report to Congress, a panel of 16 recognized experts in acquisition and procurement policy wrote of the Department of Defense: "Processes such as developing requirements, contracting, making investments, or obligating money are often driven not by a sound business case, but by arbitrary deadlines and outside pressures." It went on: "Both written rules and performance norms incentivize making decisions

that lead to suboptimal outcomes." The Department of Defense is not alone in its troubles. Policies, processes, and cultures that are resistant to change are common in government agencies.

Often this resistance is due to the weight of "outside pressures" that are beyond their control. But, what we can control as public servants is *choosing to approach government technology projects differently than we have in the past* and acting on the knowledge that *traditional attitudes and methods don't work and organizational inertia is harmful*. You can avoid "suboptimal outcomes" by applying the methods, tools, processes, and recommendations in this guide.

This work is not easy, but it is possible and it gets easier the more you do it. If you're struggling to figure out where to start, try beginning with a small project or small piece of a project. Or try using one method on a current project.

The important word here is "try." Adopting new practices and cultivating them within an organization takes trial, error, and time. Don't expect instant success. Each "mistake" will be a learning moment and opportunity to try something different. Legacy practices will likely coexist with new ones for a while. Celebrate small wins. Find champions of change and work together. Adapting to using iterative software development requires agility and openness in your attitude as well as your practices.

The real challenge is not following the practices in this guide. It's doing so again and again. Software is never done. It needs constant "care and feeding" so that it continues to work for people and our changing needs. The most difficult work is the commitment to doing the work despite its challenges.

Software will never be done

07 Resources

This appendix of resources includes:

- An in-depth guide to conducting market research
- An in-depth guide to evaluating proposals and bids, including an downloadable evaluator worksheet
- A set of best practices for open source software security
- Sample questions to ask a vendor during verbal interviews
- A sample agenda for a kick-off week with a vendor development team
- A sample Quality Assurance Surveillance Plan (QASP)

How to conduct market research

The goal of conducting market research is to understand the marketplace for a service or product before you enter into a contract of any type. Good market research prepares an agency to enter into a competitive bidding process, which is legally binding, with a clear understanding of the service or product and how to recognize quality.

Market research has two distinct phases and purposes:

1. **Market surveillance** is a continuous process to stay informed about industry trends, new technologies, and other information about a marketplace of goods and services needed to fulfill an agency's mission.

   For instance, if you were doing market surveillance research for user experience (UX) research and design services, you would ask questions like:

   - What is "user experience"?
   - What does the practice of user experience design consist of?
   - What makes for a good user experience?
   - What are the characteristics of developers of good user experiences?
   - What qualifications and experience do these developers have?

   Market surveillance is strategic. Consistent market research gives you a good grasp of accurate, relevant, and timely information about a market. This knowledge makes it easier for an agency to complete a market research report or an acquisition plan as needed.

2. **Market investigation** is research focused on specific sources, materials, or potential competitors to fulfill a particular agency requirement. It is usually done to complete a market research report for an active procurement.

   It involves more pointed questions than those used in market surveillance. If you were doing market investigation for UX research and design services, your questions would be:

   - Who has delivered products with a good user experience?
   - Where can I find a good user experience provider?
   - Are there professional associations or conferences for user experience?
     - Are there trade publications or other information sources about user experience?
     - What have they worked on? Have they worked with government agencies before?
   - Do any of these companies have existing contracts through an available Federal Supply Schedule (FSS), another Governmentwide Acquisition Contract (GWAC), or any kind of pre-established vehicle or framework that reduces acquisition effort without sacrificing quality in the final product?
   - Are any of them under a recognized socioeconomic program or status, such as the 8(a) program or Service-Disabled Veteran-Owned Small Business (SDVSOB) program?

While it's common to think that market surveillance always comes before market investigation, they often happen in parallel because each informs the other. Use both market surveillance and investigation to understand what is available.

Before starting, keep in mind that market research shouldn't identify a preferred or specific manufacturer, model, or brand. Doing so

eliminates all of a buyer's negotiating power. Market research is a *forecasting exercise.* It can't be used in place of the government's source selection or evaluation process to determine a contract award. It also can't favor a specific vendor that may eventually be awarded a contract. So, don't rush from market surveillance to investigation and forget to continue surveillance. Fight the urge to pick a single brand name or company.

Also, expect things to change. Needs often change by the end of a market research process.

SOURCES OF MARKET INFORMATION

As a best practice, government buyers should rely on many primary and secondary sources of information.

Primary sources include:

- Vendors
  - Manufacturers
  - Distributors
  - Resellers
- Other buyers
  - Private sector
  - Other agencies
    - Colleagues
  - Nonprofit organizations

- Independents
  - Experts
  - Specialized consultants
  - Research companies

Secondary sources include:

- White papers or similar position statements
- Trade journals
- News reports
- Academic journals
- Subject-matter literature
- Databases
- Case studies

ENGAGING WITH SOURCES

Agencies can engage directly or indirectly with a source.

Direct contact is when an agency communicates with a source in conversation or writing. A source may provide a lot of information, but the agency should weigh this information carefully since the source may be a future competitor.

Indirect contact is when a researcher reviews material without directly engaging a potential future competitor or talks with an impartial party.

Most market research should be done indirectly. It is easier than direct and less prone to the risk of creating bias towards a vendor.

REQUESTS FOR INFORMATION (RFIS)

Requests for Information are a popular market research tool. Often they're the only market research an agency conducts before awarding a contract.

RFIs have a place in market research, such as when the government truly has no idea how it could solve a problem or satisfy a need. But more reliable information can be found through indirect research on the internet.

Before using an RFI, consider that:

- RFIs usually consist of a set of questions that can't respond to changing agency needs. They're not a dynamic, evidence-based form of inquiry that develops over a period of time.
- Responding to an RFI is a lot of work for most businesses, especially small companies and companies new to competing for government contracts.
- Most RFIs are made up of boilerplate marketing material — regardless of the agency or topic.
- RFIs increase the likelihood of a protest because even though vendors reuse content, they are labor-intensive and often expensive for a company to prepare.

MITIGATING THE RISKS OF VENDOR SALES AND "CAPTURE MANAGEMENT"

The U.S. government is the largest buying entity in the world. In general, government agencies are primary targets for vendor sales and "capture management" or "capture planning," in which a company tries to gain an advantage for winning a contract.

Common sales tactics include:

- **Cold contact**: Someone you've never met calls or emails you about their company and offerings and how they can help you.
- **Name-dropping**: A salesperson tries to gain influence with you by mentioning the name of someone higher up in your agency than you or suggesting they've talked with someone in your agency with influence over the project.
- **Networked introduction**: The vendor develops a good reputation with one customer and then asks that customer to introduce or refer them to other potential customers within an organization.
- **Big pitch**: The vendor engages in a broad or organized effort to present to a large group of agency staff to nurture excitement and interest in buying their company's product, service, or other offering.

While salespeople can help educate potential customers about their company's offerings and reflect on their needs, sales capture is rife for potential abuse.

Government employees work on behalf of the American public and have ethical and professional standards by which they must conduct themselves. Numerous laws and regulations also guide the communication or actions of government employees.

Employees that don't adhere to these standards, such as guidance provided in FAR 9.5 for Organizational and Consultant Conflicts of Interest, may face civil or criminal penalties. Depending on the violation, this could mean fines, suspension, firing, and even felony prison time.

Those standards, laws, and regulations shouldn't deter government employees from interacting with vendors as part of market research. They provide needed protection from aggressive sales tactics.

Keep all interactions with salespeople professional, transparent, and courteous. And keep in mind:

- Any information shared could directly affect that vendor's preparation of a proposal. By law, *all* vendors that could fulfill the agency's requirements *must have the exact same information*. If one vendor is given more or different information they may gain an unfair competitive advantage. If information isn't shared consistently, it could lead to a protest.
- All government personnel have a responsibility to protect proprietary or confidential information and not share it with companies or potential competitors.
- Government personnel must avoid the appearance of commitment before the contract is awarded. Only a person that is officially delegated with the authority to award and sign contracts can obligate the government to an agreement with a contractor.

To avoid giving one vendor more or different information than others:

- Start every conversation with a disclaimer like:
  - "Nothing discussed in this meeting authorizes you to work, start work, or otherwise obligates the government. This conversation is only for market research purposes. Any assumption on your part or on the part of your company is a mistake and has no effect on the government."
  - "We are talking for market research purposes only. This conversation in no way obligates the government or should make you believe that we have entered into a contract of any kind."
- Treat all potential bidders fairly and impartially.
- Imagine that all interactions with vendors have a public audience. To assess the fairness of a potential action, consider if an impartial, casual observer would believe you, as a government employee, acted responsibly and reasonably.
- Reach out to experienced procurement professionals to learn best practices for conducting interactions, documenting exchanges, and developing requirements for competitive solicitations.

How to evaluate proposals and bids

It is common for agencies to use a scoring scheme to evaluate vendor proposals and bids. We recommend a different method that creates a detailed and defensible justification of the government's vendor selection, which a scoring scheme does not. It also allows the government to give feedback to the vendors that didn't receive the award by simply summarizing the proposal's documented pros and cons.

As you will have explained in the solicitation for a performance-based services contract, this approach regards the three technical evaluation factors — technical approach, staffing approach, similar experience — combined as significantly more important than the price in evaluating the strength of a proposal.

Following are evaluation criteria for each of those technical factors. Each set includes positive signs and red flags to look for as you review proposals. They aren't exhaustive, but should help an evaluation team get started and decide which vendors to interview.

Use our evaluator worksheet as a tool during the review process.

TECHNICAL APPROACH

Ideally, the vendor proposes to use modern software development practices. The proposed approach should be appropriate for the scope of work and demonstrate technical proficiency.

Evaluate answers from the verbal interviews as part of the technical approach.

Competency

Positive signs

- Demonstrates knowledge of their preferred tools and methods, and is able to explain why they are appropriate for the project

Red flags

- Misidentifies core technologies in a way that shows inexperience communicating about or using them
- Proposes a highly complex approach or uses highly complex language that confuses rather than clarifies
- Proposes to outsource core technical competencies
- Doesn't mention using secure code practices
- Doesn't value testing code

Lack of novelty

Positive signs

- Recommends established software and infrastructure, as well as use of proven and effective design patterns

Lack of certainty

Positive signs

- Highlights areas of uncertainty in their technical approach (Since a vendor can't know if a proposed approach will be effective until development begins, they should be candid that they can't be sure.)

Vision

Positive signs

- Interprets the intended outcomes in a way that can enable the agency's vision

Program goals

Positive signs

- Demonstrates a clear grasp of the agency's mission and project's aims described in the solicitation

Red flags

- Doesn't understand program goals that were described clearly in the solicitation

Open source software

Positive signs

- Has experience developing open source software

Red flags

- Doesn't have experience developing open source software

Collaboration and communication

Positive signs

- Expects to work with an agency product owner and for that person to be an active team member — one who communicates proactively about risks and roadblocks

User research

Positive signs

- Expects to conduct regular and ongoing user research to understand user goals and needs, and to use research findings to build features that support those goals and needs
- Includes how qualitative and quantitative data will be leveraged to inform product and design decisions
- Has a plan to conduct user research and test everything from rough prototypes to finished software with actual users throughout the entire design and development process
- Seeks research participants from diverse backgrounds.
- Describes target groups for research
- Research will be done with people who will actually use the service, ideally people with diverse perspectives and differing abilities
- Research plan involves people:
  - Who have disabilities or use assistive technologies
  - With limited digital skills or low literacy
  - Who may need help using the service in question
- Research plan mentions:
  - Respect for participants
  - Informed consent
  - Potential harms and how they will be reduced
  - Diversity, inclusion, honesty, and transparency
- Research plan methods are appropriate and the timeline is feasible
- Combines user research with usability testing to ensure that features are meeting user needs

Red flags

- Doesn't indicate that they will use user research to determine the design or the technical approach
- Proposes a process that includes working for long stretches of time without interacting with the agency and/or users
- Proposes using focus groups instead of structured one-on-one research interviews or usability testing sessions
- Doesn't use research methods appropriate to research goals (e.g., using surveys to uncover user needs or usability testing to validate user goals)
- Design is described as User Acceptance Testing, performed only at the end of a project
- Displays low maturity in UX research and design practices:
  - Research goals, questions, methods, and expected outcomes don't align
  - Doesn't understand the difference between users and stakeholders
  - Doesn't provide a user recruitment approach or interview protocol provided

User-centered design

Positive signs

- Follows a user-centered design process (They explain how they make design decisions in relation to broader user goals and specific needs learned through user research.)
- Indicates that design is considered part of the cross-functional agile development team — it doesn't operate in a silo

Red flags

- Proposes that requirements will be collected from the business owner, rather than determined according to user needs uncovered through research
- Prioritizes aesthetics over usability and usefulness
- Can't explain their design decisions

Development infrastructure

Positive signs

- Focuses on automation, reliability, testability, infrastructure as code, etc.
- Refers to modern automation and deployment tooling like Jenkins, Puppet, Chef, Travis CI, CircleCI, Kubernetes, Terraform, AWS, and Heroku

Accessibility

Positive signs

- Offers specific, detailed description for how the team will build accessibility and testing into the development process
- Lists applicable, up-to-date government accessibility standards

Red flags

- Doesn't mention accessibility or explain how they will evaluate if their software meets accessibility standards
- Offers "shall comply" without citing specifics, such as Sec. 508 and the protocol for satisfying it

Other

Red flags

- Bypasses page-limit rules in their proposal by using a tiny font size, reduced leading, etc.
- Proposes long-term staff augmentation

STAFFING APPROACH

You want evidence that the staff has experience in their areas of expertise.

In addition, if the developers have presences on social coding platforms (for example, GitHub, GitLab, Bitbucket), review them to consider:

- What kinds of projects have they worked on?
- What languages have they worked with?
- Is their code readable?
- Does their code follow best practices for organization?
- If their projects are open source, are they being actively used or forked?
- Do their projects show expertise that doesn't appear in their qualifications?

Team size and roles

Positive signs

- Fewer than 10 team members, each of which has a clear role and purpose

Red flags

- Specifies too many key personnel, especially with individuals whose expertise overlaps with that of agency staff
- Over-staffs the bid (If a vendor proposes a team that consists of people with far more experience than necessary, or more people than necessary, it suggests they either don't understand modern software development practices or are just trying to over-staff the engagement.)
- Under-staffs the bid (A vendor might try to win the bid by proposing a smaller team than it knows is needed for the project, with the plan of increasing the size of the team later.)
- Proposes positions that aren't needed in an iterative development project, such as business analysts, enterprise architects, delivery managers, etc.
- "Access to a database of resumes" is provided, but specific technical staff are not named

Team capacity

Positive signs

- The team will be assigned to the project full-time and won't split members' time with other projects (Developers, user researchers, designers, and all other key personnel should be fully staffed. A Scrum master or agile coach can be exceptions.)

Red flags

- The most qualified team member is allocated a small amount of time on the project
- Proposed staff don't currently work for the contractor and a letter(s) of intent from the proposed staff is not provided

- Key staff aren't proposed to be full-time on the project, or the project is to be staffed with mostly partial full-time personnel

Technical team's specialized experience and knowledge

Positive signs

- Experience with modern software languages, such as Python, Ruby, PHP, C# (C Sharp), or JavaScript
- Experience with web-based application programming interfaces (APIs), especially REST and GraphQL
- Experience using Git for software version control
- The lead developer's skill set and experience will enable them to conduct the work required by the project

Red flags

- The proposed lead developer lacks sufficient qualifications
- Proposes outdated software technologies that don't have an active developer community, e.g., ColdFusion, ASP, or FoxPro
- Lack of experience with test automation, aka DevOps or test-driven development (TDD)
- Proposed staff qualifications are copied in large part or completely from the internet
- Key skills don't appear in any qualifications, such as:
  - Agile development experience
  - Automated (unit/integration/end-to-end) testing
  - Continuous Integration and Continuous Deployment
  - DevOps
  - Application Protocol Interface (API) development and documentation
  - Open source software development

  - Cloud deployment
  - Building and testing public-facing sites and tools

Research, design, and product team members' specialized experience and knowledge

Positive signs

- The lead user researcher's background demonstrates:
  - Understanding of how research can inform and shape strategy, design, and development
  - Familiarity with a variety of user research and usability testing methods
  - Experience deciding the method or methods to use that suit a given research question
  - Experience recruiting research participants appropriate to a project
- The lead UX designer's background demonstrates:
  - Strong craft skills and experience generating concepts that reflect overall project strategy, user research, and user-centered design best practices
  - Experience and ability communicating those concepts visually via a variety of methods, including sketching, wireframing, prototypes, and more polished mock-ups

Red flags

- The company, proposed subcontractor, or proposed staff are responsible for poorly designed websites

- Key skills don't appear in any qualifications, such as:
  - Product management and strategy
  - User research, such as contextual inquiry, stakeholder interviews, and usability testing
  - User experience design
  - Sketching, wireframing, and/or prototyping, and user task flow development
  - Visual design
  - Content design, UX writing, and copywriting

SIMILAR EXPERIENCE

As part of the solicitation, you will have asked vendors to submit code repositories for projects that are similar in size, scope, and complexity to what the agency needs. If you do not have someone on your evaluation team that is familiar with code repositories, you should find a technical advisor.

Technical evaluations

Positive signs

- Proper use of Git, commit changes with personal accounts (not organizational)
- Use of a branching or merging strategy
- Informative comments
- Evidence of peer code reviews and collaboration (work was performed in a reasonable number of GitHub comments)
- Use of a CI/CD pipeline

- Code that conforms well to the quality expectations in the solicitation's QASP or set of quality indicators
- Substantial projects: the projects weren't created just to have something to point to for this solicitation
- Iterative incorporation of user feedback into their development process
- Demonstrates the value of testing:
  - Testing is built into the development process
  - Code tests are written well, test coverage is measured and covers most of the code
- Use of consistent code style
- Code displays modularity and opportunities for reusability
- Sensible data model approach
- Code includes evidence of accessibility considerations (e.g., appropriate alt text, ARIA attributes)
- Evidence of accessibility testing: at minimum, an automated scan; more importantly, manual testing
- The project is set up to be easily deployable by any newly onboarded developer

Red flags

- No source code is submitted
- There is no Git history or only a single commit, which indicates that this is not the actual code repository and that the code was developed somewhere else (maybe not even with source control)
- None of the provided code samples or described projects are similar in size, scope, and complexity to the project scenario in the RFQ

- The code samples provided do not demonstrate an understanding of writing a modern, maintainable application
- Code is undocumented; there are no code comments
- No automated tests
- The code has obvious vulnerabilities for attacks (e.g., missing SSL certificates, SQL injection attacks, credentials checked into the code, use of unvalidated JWTs)
- Tests are disabled, which suggests that developers may have turned testing off instead of fixing errors; there seems to be a practice of deleting tests or code until the code passes
- Code appears sloppy; there are large sections commented out, unused imports and definitions, or dead code (code that is in the project but is never used)
- No instructions for setting up the project or documentation is boilerplate (e.g., a README)
- Code contains secrets such as passwords, personally identifiable information, or access tokens
- The cited projects lead you to suspect the vendor didn't create them
- There's a finished product, but no code, or vice versa

Programmatic evaluations

Positive signs

- Work that is conceptually similar to the agency's needs
- Work that is centered on user needs
- Work that was completed by a team of a size similar to the size of the team that they're proposing

- Design artifacts that show continuous and ongoing usability testing and that indicate a user-centered approach to iterative design and development
- Illustrates getting stakeholder buy-in on research findings
- Demonstrates that they are comfortable with complexity and challenges
- Communicates openly and emphasizes transparency
- Identifies what is important to each set of stakeholders and tailors their approach accordingly
- Describes frameworks and tools that support iterative development, constant improvement, user-centered design, risk management, and product prioritization

Red flags

- The cited projects aren't similar in size, scope, or complexity to that described in the solicitation
- Work that is led by solutionism
- The projects don't include design artifacts and research plans, or the plans are incomplete
- The projects don't include design artifacts and research plans, or the plans are incomplete

Evaluator worksheet

Evaluator name: _____

Offeror or vendor: _____

Date of evaluation: _____

Have you signed and returned your COI and confidentiality forms?

From the RFQ:

> The non-cost evaluation factors are of equal importance. The three (3) technical, non-price evaluation factors when combined, are significantly more important than price. The government may make an award to an offeror that demonstrates an advantage with respect to technical, non-price factors, even if such an award would result in a higher total price to the government.
>
> - Factor 1: Technical approach
> - Factor 2: Staffing plan
> - Factor 3: Similar experience

QUALITATIVE EVALUATION

We will be doing narrative, qualitative evaluation. Quotes will not be scored numerically.

Therefore, it is critical that we evaluate the quotes based on what we put down in the solicitation.

How do you evaluate a proposal qualitatively?

- Provide as thorough of a narrative description as you can on this worksheet.
- Base your decisions on the factors and descriptions identified in the solicitation.
- Use common sense to consider real-world implications. Imagine your, or your agency's, day-to-day work needs.

Do's and don'ts

Follow these tips when evaluating quotes. Please refer to your contracting officer with any questions.

Do's	Don'ts
Do evaluate quotes against the solicitation requirements.	**Don't** make assumptions. Evaluate the text in the tech quote and do not rely on outside information for technical evaluations.
Do look carefully at the text in the technical quote that may include statements and/or assumptions that could indicate increased cost or price and/or risk to the government.	**Don't** compare proposals against one another. They must be evaluated individually against the evaluation factors in the solicitation.
Do adequately document your reasoning for any potential increased risk to the government on the evaluation form for the contracting officer's review.	**Don't** rank or compare quotes. Only determine if they meet, or do not meet, the acceptable standards specified in the solicitation.
Do provide comments that are clear and plainly written.	**Don't** take it easy or be overly harsh. Fairly evaluate all proposals against the requirements of the contract. Be critical, but fair in your evaluation.
Do be fair and consistent in the proposal evaluation. If an item is a strength or weakness for one proposal, it should also be noted as a strength or weakness when it appears in other proposals.	**Don't** consider price when evaluating technical quotes. These evaluations should be completed separately from each other.

Strengths and weaknesses

For each evaluation factor — technical approach and staffing plan, key personnel, and source code — we will evaluate and analyze strengths and weaknesses that will be used as the basis for the confidence ratings (high, some, and low) for each factor.

A **strength** is an attribute that, within the confines of the evaluation criteria, would raise the evaluation above neutral.

A **weakness** is an attribute that, within the confines of the evaluation criteria, would reduce the evaluation below neutral.

For each strength and weakness you identify in a quote, use words that qualitatively describe that strength or weakness in narrative form. For example:

> **Strength**: On page X, contractor two states that they offer their employees two weeks of paid time off to attend training sessions every year. This encourages retention and staff growth which is important to the government to maintain a consistent level of service to their internal and external customers. This also allows the contractor to provide the most qualified and trained staff.
>
> **Weakness:** Contractor one, page 10, paragraph four. The contractor does not appear to understand the direction of the program nor the intent of the contract and has specified an approach that has proven unsuccessful on this program in the past and that was communicated in the solicitation.

Don't write narrative explanations that are vague or reflect subjective opinion. For example:

> **Weakness**: Contractor one's approach to training is overly burdensome for the government compared to contractor three's.
>
> **Weakness**: The technical proposal doesn't address what we asked for.
>
> **Strength**: I really like what contractor one wrote. It's exactly what we're looking for.

Confidence ratings

Once you've identified strengths and weaknesses for each factor, you'll assign a confidence rating to the factor as defined below:

High confidence: The government has high confidence in the portion of the quotation, and that the risk to the government is low.

Some confidence: The government has some confidence in the portion of the quotation, and that the risk to the government is low or moderate.

Low confidence: The government has low confidence in the portion of the quotation, and that the risk to the government is moderate or higher.

For any questions, concerns or comments, please do not hesitate to ask your contracting officer.

You should also refer back to the solicitation if you are unsure or do not understand any portion of it.

EVALUATION FACTOR 1: TECHNICAL APPROACH

From the RFQ:

INSTRUCTIONS TO OFFERORS:

The technical approach must set forth the contractor's proposed approach to providing the services required, including the base software (if any) and programming language(s) the contractor proposes to use. The technical approach must also make clear that the contractor understands the details of the project requirements. The technical approach must also identify potential obstacles to efficient development and include plans to overcome those potential obstacles. The technical approach must also include a description of the contractor's plans, if any, to provide services through a joint venture, teaming partner, or subcontractors.

EVALUATION BASIS:

In evaluating a contractor's technical approach, the government will consider (a) the quality of the contractor's plans to provide the open source, agile development services required, including user research and design, (b) the extent of the contractor's understanding of the details of the project requirements, and (c) the extent to which the contractor has identified potential obstacles to efficient development, and has proposed realistic approaches to overcome those potential obstacles.

EVALUATOR SECTION – FACTOR 1

Strengths:

Weaknesses:

Factor confidence rating:

Comments and questions:

EVALUATION FACTOR 2: STAFFING PLAN

From the RFQ:

INSTRUCTIONS TO OFFERORS:

The staffing plan must set forth the contractor's proposed approach to staffing the requirements of this project, including the titles of each of the labor categories proposed and proposed level of effort for each member of the contractor's development team (full time, half time, etc.).

The staffing plan should identify the proposed qualified individuals for the three (3) key personnel.

Contractors proposing key personnel who are not currently employed by the contractor or a teaming partner must include a signed letter of intent from the individual proposed as key personnel that they intend to participate in this project for at least one year. The staffing plan must also set forth the extent to which the proposed team for this project was involved in the development of the source code referred to in the next paragraph.

The staffing plan must set forth and explain the extent to which the contractor will provide individuals with experience in most the following areas:

- Agile development practices
- Automated (unit/integration/end-to-end) testing
- Continuous Integration and Continuous Deployment (CI/CD)
- Refactoring to minimize technical debt
- Application Protocol Interface (API) development and documentation
- Open source software development
- Cloud deployment
- Open source login and/or authentication services
- Product management and strategy
- Usability research, such as (but not limited to) contextual inquiry, stakeholder interviews, and usability testing
- User experience design
- Sketching, wireframing, and/or prototyping, and user-task flow development
- Visual design
- Content design and copywriting
- Building and testing public facing sites and tools
- User outreach and/or user adoption
- Database design and SQL queries
- Security and compliance

In addition to these baseline skills, you must also provide information about your recruitment, retention, and training for your personnel as the needs of the individual team composition may change over time during the course of development.

To understand your approach to recruitment, identify and provide an adequate description of your strategy to find qualified personnel generally and for the proposed personnel in your quote submission. As part of this, please provide an explanation of the process undertaken to ensure proposed employees staffed in each labor category meet the specific qualifications and have the requisite skills for the position. To understand your approach to retention, identify and provide an adequate description of your strategy to minimize staff turnover.

EVALUATION BASIS:

In evaluating a contractor's staffing plan, the government will consider (a) the skills and experience of the key personnel and other individuals that the contractor plans to use to provide the required services, (b) the mix of labor categories that will comprise the contractor's proposed development team, (c) the contractor's proposed number of hours of services to be provided by each member of the contractor's proposed development team; and (d) the contractor's approach for recruiting and retaining qualified personnel.

EVALUATOR SECTION – FACTOR 2

Strengths:

Weaknesses:

Factor confidence rating:

Comments and questions:

EVALUATION FACTOR 3: SIMILAR EXPERIENCE

From the RFQ:

INSTRUCTIONS TO OFFERORS:

You shall submit two (2) source code repositories.

This must be either links to Git repositories (either credentialed or public) or to equivalent version-controlled repositories that provide the evaluation team with the full revision history for all files. If a contractor submits a link to a private Git repository hosted with GitHub, the government will provide the contractor with one or more GitHub user identities by email, and the contractor will be expected to promptly provide the identified user(s) with access to the private Git repository.

The source code samples should be for projects that are similar in size, scope, and complexity to the project contemplated here. The source code must have been developed by either (1) the contractor itself, (2) a teaming partner that is proposed in response to this RFQ, or (3) an individual that is being proposed as key personnel for this project. The government would prefer that the source code samples have been for recent projects involving teams of approximately four to seven full-time equivalent (FTE) personnel.

If the references to source code samples provided do not include associated references to user research plans and design artifacts demonstrating how ongoing user research was incorporated into the project, then the contractor must submit a user research plan and design artifacts relating to at least one (1) of the source code samples provided.

EVALUATION BASIS:

In evaluating a contractor's similar experience, the government will consider the extent to which the contractor has recently provided software development services for projects that are similar in size, scope, and complexity to the project described in this RFQ, and the quality of those services. In evaluating the quality of those services, the evaluation team will consider, among other things, the revision history for all files in the source code samples provided. The government will also consider the user research and design-related artifacts that were associated with the source code samples provided or submitted separately. In considering a contractor's similar experience, the government may also consider information from any other source, including contractor's prior customers and public websites.

EVALUATOR SECTION – FACTOR 3

Strengths:

Weaknesses:

Factor confidence rating:

Comments and questions:

Best practices for open source software security

This is a high-level explanation for keeping data, static assets, secrets, and code safe in an open source project. Always work with your security team to make sure your project aligns with your agency's requirements.

Learn more about secure implementation of open source software in the DoD Open Source Software FAQ.

KEEP DATA SAFE

- Keep data, such as page content or form responses, in a database.
- Ensure data is not shared when source code is published.
  - To do that, don't hard-code data into the code and use a database or API for data retrieval.
- Databases need to be encrypted at rest and enable necessary logging.
- Manage your databases by routinely checking on access permission and logs.
- Make sure necessary data is backed up.

KEEP STATIC ASSETS SAFE

Static assets are information that is kept in files, such as media uploaded by site administrators or configuration files copied from the code base. To keep them safe:

- Separate private files from public files.

- Require access keys to read or write to secret files.
- Automatically rotate credentials, log access records, and audit permissions and access.
- For public files, there will be publicly viewable assets, but they should never have public write permissions.

KEEP SECRETS SAFE

Secrets like passwords, database configuration, connection, or account information should never go into a code base. To keep these safe:

- Store secrets and passwords for production in your hosting environment with configuration tools that encrypt secrets. This prevents the hosting service and malicious actors from getting access to them.
- As part of change control tooling, developers should use tools that check for common patterns of secrets in code in their local environment so they don't become public.
- CI/CD tooling should not write out any secrets in its build output so anyone with access to the build logs won't be able to see that data.

KEEP CODE SAFE

In this context, "code" refers to the program instructions that make the application run that aren't covered in the above categories. These include:

- Basic configuration that should be the same across environments
- The instructions and rules for your application
- Which modules to install

- Themes and templates (css, html, etc.)
- Images like logos and favicons
- Tickets and nonsensitive discussions about code

To keep code safe:

- Keep private information and files out of the codebase and with your secure data and assets, as described above.
- Carefully review third-party packages.
- Use alerts for software updates and promptly apply security patches.
- Discuss potential security vulnerabilities in a private setting.

When an open source code base is used by a strong community of developers, everyone benefits from this active refinement as it continuously improves the code's quality and security

Sample verbal interview questions

During acquisition, verbal interviews are an opportunity to clarify the technical approach described in a contractor's proposal. (In the federal context, they are different from oral presentations under FAR Part 15 and do not permit a contractor to amend or change their proposal.)

Draw from the following questions to get details that will help you evaluate a vendor's experience with modern software development practices. Ask clarifying questions about a vendor's answers, either to reveal more information pertinent to a project or to explain an answer that seems odd.

ENGINEERING

- Talk about your process for determining which software and programming languages the development team would use to build the software, and explain the rationale for choosing those languages.
- What is your technology stack of choice for this project? Why? Which technology stacks does this particular team have the most experience with? What other stacks or technologies is the team experienced with?
- Describe your technical development and collaboration process. Please specify your approach to version control, testing (and test-driven development), accessibility, and continuous integration and continuous deployment.
- Discuss the technical decisions you've made in your proposal and any questions they raise for this project.

- How will you approach technical oversight? How would you track the standards described in the quality expectations (or QASP)?
- How would you identify deep problems within a code base? How would you address those problems to reduce technical debt? What types of re-factoring strategies would you consider?
- What do you anticipate as the largest risks in back-end development?
- How do you intend to address data security needs and requirements?
- Tell us about a time you came into the middle of a development effort. What challenges did you face? How did you overcome them? How do you envision integrating yourself within the existing development effort?
- Please describe your technical lead's experience with [name of programming language].
- Tell us about a system you built on top of some infrastructure- or platform-as-a-service.
- Tell us about an infrastructure problem that you helped solve, such as slow application performance, unexpected downtime, a security breach, etc. What was the problem and how did you solve it?

COLLABORATION

- How would you ensure good communication within the team and with government partners?
- How do you see the designers and developers interacting as you build the product?
- How would you like the [agency] to be involved as you design and build the product?

- What activities do you plan on engaging in to build and ensure strong collaboration among the team?
- How can you ensure that the various members of your team coordinate and collaborate across functions during performance and delivery?
- How do you typically communicate your findings and strategic recommendations to a client? How do you frame findings that might challenge your client's assumptions?
- Have you ever worked with a remote or distributed team before?
- If yes: What tools and/or mechanisms have you used to help promote open dialogue and foster communication on the team? How have you overcome communication challenges and barriers?
- If no: What challenges do you anticipate? What do you think you'll need to succeed?

STAFFING

- If awarded the contract, how would you quickly staff your design and/or development team?
- Tell us more about the different team roles you envision for this project.

RESEARCH AND DESIGN

- How do you plan to address the needs of the multiple user groups for this product?
- How have you incorporated changes into projects based on user research?
- What do you think are the most important features that users will interact with in the system?

- How would you design a usability test for an iteration of this product? What participants would you recruit? What tasks would you test? How would you analyze the results?
- How would you go about identifying a visual feel and content tone for the project?
- How would you bring the full team and stakeholders into the research and synthesis process?
- What is your experience with usability testing?
- Describe a time when user research findings disproved a team's assumption on one of your projects. What was the situation? What did you do? What did you learn?

PRODUCT AND STRATEGY

- What do you think are the most important risks for this project and how will you help [agency name] mitigate them?
- What do you need from the government product owner to make this project succeed?
- How will you help develop the product vision and prioritize features for development? How will you approach the process of prioritizing features to build throughout the project?
- Tell us about a project you led that was particularly challenging or complex. How did you approach it? How did it work out?
- How would you handle a request from the agency for a feature that you don't think is needed?

ITERATIVE DEVELOPMENT

- Describe the agile project management practices and tools you would use for estimating, planning, and managing risk, and for team collaboration and communicating status. Why use them in particular?
- How will you keep developers, designers, and researchers engaged in building features to fulfill a user story without having to extend a sprint or rely on a waterfall development process?
- How will the development team interact with the government product owner to ensure sprints are sized reasonably for the development team?
- If the team encounters a task that requires more work than originally anticipated and that can't be completed in the current sprint, how would you alert the government product owner?
- Tell me about your experience with agile software development or other iterative development styles. How does practicing agile affect the technical choices you make?

Verbal interviews are a critical quality control measure

Sample kick-off week agenda

DAY 1

Session 1: Introduction (60 minutes)

Purpose: The vendor team meets agency staff responsible for the project.

Attendees: Active project members only. Consider inviting a senior executive to kick off the meeting to signal the importance of the project and how invested the agency is in its success.

Session 2: Contract logistics (30 minutes)

Purpose: The contracting officer goes over contract administration items such as invoicing, delegation of duties, etc.

DAY 2

Session 1: What the agency knows or has learned so far (60 minutes)

Purpose: Recap discovery phase findings.

Session 2: QASP and deliverables (90 minutes or less)

Purpose: Review the deliverables and associated quality indicators or QASP elements. Create a schedule for when to revisit the quality indicators or QASP (and when to update it, if needed).

Session 3: Group alignment exercise (60 minutes)

Purpose: Surfaces risks, hopes, and fears from the development team, including the product owner, technical lead, and contracting officer representative. Some exercises include Assumptions and Risks, or Hopes and Fears.

DAY 3

Vendor asynchronous time (half to full day)

Purpose: Gives the vendor time to read over materials and prepare questions for future sessions.

DAY 4

Session 1: User research findings (60 minutes)

Purpose: For review of any user research already conducted, including methods and user groups who've already been contacted, and to begin discussing how findings from the user research should inform the future build.

Session 2: Technical overview of constraints and architecture (60 minutes)

Purpose: Describes the agency's technical landscape and any limitations that the development team may encounter. May also include an overview of the Authorization to Operate (ATO) process and other compliance requirements.

DAY 5

Session 1: Write team charter (60 minutes)

Purpose: Government and vendor team discuss how they want to work together and make decisions.

Session 2: Open working session (60 minutes)

Purpose: Allows the vendor to select the topic area and lead this session.

Sample Quality Assessment Surveillance Plan (QASP)

Deliverable	Performance Standard(s)	Acceptable Quality Level	Method of Assessment
Tested code	Code delivered under the order must have substantial test code coverage and a clean code base	Minimum of 90% test coverage of all code	Automated testing
Properly styled code	Meets acceptable quality level	0 linting errors and 0 warnings	Styling standards and linters
Accessibility	Web Content Accessibility Guidelines 2.2 – 'AA' standards	0 errors reported using an automated scanner, and 0 errors reported in manual testing	Automated and manual testing
Deployed code	Code must successfully build and deploy into a staging environment	Successful build with a single command	Live demonstration
Documented code	All dependencies are listed and the licenses are documented Major functionality in the software/source code is documented in plain language Individual methods are documented in-line using comments that permit the use of documentation generation tools such as JSDoc A system diagram is provided	Vendor provides documentation	Manual review

Deliverable	Performance Standard(s)	Acceptable Quality Level	Method of Assessment
Security	Open Web Application Security Project (OWASP) Application Security Verification Standard 4.0.3	Code submitted must be free of medium- and high-level static and dynamic security vulnerabilities	Evidence of automated testing per OWASP
User research	Usability testing and other user research methods are conducted at regular intervals throughout the development process (not just at the beginning or end)	Artifacts from usability testing and/or other research methods with end users are available at the end of every applicable sprint in accordance with the vendor's research plan	Demonstrated evidence of user research best practices

NOTES