CODE *for* AMERICA

Elements of Successful Systems Design for H.R. 1 Implementation

Four strategies for designing resilient technical systems that can adapt to rapid policy change

October 2025

Summary

H.R. 1—also known as the “One Big Beautiful Bill Act”—mandates large changes to Medicaid and Supplemental Nutrition Assistance Program (SNAP) administration, and state agencies are planning changes to their systems accordingly. By incorporating modern system design practices now, states can maximize the flexibility of their systems today and minimize the inevitable cost of making changes in the future. This resource offers state agencies insights on four systems design practices to consider when upgrading benefits systems to comply with H.R. 1. When working with vendors, state agencies can refer to these strategies to ensure their systems are responsive to their current needs and can adapt in the future.

- **Feature Flags**: Deploying features with an on/off “switch”
- **Modularity**: Building systems as a collection of modules that can be tested, upgraded, and swapped independently
- **Configurability**: Using configurations to modify a system’s behavior with minimal effort
- **Phased Approaches**: Simplifying deployments by developing and/or migrating systems in phases

Background

H.R. 1 makes sweeping changes to benefits administration, and state benefits agencies are modifying their systems accordingly. The complexity of this process is compounded by the incredibly short timeline and uncertainty about when states can expect guidance from the federal government. Furthermore, it’s possible that future legislation or policy updates will mean states will have to modify their systems further, or pause or roll back some changes. As state governments prepare to pay more for federal benefit programs, minimizing expenditures on system changes—which can be huge expenses—is a potential route to reducing overall spending.

Modern system design principles can help in these situations. By building systems that provide for quick and easy modification in the future, state agencies can ensure that when rules, guidance, or other requirements change, their systems can quickly adjust accordingly. If states design their systems to be flexible from the start, they can limit the cost of pivots and yield benefits far into the future. Even in areas where there currently is certainty, an upfront flexible design can ensure states are equipped to respond should changes arise in the future.

At Code for America, we utilize modern software design principles every day to produce flexible and adaptable systems. Below, we’ve outlined elements state agencies should ask for—and how to ask for them—when building under ambiguity. State agencies can use these four practices to prepare for H.R. 1 implementation now, as well as get ready for changes they may need to navigate in the future.

Feature Flags

A “feature flag” acts like an “on/off” switch that is linked to a particular feature. If a developer builds a flag into the system from the start, turning that feature on or off as needed thereafter is easy—no new deployment or code change required.

Feature flags are common in modern systems. A flag can be used to experiment with features still in development, or to “go live” with a feature at the exact moment it needs to be turned on—and vice versa. Importantly, feature flags allow quick modifications to a system’s behavior without the burden of a code change.

EXAMPLE

ABAWD Feature Flag

Some states and counties that have had waivers for Able Bodied Adults Without Dependents (ABAWD) work requirements are building or updating ABAWD rules in their SNAP systems right now. As waivers expire, or are granted in the future due to labor conditions, these rules may need to be “turned on” or “turned off” quickly. By implementing these rules with a feature flag, state staff can request ABAWD rules be engaged or disengaged as soon as policy changes, without requesting an entirely new code change and deployment.

Feature flags have a variety of use cases. State staff should inquire about flags whenever a policy change could result in a feature being turned off or on—even if the policy is currently solid. They should then work with the vendor to determine the trade-offs and behavior of the system. When a state agency requests a feature flag from a technology vendor, it’s important to ask the following questions:

- **What exactly does the system do** when this flag is “on”? What does it do when it is “off”?
- **How long will it take to switch the flag on and off**? This timeframe should be on the scale of hours, not days.
- **What will the user experience be like for caseworkers** when this feature is “turned off”? How is client data maintained? Is it preserved or deleted? How will it be treated by caseworkers when it is turned back on? Is it clear that the data is from the period before the feature flag was toggled?

Modularity

Modern systems function best as a collection of self-contained "modules" that all work together as one system. Among the many benefits of using modules is that they make the system particularly resilient to sudden changes. Rather than having to alter the whole system in the event of a new policy change, the vendor quickly and easily can conduct an *independent upgrade*, changing only the relevant parts.

EXAMPLE

Work Verification Module

Let's imagine that a vendor is building a system to verify Medicaid eligibility. Part of the system will do work verification—it will use income data to determine if someone is compliant with the new Medicaid community engagement requirements.

Now imagine that the state identifies a new income data source that has more comprehensive coverage. If the vendor built the work verification process as a module, *it can be upgraded in place*, without impacting the rest of the system. This leads to a quicker and less costly upgrade.

Remember: **modular systems do not require modular contracts.** A single vendor can build a system with multiple modules. Modularity is a *systems* approach, not necessarily a procurement approach. It's also important to remember that **the entire system does not need to be modular to support modules**—they can be built one at a time, as new features arrive. When appropriate, a single feature can be a module while the rest of the system is one unit.

When requesting that a system be built in a modular fashion, it's important to ask the following questions:

- **What parts of the system work best as modules?** Systems experts from the vendor team can provide valuable input here.
- **What new functionality can be packaged as a module?** What new requirements are expected to be implemented in the near future? Can upcoming changes to the system be leveraged to rebuild a piece as a module?

Configurability

Configurability describes a system's ability to operate based on "configurations" that are quick and easy to change. A configuration is a value or variable that can be adjusted without a new deployment or code change. Changing this value changes the way the system operates.

EXAMPLE

Medicaid Income Thresholds

As a result of H.R. 1, states may be making changes to the income thresholds that define eligibility. These thresholds are single values (like 200% or 138% of the federal poverty level).

If these values are configurable within the system, the state can simply request a change to the value itself rather than requesting a new deployment or code change in order to support the changing threshold.

Configurability is best utilized when a system's behavior is dependent on a set of simple—often numeric—values. Even complex features often follow this pattern. If a feature relies on a simple set of values that can change, state agencies should explore the possibility of making those variables configurable. When requesting that a feature be configurable, it's important to ask the following questions:

- **What values can (or will) change about this feature?** These will become the "configurable values" for this feature.
- **How long will it take to change these values once requested?** The timeframe here should be on the scale of hours.
- **What will the user experience be like for caseworkers** when this value is changed?

Depending on the implementation of configurable systems, configurations may be changeable without an engineer. In these cases, users of the system (including state staff and program administrators) can make changes themselves through a user interface designed for that purpose. There are pros and cons to this approach; these user interfaces are more costly to implement up front, but allow benefit administrators to change the system's behavior without involving any technical staff.

Phased Approaches

A set of changes can be deployed to the system as a bundle or "big bang", in which all the changes are deployed at the same time. This approach creates risk—if the deployment doesn't work, the entire set of changes must be rolled back, reworked, and reattempted.

An alternative approach is to deploy changes in phases. In each phase, a subset of the changes is deployed. If a rollback is needed, only the most recent phase needs to be reworked. If the work is interrupted, or needs to pivot, future phases can be altered to do so.

EXAMPLE

Medicaid Exemptions

In implementing H.R. 1, Medicaid agencies will need to capture exemptions for clients who are not subject to work requirements; each of these exemptions may have a data source. If all of these data sources are integrated and deployed simultaneously, they will all need to be rolled back if any single one of them has an issue.

Instead, each "phase" of the deployment could be a single data source that is verified, tested, and deployed before proceeding to the next phase. Then, if any single data source has a problem, it's possible to roll back that *individual* phase without affecting the others.

When requesting a set of features be built and deployed in a phased fashion, it's important to ask the following questions:

- **What parts of this feature set can be broken into phases?** Phases should be small enough that they are easy to deploy and roll back, but large enough to capture an entire logic change.
- **How does the system work between phases?** If a phase is completed, it should not block the system from functioning normally without the subsequent phases.
- **What happens if each phase fails?** What is the rollback plan? This should be a very easy and short process.

Getting Assistance

A discussion of these strategies can be very technical, and states should be prepared to get into the details with their vendor when making a request. The civic technology community can be partners in this area—technical experts can assist even for single, short conversations about the feasibility of an approach and the expected implementation burden. For longer discussions or projects, Code for America and our partner organizations stand ready to work alongside state staff and vendors to ensure the best outcomes. It's our goal to empower state agencies with the knowledge they need to make informed decisions about these strategies.

Moving Forward

Code for America supports implementing these principles in all our state partnerships. We recognize that the options described here add complexity to a build, and systems are less expensive without them. But these up-front costs in time and complexity are significantly lower than the cost of a code change and deployment to reverse the original change. When change is a certainty, including these design options can yield major long-term benefits while minimizing short-term costs. These flexibilities are important not just in areas of uncertainty, but in any aspect of a system that may require changes in the future.

As states begin having conversations with their vendor and implementation partners about H.R. 1 response, Code for America is ready to work side-by-side with you to ensure that systems are designed and built in the most modern, flexible, and cost-efficient way possible. Doing so will not only yield better outcomes for clients and beneficiaries, but also for states as they continue to implement changing requirements.

Are you looking for direct support implementing work requirement policies?

Code for America works alongside government agencies to design technologies and systems that enable public benefit delivery. **Reach out to explore how we can work together to respond to H.R. 1 mandates.**