18F

Administrative Office of the U.S. Courts

CM/ECF

Path Analysis

[March 31, 2021] Public Version

Note: several details about CM/ECF system architecture and security posture have been removed from this report to protect system security.

Team

Alex Bielen, alex.bielen@gsa.gov
Donny Kwan, donny.kwan@gsa.gov
Dave Luetger, david.luetger@gsa.gov
Norah Maki, norah.maki@gsa.gov
Vicki McFadden, victoria.mcfadden@gsa.gov
Miatta Myers, miatta.myers@gsa.gov

Table of Contents

Executive summary

18F, a team within the General Services Administration (GSA), carried out an 11-week Path Analysis on the federal judiciary's Case Management and Electronic Case Files (CM/ECF) system. Our research focused on user needs, business agility, organization and processes, and the Administrative Office of the U.S. Courts' (AO) culture and legal mandates.

Our synthesis of the information we gathered from 100+ participants across 77+ sessions shows the pain points (i.e., specific problems users are experiencing) of the current state of CM/ECF:

- Unsatisfactory user experiences for both internal CM/ECF and external Public Access to Court Electronic Records (PACER) users.
- Court business needs are not being met, resulting in 200+ software versions with many local modifications.
- Although nearly all courts have upgraded to the Next Generation of CM/ECF (NextGen) or signed up to do so, more than 50 courts have not yet gone live on NextGen.
- The foundational technology is outdated and some components are becoming obsolete; it is not sustainable.
- Decentralization and complexity are causing system instability, high maintenance costs and security risks.
- Current contracts make it difficult to hold contractors accountable to quality standards.

18F recommends the following path forward for the judiciary:

- Build a new, open source system with modern technology and architecture. Some components of the new system could be built in-house, but for others it may be more cost-effective and efficient to use contractors who have relevant domain expertise.
- Consider cloud computing as an option.
- Simplify AO operations and encourage courts to innovate through Application Program Interfaces (APIs).
- Design strong data standards and investigate a data catalog and governance framework.

- Align database schemas across all court types and adopt a new database type that can handle multimedia and other large files. PACER should share the same database.
- Adopt a DevSecOps culture to support the integration of security and operations concerns at every phase of the software development lifecycle; from the architectural design phase through, testing, deployment, delivery, and production.
- Stand up a new cross-departmental and cross-functional team to develop and maintain the new national system for courts. This team will need some domain-knowledge support but should be empowered to build the new system independently.
- Incorporate a Quality Assurance and Surveillance Plan (QASP) in contracts to hold contractors accountable for quality deliverables and continue using the time and materials (T&M) contract type.
- Meet with CM/ECF stakeholders to leverage their domain expertise during planning and building phases.

1. Introduction

The AO engaged an 18F team to carry out an 11-week Path Analysis of CM/ECF. The purpose of this project was to conduct user-centric research, identify user pain points, and recommend improvements and a roadmap for the judiciary to move forward.

18F is a government digital consultancy housed within the GSA's Federal Acquisition Service. It aims to help government agencies deliver exceptional digital experiences by practicing user-centered design, releasing features often, and deploying products in the open.

At 18F, each Path Analysis is customized to the needs of an agency, with the goal of moving from identifying a problem to working on a solution. We developed an action-oriented analysis of routes to pursue, places to narrow the project's scope, and the best ways to deliver value to CM/ECF users.

Our holistic approach looked at CM/ECF from multiple angles:

- User and business needs,
- Organization, culture, and communication patterns,
- Technology and development processes,
- The AO's legal mandates, including the implications if the Open Courts Act[1] passes, and
- Governance, Risk and Compliance (GRC) requirements.

We also reviewed the Judicial Conference of the United States' mission statement, core values, and strategic plan, to ensure our recommendations align.

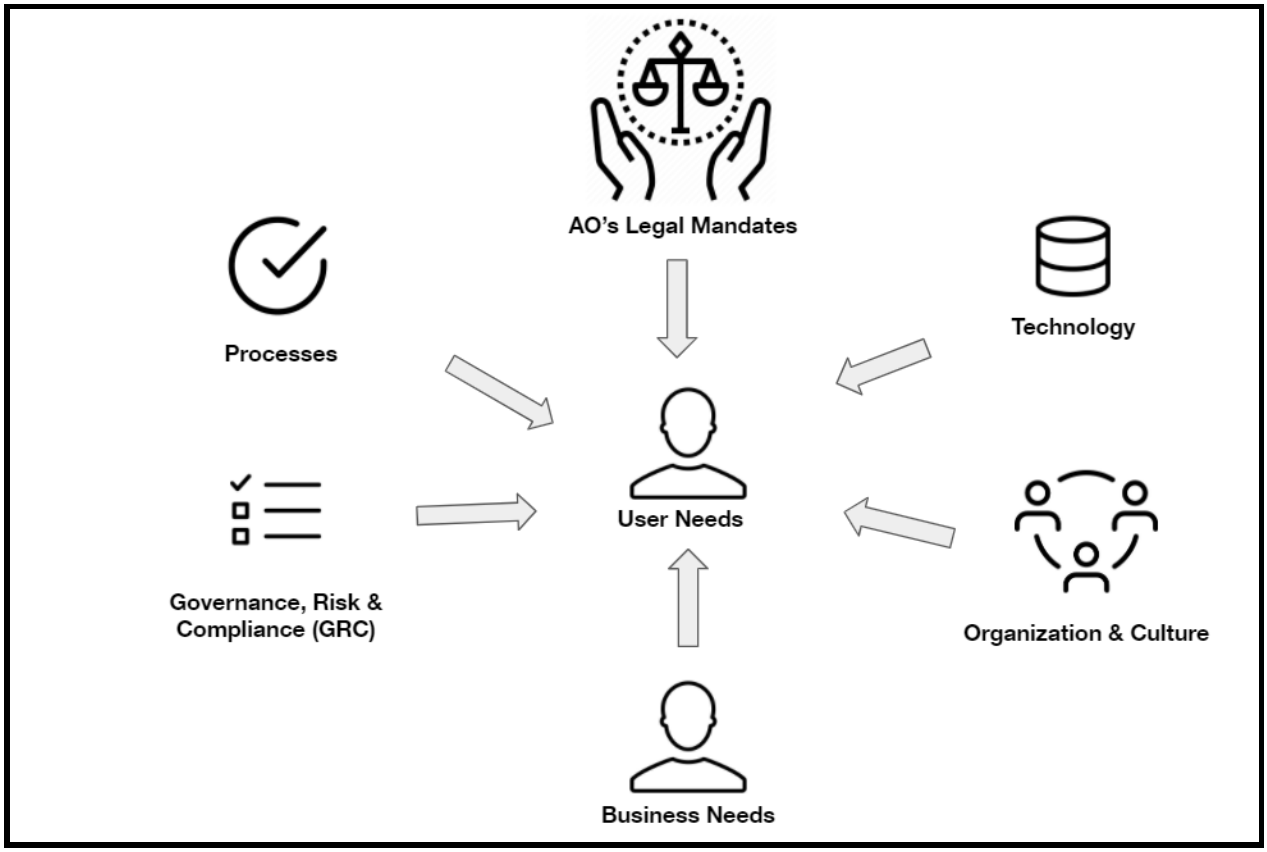
The judiciary's business and mission drivers are:

- To be the best place to work, that attracts talents and diverse employees,
- To improve business agility,[2]
- To provide support to all federal courts to ensure that all people have easy access to justice.

[1] Open Courts Act of 2020, https://www.congress.gov/bill/116th-congress/house-bill/8235/text
[2] Availability, Reliability, Performance, Integrity, Intuitive, Confidentiality & Security.

CM/ECF is an electronic version of a paper filing cabinet for the federal courts. It was designed and originally built in the 1990s, and approximately 14 years ago, the judiciary undertook a major system upgrade, a.k.a. NextGen, which is still in process. CM/ECF was the best technology and platform available at the time and has indeed served the federal courts and public well. However, broader technology trends have changed significantly in the last 20+ years and the judiciary's technology and process needs are very different today. Courts need to be able to process and administer cases, populate initial filing data across all aspects of a case, docket and perform automatic quality control of filed data, create analytic reports, analyze big data, and present information in an effective way to serve and benefit court staff, litigants, and the public. In short, "*IT has become a means by which Judges and Judiciary staff do their jobs."*[3]

The judiciary needs to improve CM/ECF to *"meet the needs of jurors, court users and the public in a timely and effective manner."*[4]



[3] Long Range Plan for Information Technology in the Federal Judiciary, fiscal year 2021 update.
[4] Strategic Plan for the Federal Judiciary (September 2020).

2. Our approach

At 18F, agile product development is realized in the combined practices of iterative software development, product management, user-centered design, and DevSecOps.

We start with a product vision and strategy, informed by users and the overall mission of our partner agencies. We do this so that the work always stays connected to an overarching goal that everyone understands and is excited about.

We also work to ensure that the infrastructure and processes are there to enable continuous delivery of software to end users (DevSecOps), and that a clear agile delivery process is established. Teams are free to tailor their agile process to suit their own situations.

We conduct discovery research before we build anything. Depending on the complexity of the problem space, this can take up to 2 to 3 months. As opposed to “requirements gathering,” this process involves speaking with users and showing them prototypes to test out multiple concepts quickly before investing a lot of time or money in a build.

When we build, we aim to release early and often to end-users using agile development methods. Ultimately, the government’s investment should be measured in working software, not phases, documents or milestones. Only working systems are of value to real users. “Waterfall” is a software development methodology that focuses on completing steps sequentially in order to deliver a software product -- this means that requirements gathering and design are completed before development starts, and development is completed before testing starts. In a “Waterfall” model, end-users only receive value at the end of the project, after all the costs have been incurred. Waterfall is a risky way of building software. Agile development allows the government to provide value and measure success at more regular intervals and, if necessary, make course corrections. The ability to measure and adapt reduces the overall risk to the project.

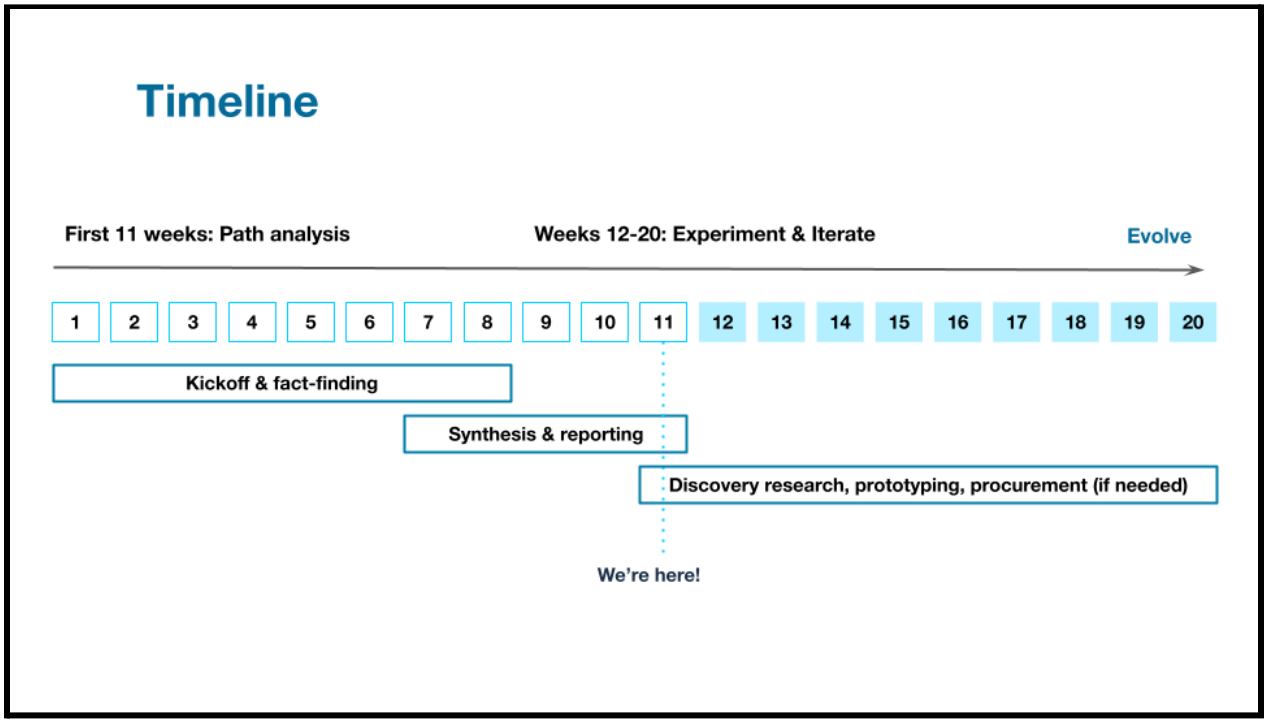
Having well-researched hypotheses beforehand allows us to be deliberate about what we build and why. Research continues throughout the agile process so that we can test our hypotheses and pivot when needed. Since our work is centered around user research and user needs, it is important to regularly show software to end-users and get their feedback.

A common pitfall is expecting agile to be a silver bullet to all that ails software development, and to expect agile to eliminate all project risks. In reality, agile does not eliminate risk completely; it provides techniques to manage risk more effectively and accepts that current unknowns will lead to change down the road. Agile treats change as an integral part of the process, as opposed to exceptions that need to be resolved via change control mechanisms. This enables agile teams to manage risk by allowing change to drive course corrections. Effective agile adoption enables an organization to be nimble and respond effectively to the inevitable change that arises during software development.

3. Our research methods

Structure of the engagement

Asking the right questions, understanding and solving the right problems.



Due to the size and scope of the Path Analysis and subsequent Experiment and Iterate (E&I) phase of work, 18F took a broad approach by researching and interviewing CM/ECF users across court types, user types, and AO employees. During the subsequent phase of work, 18F plans to work with the AO and the court community to build a common use case that can quickly improve the CM/ECF user experience.

User-centered design and research

Collecting information

18F employed several different research methods throughout this project. Our primary research method was semi-structured interviews. Semi-structured interviews are a qualitative research method where interviewers begin with a structured set of targeted

questions, but are allowed to ask follow-up questions during the course of the interview and take the conversation where it naturally flows. The benefits of semi-structured interviews are that they help to generate richer context about user experience and get more valuable insights over methods like surveys or focus groups.

Our interviews were conducted primarily 2-on-1, with a facilitator and note-taker from 18F, and one participant. Occasionally, more 18F observers were present. Also, on occasion, more than one participant would be invited, and questions would be asked generally or be targeted to individuals.

Our research goals were to identify pain points, develop further research questions, and uncover opportunity areas for future prototyping. Roughly half of our interviews focused on the experience of "making the thing," meaning the internal goings-on at the AO and courts related to building, maintaining, and supporting CM/ECF. The other half focused on the experience of "using the thing," meaning the judicial process and its touchpoints with CM/ECF. These interviews occasionally included users walking us through parts of the system and sharing their screens.

Participants

We interviewed 100+ participants across 77+ sessions. Roughly half of our participants represented "technical" staff including roles such as IT managers, software developers, and user support. The other half represented "court staff" including roles such as judges, clerks (and staff), and chamber staff. The participants were from across the country, representing all 3 court types (appellate, district, and bankruptcy), from small to large courts. Appendix A shows the statistics of the interview groups.

We also spoke with many court IT groups that have developed applications for the different court types to use. These groups include, but are not limited to, the District of Utah's Chamber Automation Program (CHAP), the 2nd and 9th Circuit's Appellate Case Management System (ACMS), and the Chambers Electronic Organizer (CEO).

4. The current state

In this section, we summarize our user experience findings so far and identify pain points and issues stemming from the current technology, organization and culture, and acquisitions. The following section presents recommendations that are targeted to address these current-state challenges.

User experience findings so far

We heard of many negative user experiences. These include, multiple examples of crashes and system errors, a system that cannot currently accept large media files, reports that affect system performance, and slow application response times.

However, not every user experience speedbump is directly related to technical aspects of the system. On this project, we also sought to further understand examples of ways users' needs may be better met.

We took a very broad approach to this phase of the project. Speaking to 100+ participants means we have saved most of our deeper synthesis and analyses for the next phase of work. But, scratching the surface has revealed a few opportunities we are excited to dig into deeper during the next phase, described below as "how might we" statements.

How might we improve the stability and performance, or its perception, broadly across CM/ECF?

Throughout our research, we observed several instances of application crashes and sluggish response times, and gathered several participants' reports about similar issues. While we have not yet discovered mass consistency among these incidents, these sorts of experiences add up over time for individual users, diminishing their experience and creating distrust and dissatisfaction with the system.

In the next phase, we would want to:

- Inventory these errors, and potentially read more user error reports, to discover patterns or if there are any consistent errors across instances.

- Learn what the root cause is for some of these errors – for example, is it the decentralized nature of the system?
- Evaluate the time or effort cost caused by these errors.

How might we improve the efficiency, accessibility, and usability of CM/ECF?

In addition to technical errors, we also observed several places throughout the CM/ECF interface that are difficult to use, and some tasks that require repetitive clicking or that may be inaccessible to users with disabilities. Again, more rigorous usability and accessibility analyses would be needed to say anything conclusive, but improvements in these areas could represent a number of small wins.

In the next phase, we would want to:

- Thoroughly explore one or two workflows within the system and do a proper usability test and/or heuristic analysis.
- Perform an accessibility evaluation on one or two workflows within the system.
- Evaluate the costs associated with usability errors.

How might we make CM/ECF's docketing tools more flexible, efficient, and easy to use?

Docketing workflow is likely the most important or core aspect of any generation of CM/ECF. During our research, however, we observed that many court users have overwhelming caseloads and that each individual case has numerous docket entries. We see a lot of room for improvement in the processes for creating and maintaining docket entries in CM/ECF, and think improving this process to make it more flexible and easier to use could help court users manage their caseloads more efficiently.

In the next phase, we would want to:

- Map out the full docketing workflow and understand pain points along the way.
- Understand more specifics about how users' experience with the CM/ECF docketing tool is affecting their ability to efficiently manage their caseloads.

How might we improve and better support the judge workflow, including scheduling, annotation, and managing caseloads?

From what we have observed so far, we have learned that the entire judge workflow, from scheduling to annotation and managing caseloads, is an area for improvement. Judges want to be able to see where they need to be next, and understand the context of a case quickly, all while potentially managing multiple cases, hearings, and appointments each day. A “white whale” for the judiciary over the last several years has been CM/ECF calendaring.

In the next phase, we would want to:

- Map out the experience of a judge and their staff and understand their pain points.
- Conduct a usability audit of the current calendaring tools available in CM/ECF.
- Observe judges and court staff using other calendaring tools and workarounds to better understand what works for them and why.

How might we improve the ways in which users are able to access and view summary information?

We heard a lot about reports in our interviews: why they are needed, what information users are looking for, and how frustrating they are to make and run. What might a better report environment look like? Are there ways to get data without “running a report?”

In the next phase, we would want to:

- Understand the “why” behind popular reports. Who is creating them? Who is viewing them? What questions are they helping to answer?
- Learn about the technical limitations of the system and the ways in which users might be able to view real-time information in the system.

How might we improve the look and feel of the CM/ECF interface, such that a sense of trust and professionalism can be maintained?

We heard a lot about the current CM/ECF interfacing looking “out of date” and requests for a more “modern” interface. While it is important to be specific about what

these phrases really mean, and focus on usability first, there is a very real effect that can occur with interfaces perceived to be out-of-date. Trust in the system's functionality and professionalism can be eroded over time, and a user's perception of inconveniences can be much worse when the look and feel of the interface is dated.

In the next phase, we would want to:

- Dig more deeply into what participants mean when they use phrases like "out of date" or "modern."
- Learn more about interfaces users perceive to be more modern and why those work well for them.
- Work with CM/ECF stakeholders and users to determine what the "look" of a better CM/ECF could be and test it with users.

How might we best support the experience of the public, from users seeking information to users filing on their own behalf?

We did not touch much on the details of PACER or practitioners' experiences with CM/ECF during this first phase of the project, nor have we spoken yet with any participants from the general public.

Having learned a bit about the workflow of pro se filers, we are very curious what opportunities are out there when it comes to better serving users who are representing themselves.

In the next phase of work, we would like to better understand the experiences of users outside of the courts.

In the next phase, we would want to:

- Speak with members of the public regarding their experience gathering case information or filing on their own behalf.
- Map out the various touch points between a public user and CM/ECF and/or PACER.
- Dig a bit deeper into the gray area of "providing legal advice" versus providing usable wayfinding (moving from point A to point B) through a web interface.

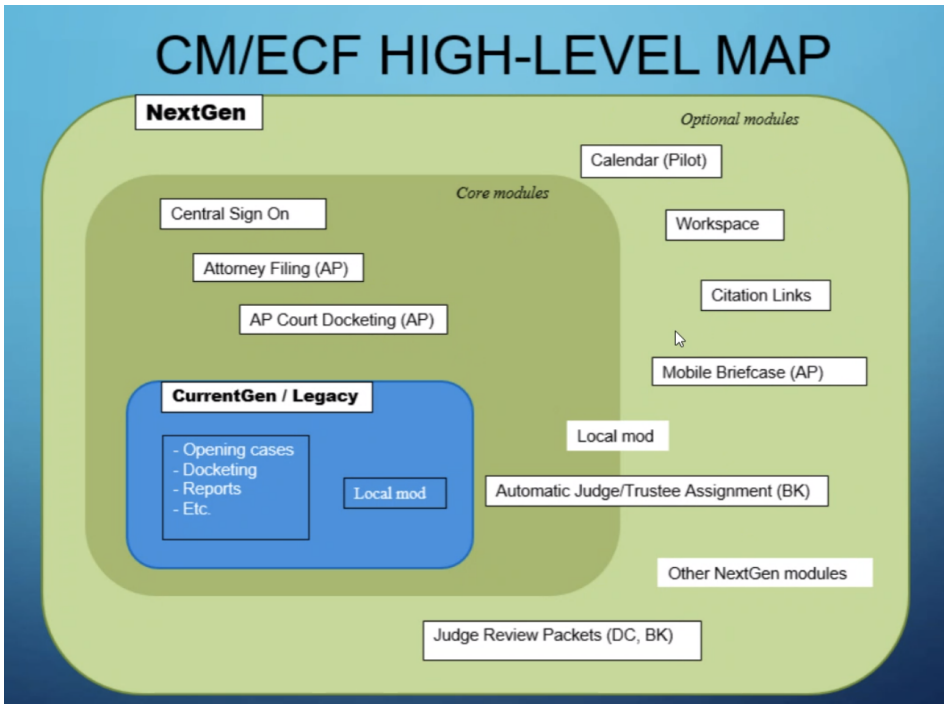
Current system strategy

Dated technology, decentralized deployments, and heavy customization are leading to long feature lead times, unhappy users, unstable systems, and security and reliability risks.

Application

The original CM/ECF system, a.k.a CurrentGen, was built in the 1990s. It provided an electronic case filing system to the Courts and was written in Hypertext Markup Language[5] (HTML) and Perl[6] programming language. The application was designed with Apache Static Web Service and Informix Database. The application servers were on-premise. The operating systems were Redhat Enterprise Linux (RHEL).

About 14 years ago, the judiciary initiated a major system upgrade, NextGen, by adding new modules and adopting Java as the new programming language and Apache Tomcat service. In the meantime, the judiciary also moved away from on-premise servers to a coastally-distributed dual data center approach, with each data center serving as a primary and a backup according to court proximity. All the services are also running as virtual instances (VM) on the VMWare platform. Courts are accessing the web application and database service remotely via the internal network.



[5]HTML is the standard markup language for documents designed to be displayed in a web browser.
[6] Perl is a general-purpose Unix scripting language to make report processing easier.

Our research found that there are mixed feelings about the success of NextGen. As of March 2021, there are still 52 courts that have not yet gone live on NextGen. Users told us that some of the new NextGen modules have usage and performance problems.

Many courts have developed "local mods" (locally developed applications and patches to CM/ECF) to meet their particular needs, which has created problems ranging from high cybersecurity risks to high operational costs. Courts oftentimes change code to meet court-specific requirements or to work around bugs in CM/ECF, which could potentially introduce vulnerabilities into the system.

We also found there are a few "nationally funded, but locally supported" programs like CHAP and CEO which are being used by many bankruptcy and district courts, respectively. The AO is also funding ACMS, which is being piloted by the 2nd and 9th Circuit Courts of Appeal.

There are 200+ versions of CM/ECF, which means that doing any upgrade or patch is a substantial burden for the AO and the courts. The Case Management Systems Office's (CMSO) CM/ECF division has been running a "keep-the-lights-on" operation for the last two years. They are doing security and compliance patching and limited critical bug fixing, but are not releasing new features to end users, except limited new functionality that is required by law or directed by the Judicial Conference.

Foundational technology has fallen out of favor; better alternatives exist.

The foundational technology of CM/ECF is difficult to maintain.

The CurrentGen and NextGen applications have significant components written in the Perl programming language. There are several characteristics of the Perl programming language that make it difficult to maintain. It is a dynamically-typed scripting language, which means certain types of errors that are easy to detect in other languages are very difficult to detect safely in Perl. While dynamically-typed scripting languages like Perl can be adequate for small projects, they present significant costs for large enterprise applications, including application downtime, bugs, and extended testing time.

The unpopularity of Perl will affect the judiciary. The TIOBE Language Index[7] lists Perl at #17 in popularity on its March 2021 index.[8] Perl's popularity has steadily declined

[7] The TIOBE Index is an indicator of the popularity of programming languages.
[8] TIOBE Index, https://www.tiobe.com/tiobe-index/perl/

from #3 over the last 20 years. Additionally, Perl has been listed in the top 3 "most dreaded programming languages" for software developers.[9] This means that the judiciary will have difficulty finding Perl developers in the future and that the ecosystem of tools and libraries written for Perl will not receive the necessary investment needed to integrate with the ever-changing technology landscape.

One illustrative example of the negative impact of Perl on CM/ECF is the Outserve component. Outserve is a central component of CurrentGen and NextGen CM/ECF. Outserve is an example of "legacy code": it is written in a programming language that has fallen out of favor (Perl), it lacks a substantial set of automated tests, and it is currently poorly understood by the judiciary's IT staff and contractors, which limits improvements. Although it has since been "encased"[10] in Java in NextGen, Outserve was written in Perl for CurrentGen.

Outserve lacks adequate automated testing. Although there has been an effort in the last 18 months to improve the automated test coverage of the Outserve component, retroactively adding testing to a codebase, without refactoring the code under test, does not have the same benefits as integrating automated testing into the design and implementation of the codebase from the start. Automated tests increase software quality in both qualitative and quantitative ways including regression detection, method signature simplification, higher cohesion, and looser coupling between dependencies; and serves as a form of documentation for the codebase. Automated testing also makes it easier to change software development contractors as needed; different software developers can read the tests and understand the objectives of the underlying code.

Lastly, Outserve is an extremely complex piece of code. It is made up of more than 1,100 different Perl scripts with a centralized dispatcher that calls other libraries. Judiciary IT staff described these scripts as "very complex" and "extremely high-risk to replace." For this reason, the judiciary limits Outserve improvements to "tweaks and patches … because no one really knows [the codebase]."

Another component of CM/ECF that has fallen out of favor is the Informix database.[11] The Informix database was first released in 1980 and it is not designed for today's

[9] StackOverflow Trends, https://insights.stackoverflow.com/trends?tags=perl
[10] https://18f.gsa.gov/2014/09/08/the-encasement-strategy-on-legacy-systems-and-the/
[11] Informix Software was acquired by IBM in 2001. HCL and IBM have since partnered to co-develop, market, and support the product.

information technology needs. The Structured Query Language[12] (SQL) schema is likely "over-normalized" which means that many "joins" are needed to perform useful queries on the CM/ECF database. We heard from numerous court IT experts that useful queries require 15-20 joins. Requiring numerous joins causes application and database performance issues and deadlocks. End-to-end, this leads to a "sluggish" user experience in CM/ECF. There are modern database solutions available that can better handle the judiciary's needs. These solutions can be fit to transactional or analytical data use cases, and schema can be designed around application use cases.

Development, deployment, and operations

Configurable and decentralized by design.

Configurability and decentralization have been important design considerations in CM/ECF in order to support judicial independence and individual court autonomy. The current, highly configurable implementation, however, has introduced a host of challenges including security issues, unpredictable systems, and high cost of change.

Configurability leads to unpredictable systems and high maintenance costs.

Software complexity drives costs and makes systems difficult to maintain. CM/ECF has thousands of customization options, which makes the number of possible customized states well over a manageable level.

This complexity leads to hard-to-predict behaviors. Configurability can lead to feature incompatibility and hence unexpected behavior due to interrelationships between configuration settings. As a simplified example, you could unintentionally set the background and text color to a combination that makes the text unreadable or inaccessible.[13] If something stops working, it is often difficult to diagnose the problem because there are simply too many options to explore. CM/ECF's complexity also makes it difficult to adequately test the software. There are too many paths to test, which means that it is hard to predict how new features and updates will impact existing CM/ECF installations when new code is released to courts.

[12] SQL is used to communicate with a database.
[13] https://designsystem.digital.gov/design-tokens/color/overview/

Courts are hesitant to install new versions of CM/ECF.

We heard several times in our interviews that new features often break existing functionality or court-developed modifications. This often requires workarounds or re-work by the courts.

In addition to the risk that new releases present to local modifications, courts are also hesitant and/or unwilling to go first in installing new software releases because they expect there will be regressions and incompatibilities with their local modifications and configurations. The hesitancy that courts experience with installing new versions perpetuates the problem of various courts running different versions of the application. This in turn adds to the operational burden on CMSO's CM/ECF division, as they support the many different versions that are running in production.

Decentralization and complexity lead to significant costs to courts.

The costs of operating and maintaining CM/ECF are high for the judiciary. Courts currently need to dedicate resources to maintain, test, install, and debug the software.

It takes years of experience and significant training for court CM/ECF administrators to become comfortable with their specific version of CM/ECF. This is due primarily to each court's "Dictionary" setup (the customization mechanism), Docket Processing Functions, and local modifications. Even with extensive training and experience, the system is error prone and it is difficult to keep track of the numerous local modifications and system dependencies. We heard from many CM/ECF administrators that their courts would be in a bad position if they left their positions because they are often the sole person who understands all the complexity of their court's version of CM/ECF.

This complexity also impacts the CM/ECF end-user experience. New upgrades and installations often create unintended consequences for their users. The system generates errors that are hard to diagnose and fix. The system is slow and reports must be run over night. Users have to wait a long time for feature enhancements or bug fixes due to system complexity. It takes a long time to train new employees on how to use the system. All of these challenges impact the courts' business agility.

Complexity and decentralization are introducing security risks.

There are security risks associated with operating complex and decentralized software. Because of the decentralized architecture of CM/ECF, the AO is reliant on individual courts to manage these patches. Moreover, due to the autonomous and decentralized design, courts have the ability to modify the CM/ECF codebase.

Infrastructure

The current infrastructure is shown in Appendix B. CM/ECF data centers are linked with the AO's virtual private networks and each center acts as a backup to the other. The applications and database servers are running as virtual machines on the VMware platform. We heard from many courts that there is no automated warm/standby failover during a server failure. When a server fails, court administrators must bring up the standby server manually. As a consequence of this, failovers could lead to extended downtime if there was a significant outage.

> *"We had a [partial outage]. It affected almost 100 court systems. It happened on the weekend, thankfully. If it had happened during the week it would have been front page news."*

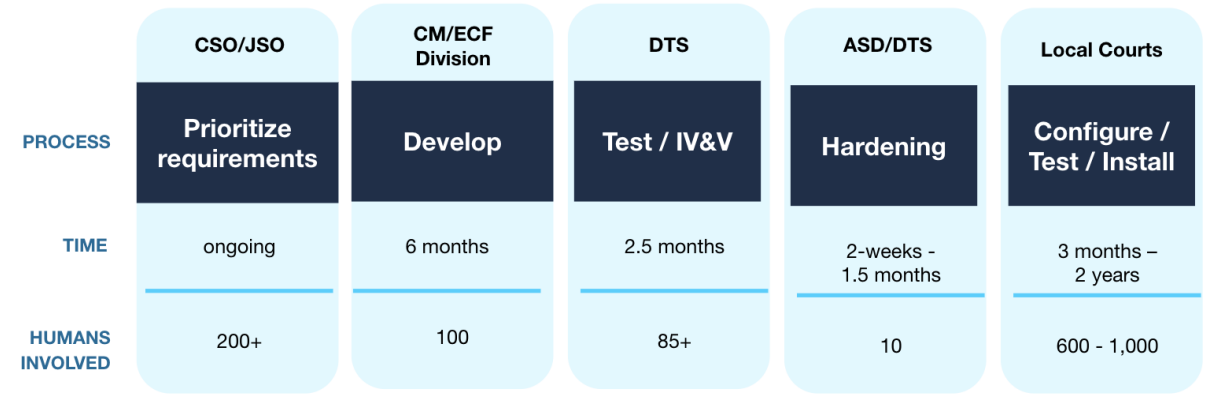
Organizational context

Unclear operations responsibilities are leading to significant risks to CM/ECF reliability.

Responsibilities for CurrentGen and NextGen operations are shared between CMSO's CM/ECF division (which is located within the Department of Program Services (DPS)), various offices within the Department of Technology Services (DTS), and courts. The CM/ECF division is responsible for development, bug fixes, and second- and third-tier support. DTS handles testing, hosting, networking, and first-tier support. Courts are currently operating, patching, and maintaining CM/ECF on servers provided by DTS's Cloud Hosting and Networks Office (CHNO).

The process of delivering a new feature is time-consuming. From identifying a need to delivering working software that courts can use, can sometimes take years and

requires work by hundreds of people.



When a new version of CM/ECF is released, each court decides whether and when to install it, and court IT staff test and install it. Once courts decide to install a new version, there are often prolonged periods of local testing due to incompatibility between the new national version and existing local modifications. CM/ECF upgrades require temporary local outages, which courts plan on weekends to avoid interruptions to court operations.

In short, fragmented operations responsibilities are leading to unclear responsibility boundaries, disagreements over operational priorities, and some judiciary IT staff feeling unempowered to advocate for improvements.

Software development is driven by requirements-setting bodies and product owners are not empowered.

At the AO, software development is driven by requirements gathered from courts, technical expert panels, and DTS. The AO's backlog contains more than 15,000 requirements and bug fixes. Requirements gathering causes issues in government modern software development and is an artifact of waterfall development, which is discussed in more detail above. This differs from user-centered design, which involves observing users interacting with the system, identifying common challenges, and iteratively building a product with frequent user feedback in order to address the observed needs.

There are product owners at the AO within DPS's Court Services Office (CSO) and Judicial Services Office (JSO) but they are not empowered to act on their understanding of user needs and have limited input in determining development

priorities; rather, CMSO's CM/ECF division sets the priorities for development based upon available resources, security priorities, and "technical debt" that is required to be addressed.

Lack of data standards is impacting external and internal user experience.

There are limited data standards for CM/ECF. This makes it difficult for attorneys to work in multiple courts as they have to deal with multiple, different CM/ECF versions. It is also difficult for PACER users to find information. Transferring case information between courts is very challenging because of different database schemas between courts.

High cybersecurity risks.

There is the potential for many cybersecurity vulnerabilities resulting from the way CM/ECF software is built, deployed, and maintained. Security and compliance are monumental tasks for courts and the AO's visibility into courts' security posture is limited due to the decentralized nature of the application.

Acquisition

AO contracts already include some agile acquisition best practices but there are still issues with holding contractors accountable.

We examined the current two task orders for CM/ECF development support services and the structure of the development teams that are managed by CMSO. In short, the AO is following many agile acquisition best practices. Compared to what we have experienced at other federal agencies, the AO's contracts do not require big changes to align with modern agile acquisition best practices, but there is nevertheless room to help the AO hold its contractors accountable.

While the AO is not part of the executive branch and thus does not have to follow the rules and procedures of the Federal Acquisition Regulation (FAR), the AO still follows the FAR to the maximum extent possible. For example, by leveraging the efficiencies of the Federal Supply Schedules (i.e., GSA schedules) as described in FAR Subpart 8.4, the AO has streamlined its acquisition process and is able to award contracts quickly and with as few steps as possible. There are many agencies that are covered by the FAR that do not take advantage of the benefits of using the Federal Supply Schedules, so the AO is ahead of the curve in this regard.

Another good practice is that the AO's contracts use minimal contract line item numbers (CLINs), which reduces the administrative burden on the contracting officer and acquisition team, especially when there are staffing changes. The use of this CLIN structure minimizes the need to issue modifications every time there is a net-zero change to the contract (e.g., adding one labor category while removing another). By using one labor CLIN per performance period, the contracting officer issues fewer modifications over the life of the contract.

Another example is the AO's use of the time and materials (T&M) contract type with a not-to-exceed ceiling. Use of T&M contracts has the following benefits:

- Flexibility in making in-scope changes without the need for contract negotiations or a contract modification (something that is required for firm-fixed-price (FFP) contracts).

- The AO only pays for actual hours worked. Whether a contractor employee goes on leave or the government shuts down, if the contractor is not working, then the AO does not pay.

- In coordination with the product owner, contracting officer's representative, and/or contracting officer, T&M contracts allow the contractor to add more people to the development team as priorities shift from sprint to sprint.

The combined use of minimal CLINs and the T&M contract type with a ceiling maximizes the flexibility of the AO's contracts and is something we rarely see already in place at our partner agencies.

The AO's software development task orders are for the services of over 100 contractor employees from two contractors. The contractor employees are assigned to various development teams across the AO that, in turn, support multiple nationally deployed products, including CM/ECF, PACTS, and eVoucher. CM/ECF development teams are a mix of government employees and contractor employees from each of the two contractors. The time and materials (T&M) task orders do not include working code (i.e., code that actually works) as a deliverable of the contract nor do they include a quality assurance surveillance plan (QASP) which would set a minimum quality standard for the delivered code.

As a result, it is difficult for the AO to hold either contractor accountable for issues with CM/ECF code because: (1) without including deliverables on the T&M contracts, the AO is paying the contractor just for their time, and (2) the entire development team (not a single individual) is responsible for the code, so having individuals from the both contractors (as well as AO employees) on the same team makes it difficult to hold either contractor responsible for issues with CM/ECF code.

5. The future state and how to get there

> *"Harness the potential of technology to identify and meet the needs of judiciary users and the public for information, service, and access to the courts."*
>
> *- Strategic Plan for the Federal Judiciary, Strategy 5.1*

In this section, we start with a succinct problem statement and future product vision. Building on that, we detail a few options for a high-level approach to the product vision. Lastly, we go into detail about the recommended future state for **technology, organizational context,** and **acquisition,** along with the first steps to take to realize the future.

Problem statement

CM/ECF is not sustainable. System complexity is leading to long development and installation timelines, long training periods for new staff, a negatively impacted experience for users, high costs, and security risks. The foundational technology is dated and will be hard to maintain into the future.

Product vision

A forum, designed with and for users, that supports the administration of justice by facilitating efficient and effective case management and filing.

High-level approach

The judiciary should build a new system.

Taking into consideration what we have learned about CM/ECF user needs, the results of our SWOT[14] analyses, CM/ECF's integration with PACER, and the potential impact of the Open Courts Act or other similar legislation, we recommend that the judiciary build a new, open source case management and filing system. In this system, the judiciary should offer the **most valuable and well-designed** functionality. The system should be user-centered, and developed based on quick, iterative feedback loops.

[14] Appendix D: SWOT stands for Strengths, Weaknesses, Opportunities, and Threats. A SWOT analysis is a technique for assessing these four aspects of your business.

By using updated technologies, modern architecture, improved cybersecurity, an intuitive user interface (UI), and other modern best practices, the new system would ultimately improve the end-user experience. It would be more reliable and easier to maintain, both for the AO and courts, and it would be less expensive to operate in the future. Courts should retain some capability to customize the UI and business logic layers for local needs. However, courts would not be allowed to access source code or the database directly. Instead, all court and third-party systems would be able to access the new system via an API.

Building a new system will be challenging, but the benefits will be tremendous – better user experience, improvements delivered more frequently, less costly, better security posture, and easier for the judiciary to maintain into the future.

Build a strong product thinking capability to build the most valuable software for your users.

The AO should develop strong, empowered product ownership capabilities within the organization. It is important that the AO and courts understand when local customization is necessary and when it is not. The AO should seek to determine when desired functionality is substantially similar across court types, when a single offering can be created to reduce complexity and cost, and when customization really is necessary. The AO should also develop a strong prioritization and feedback process to ensure that users are getting the most valuable functionality that meets their needs.

Strong user-centered design.

The new user interface and workflows should be based on deep user research and understanding. The notion of “requirements gathering” would go away. Instead users would participate in user research and provide feedback on system functionality, iteratively.

Technology

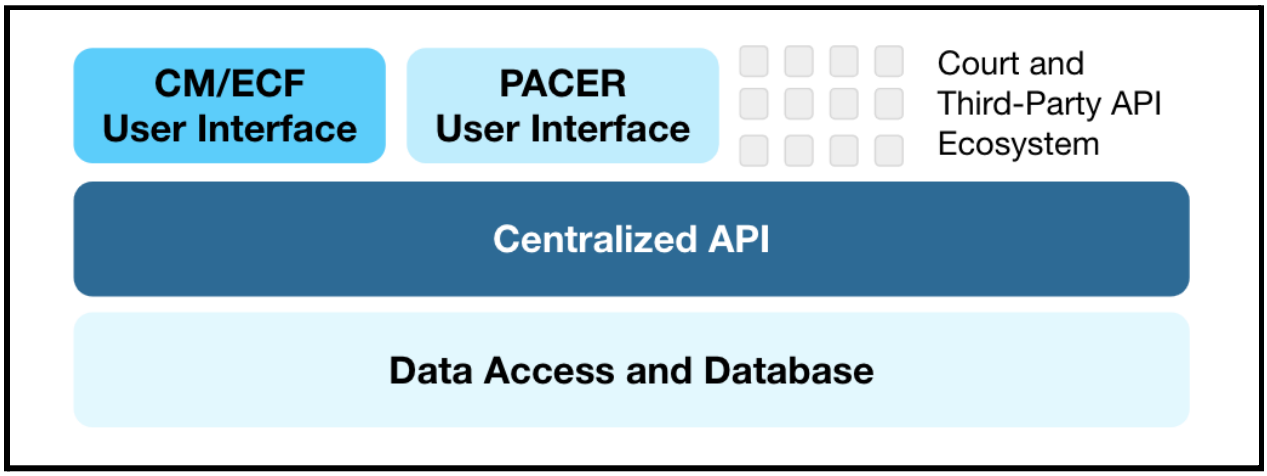
Our key recommendations are:

- Decouple the current monolithic application to a layered architecture with:
    - A user interface layer for CM/ECF, PACER, and court-developed applications;
    - A centralized API and business logic layer; and

    - A data access layer
- Enterprise-wide Findable, Accessible, Interoperable, and Reusable (FAIR) data standards using the National Information Exchange Model (NIEM) or similar
- Single base SQL schema
- Adopt appropriate data access technology for different data use cases
- Evaluate data cataloging and governance frameworks
- Evaluate and plan for cloud computing transformation

The CM/ECF application

The judiciary simplifies operations and encourages local innovation through APIs.



The judiciary should consider building a centralized REST[15] API. While this is a big shift from the decentralized model that CM/ECF is currently using and will require significant change management, there will be considerable gains in cybersecurity and simplified operations, while still encouraging court-level innovation.

The centralized API would be built, maintained, and operated by the AO and contractor teams. The use of APIs would simplify AO operations while still enabling an innovative ecosystem that is not dependent on the AO for intervention. In addition to the centralized API, court and third-party APIs would continue to enable court-level innovation and flexibility; support local, unique businesses processes; and connect CM/ECF to third-party products. APIs would provide a secure environment where courts could continue to innovate while allowing the AO to observe how data is being

[15] Representational state transfer.

consumed. In an environment where all courts would be using a common, core API, these locally created applications could be more easily shared with similar courts with similar business needs.

In the next phase, we would want to:

- Undertake additional exploration to understand who is using the existing ecosystem of third-party software that interfaces with Current/NextGen CM/ECF.
- Designate "developer advocates" and solutions architects to talk to these groups about how to switch to API.
- Build relationships with these groups and prepare them to transition to a centralized API.

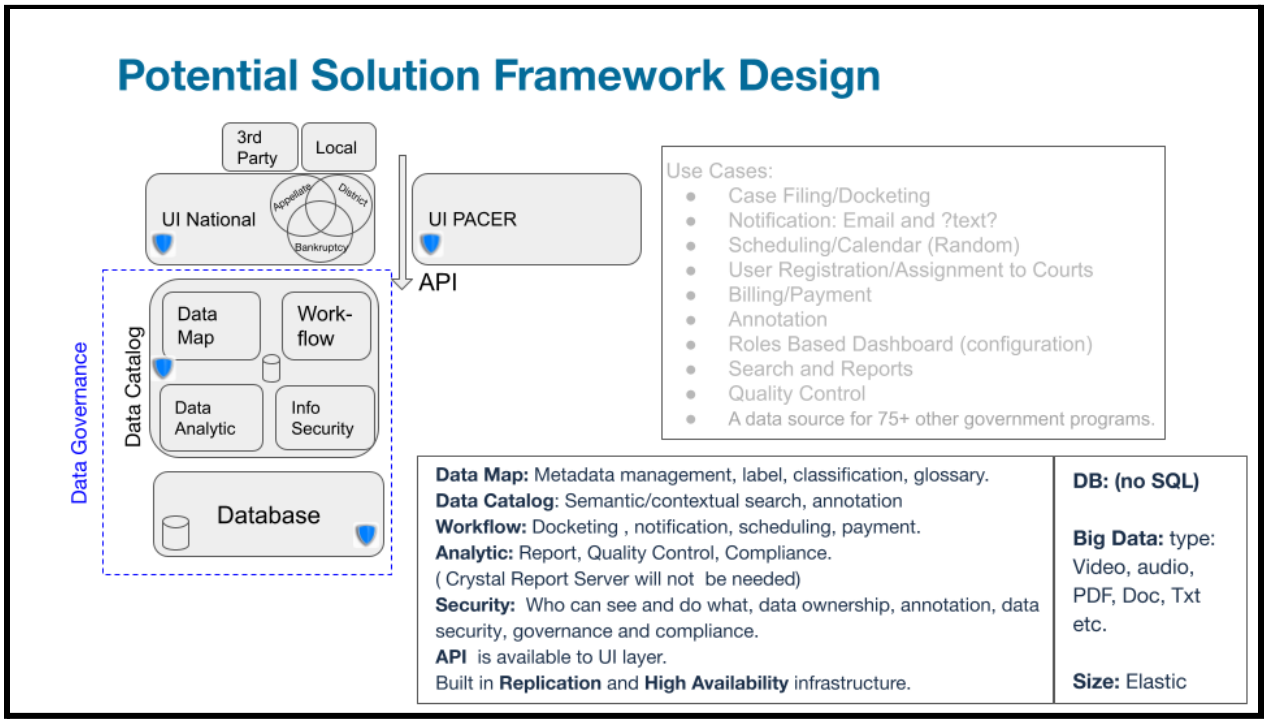
Data Standards & Architecture

We recommend that the judiciary develop and implement data standards to make data FAIR. Data standards should be established to improve the user experience of both CM/ECF and PACER users and enable data transfer capabilities within and between courts. We found a few existing standards in the CM/ECF ecosystem such as NIEM, Information Exchange Data Package for Bankruptcy Case Opening, and Appellate XML Tags.

Our research also identified several open source standards, solutions, commercial products, and technical reports that the judiciary should consider in building its future state solution. Listed below are few examples of open source solutions that would fit the judiciary's needs in the future state of CM/ECF:

- Apache Atlas provides open metadata management and governance capabilities for organizations to build a catalog of their data assets, classify and govern these assets and provide collaboration capabilities around these data assets.
    - https://atlas.apache.org/#/
- Apache Spark is a unified analytics engine for large-scale data processing.
    - https://spark.apache.org/
- The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing.
    - https://hadoop.apache.org/

Appendix E provides characteristics of an automated data catalog/governance framework solution that uses the above open-source solutions. Many commercially available solutions adopt this framework as well.



The figure above shows one potential solution design for the future state. This is by no means the only solution that could work for the judiciary's needs, but it provides a good starting point and direction for the judiciary to consider in making decisions about the future direction of CM/ECF.

As the figure demonstrates, it is recommended that the judiciary replace the current monolithic design with a new loosely coupled design. In short, this means that the user interface (UI) is decoupled from the business logic and API layers, which are, in turn, decoupled from the database layer.

During our research, we looked at the most common use cases of CM/ECF across the three court types (appellate, district, and bankruptcy), and mapped each of these most common use cases to a software component in the business logic layer. For example, contextual search is handled by the data catalog component, reports are handled by the analytic component, and docketing is handled by the workflow component. We understand that this list of common use cases is not an exhaustive accounting of all the courts' use cases, but this platform will allow developers to build new workflows

(i.e., use cases) beyond those listed here. We also do not necessarily recommend that the judiciary build an entirely new framework from scratch. The judiciary could procure some components and build others.

Align relational database schema and adopt appropriate database and storage solutions for the different needs in CM/ECF.

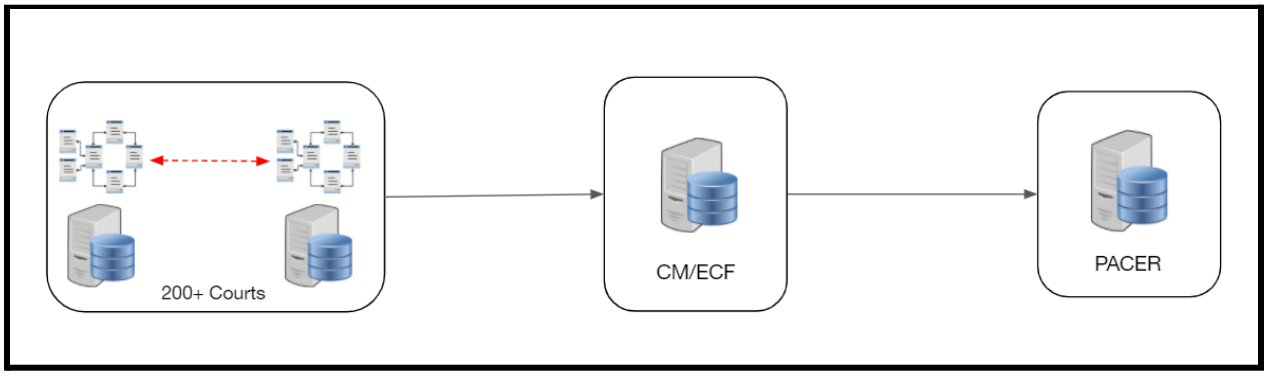
When the original CM/ECF system was built, it only required a database and storage solution that was capable of storing text, word, and scanned graphics files that were small in size. Today, the database and media storage needs of CM/ECF have changed significantly; courts now need to store files that are much larger in size, including PDF, editable PDF, and video and audio files. Additionally, the volume of data stored on the CM/ECF database has increased drastically in the past 30 years. CM/ECF now has more than one billion rows of data.

We did not perform a detailed analysis of the judiciary's database schema during the 11-week Path Analysis, but based on the user feedback we received and our technical review, we strongly recommend that the judiciary migrate from the Informix database to a modern database and a separate media storage solution that can handle the data types and volume the courts need.[16]

The current relational database design is causing poor CM/ECF performance. The schema has design issues including "over-normalization" which leads to sluggish performance. We believe there are opportunities for courts to share a common relational database schema and we recommend that the judiciary use a single relational database for the transactional data and day-to-day business needs. A single database does not mean there is one physical server. Rather, it means there would be one system with multiple replicates to handle simultaneous access from users.

To meet the specialized needs of PACER, the judiciary should consider using a full text search-optimized data solution. This database solution could be optimized for reads and for analytical needs. Importantly, this approach would provide the judiciary with a path to compliance with the proposed Open Courts Act. Since PACER allows the public to access much of the same data that is stored in CM/ECF, the judiciary can design a data solution to improve the system's overall security, performance, and accessibility.

[16] More recent Informix versions enhance its support for Data Warehouse and offer extensions to support data types that are not part of the SQL Standard.



Managing cybersecurity risks.

Cybersecurity risk is a business risk. CM/ECF downtime due to a cybersecurity breach would create a monumental disruption to court operations. Hackers will always try to exploit the vulnerability of a government system. The threats are real and a headline of a successful cyberattack on CM/ECF will weaken the public's trust in the judiciary. Moreover, the judiciary has legal and compliance obligations surrounding cybersecurity. Having a strong cybersecurity culture will reduce the judiciary's risk management costs and reduce the risk of disruptions to court operations. Cybersecurity is everybody's responsibility, from end users to staff in DevSecOps. Changes to organizational culture and operations are needed to ensure the confidentiality, integrity and availability (CIA) of the judiciary's data.

Development, deployment & operations

The judiciary should adopt a DevSecOps culture.

> *"DevOps is a culture, not a team."*

DevSecOps is a culture and approach to software development and delivery, cybersecurity, and operations. DevSecOps cultures emphasize the entire system: development, security, and operations are treated as interrelated and mutually reinforcing practices. DevSecOps culture values continuous feedback and believes in improving processes and products over time. DevSecOps cultures break down the barriers between traditionally siloed development, security, and operations groups, often by incorporating operations and security concerns earlier in the development process.

In addition to these organizational and communication traits, DevSecOps has a set of practices and tools that support the culture. These include Continuous Integration/Continuous Deployment (CI/CD), automated quality and security testing, and others. We will dig into each of these specifically in this section.

The judiciary should test the DevSecOps approach with a small team to learn about the specific challenges of the AO context.

We would like to test a DevSecOps approach with the judiciary by building a small, empowered team composed of members representing development, cybersecurity, operations, and the courts. This team will work in 2-4 week sprints using an iterative, agile process and will test a DevSecOps culture and approach within the judiciary and learn from the particulars of the context.

The judiciary should adopt CI/CD practices to support DevSecOps culture.

In developing, supporting, and maintaining the new system, the judiciary should use continuous integration and continuous deployment (CI/CD) to allow for significantly shortened lead times for the delivery of new features. Canary deployments[17] or blue/green deployments[18] can be used to minimize risk and downtime associated with deployments. A high-level visualization of these techniques can be found in Appendix H. In this future state, there would be no need for the AO to maintain and support multiple versions of the CM/ECF software because all courts would always be running the same version.

Courts would receive rule updates, security and reliability patches, and new features from the AO. Courts would no longer need to install and maintain their own versions of the software. This would reduce the amount of customization, installation, testing, and maintenance happening at each individual court and would free up court-level resources to work on higher-value tasks. Courts would no longer have direct access to modify source code, which is currently a high security risk; instead, the new platform would provide limited customization to courts and access to data via APIs. The system would also be easier for IT staff and contractors to maintain.

---

[17] Canary release is a technique to reduce the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users before rolling it out to the entire infrastructure and making it available to everybody.

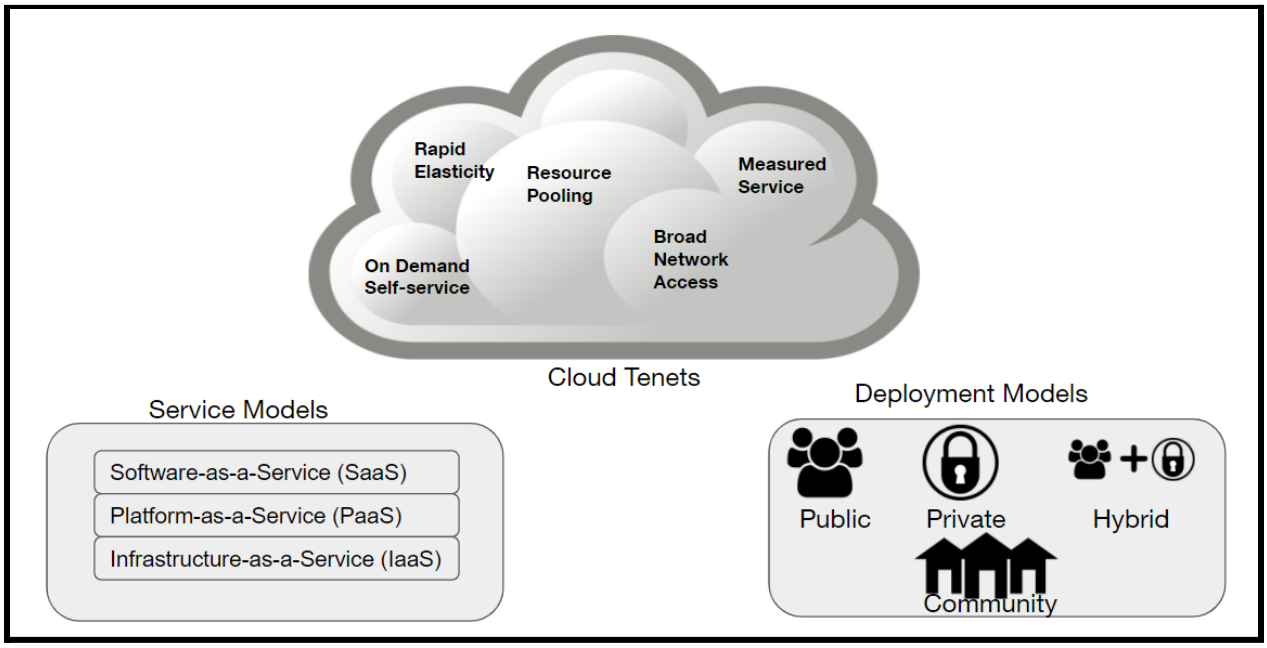
[18] Blue/green deployment is an application release model that gradually transfers user traffic from a previous version of an app or microservice to a nearly identical new release—both of which are running in production.

Infrastructure

Server and database management have been commoditized. Amazon Web Services, Azure and other cloud[19] providers use "utility" pricing. Migrating to cloud computing involves multiple phases and considerations, such as security and compliance impact, virtualization of servers, application updates, and utilizing cloud features, but we recommend that the judiciary move to the cloud.

Running data centers is not the judiciary's core business. There is an opportunity cost to the judiciary when spending time focusing on server and database management. Letting a cloud contractor handle this would enable the judiciary to focus more on the mission-delivery aspects of CM/ECF.

The judiciary should look into a cloud strategy as part of building new. Depending on user needs and other factors such as cybersecurity, pricing, accessibility, and scalability, the judiciary can decide which components to deploy to the cloud.



Opportunities to explore.

In future phases of work, we recommend that the judiciary also consider the possibility of a single case docketing process and other similar use cases/workflows. We understand that cases flow from the district and bankruptcy courts to the appellate

[19]http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

courts, and sometimes back again. When this happens, the parties and case documents (i.e., data) remain the same. Having a solution that allows for a seamless flow of information between court types would dramatically reduce the time-consuming processes for transferring cases between courts that exist today.

In future phases of work, we also recommend that the judiciary consider leveraging the U.S. Tax Court's open source case management system, which includes functionality for filing a petition, docketing, quality control, workflow, notes, calendaring, sealed cases, and public searching.[20] There may be opportunities to reduce development time and costs for the new CM/ECF system by leveraging this and other existing, free codebases.

In future phases of work, we recommend that the judiciary consider ways to continue and proactively foster an innovation culture. For example, allowing courts to "pitch" ideas to receive funding from the AO to test, learn, and scale products could help build an innovative ecosystem of shared products. There are several examples of other agencies using similar approaches. GSA's 10x[21] is one example.

Organizational context

Opportunity to add product ownership best practices to CM/ECF.

CM/ECF functionality should be developed using best practices in product ownership to ensure that the right functionality is being built to meet end-user needs. To implement product ownership best practices, the AO should develop strong, empowered product owners. Product owners are foundational to modern software development in government. These individuals are government employees and work daily with the development team and end users to set priorities and accept work. They protect the project scope and can help the judiciary understand where desired functionality is substantially similar across court types, when a single offering can be created to reduce complexity and cost, and when customization really is necessary. Lastly, product owners can help the judiciary build a strong prioritization and feedback process to ensure that users are getting the most valuable functionality that meets their needs.

[20] https://github.com/ustaxcourt/ef-cms
[21] https://10x.gsa.gov/

Technology alone will not solve the problem.

While we have discussed a lot of technology in this report, we want to emphasize the importance of the organization and its culture to the success of building and supporting a new system. Building a new system encompasses a lot more than writing the application code. A holistic approach is needed – that includes improving the AO's development, deployment, and operations processes; its cybersecurity posture; and the infrastructure supporting the CM/ECF system. These efforts will improve the judiciary's business agility. Appendix F shows the value drivers to increase business agility before and after building a new solution.

Open and frequent communication among stakeholders.

We also found that the judiciary has an abundance of smart, dedicated, creative, and innovative staff, and that most of them are ready for change. The judiciary was at the forefront of creating a digitized, more efficient, and better user experience with CM/ECF in the 1990s and there is a desire to continue building on that legacy.

> *"If the assessment is that CM/ECF should be scrapped because technology has advanced, then that's what we ought to be doing – more efficient, more elegant, less costly."*

However, the current CM/ECF system and the processes that support it reflect the siloed and complex communication structure among staff and different teams, departments, and courts. This is a common pattern in software development, known as "Conway's law."[22] In order for a software solution to function, multiple stakeholders (end users, development, cybersecurity, operations) must communicate frequently with each other. The judiciary can facilitate this change by encouraging open communication among the cross-departmental and cross-functional teams.

Agile acquisition strategy

The CM/ECF software support contracts are in a relatively good place, but a few tweaks will ensure that the AO receives quality work from its contractors.

[22]According to Conway's Law, any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

In addition to the best practices the AO is already using that are described above, future acquisitions should incorporate the following elements to ensure that the AO is receiving high quality and valuable work from its contractors.

Hire a software development team as a pre-formed unit and use modular contracting methods.

The AO should award contracts for the services of a software development team as a pre-formed unit, especially when building new software. Hiring a development team as a unit allows the contractor to staff the right skills, experience, and temperaments to the team based on the AO's stated needs. The contractor has more performance information about its employees than the AO could discern by reviewing a resume. Also, issuing a contract for a pre-formed team will eliminate deflection of accountability, since delivered code and features are the responsibility of the development team and the development team is from a single contractor. The average cost of hiring a contractor team varies depending on the place of work (i.e., remote or on-site), whether team members are required to obtain security clearances, how many members are on the team, and other factors. The table below provides an example of the cost of a full-time, eight-person software development team for 12 months of work.

Labor Category	Hours	Rate	Base Period (12 months)
Software Developer, Lead (1 full time equivalent (FTE))	1,920	$136.00	$261,120.00
Software Developer (3 FTEs)	5,760	$106.00	$610,560.00
Designer (1 FTE)	1,920	$138.00	$264,960.00
User Researcher (3 FTEs)	5,760	$131.00	$754,560.00
ODCs (non-travel)	------	------	$5,000.00
Travel	------	------	$10,000.00
Total			**$1,906,200.00**

This estimate is also based on the team performing a specific increment of work as part of a modular contracting approach (FAR 39.103).[23] Modular contracting is the preferred approach for the acquisition of information technology. It aligns with agile software development principles because it supports the development of a loosely-coupled system in increments versus awarding a single contract to build a monolith software system.

We recommend starting with no more than two development teams for the first increment or "slice of work." As the AO becomes more adept at working with contractor-formed agile development teams that support specific increments of work, the AO could contract for the services of additional development teams to support other increments of the system. The E&I phase will inform the slice of work on which the development team will work, as well as the number of development teams needed to perform the work.

Include performance metrics in the AO's contracts.

Requiring that the vendor's code actually works (i.e., working code) should be included in the contract as a deliverable. Including working code as a deliverable ensures that software that meets actual user needs is the focus of the contract, not just the contractor's time.

Another best practice is to include quality assurance surveillance plans (QASP), similar to the one in Appendix G, in custom software development contracts.

Quality Assurance Surveillance Plan (QASP) Excerpt https://derisking-guide.18f.gov/qasp/	
Deliverable	**Performance Standard**
Tested Code	Code delivered under the order must have substantial test code coverage and a clean code baseVersion-controlled, public repository of code comprising the product, which will remain in the government domain
Properly Styled Code	GSA 18F Front-End Guide

[23] Also for the purposes of this table, full time equates to 1,920 hours per year, and the average rates were found using CALC, a tool that provides GSA labor rates for services offered under GSA schedule contracts (https://calc.gsa.gov). This estimate is not intended to capture the total cost of software development and ownership for the new system and thus should not be extrapolated or otherwise used to estimate the cost of building the recommended new system.

	https://engineering.18f.gov/frontend/
Accessibility	Web Content Accessibility Guidelines 2.1 AA standards
Deployed	Code must successfully build and deploy into staging environment
Documented	All dependencies are listed and the licenses are documented. Major functionality in the software/source code is documented. Individual methods are documented inline using comments that permit the use of documentation-generation tools such as JSDoc. A system diagram is provided. https://jsdoc.app/
Security	OWASP Application Security Verification Standard 4.0, Level 2 https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf
User Research	Usability testing and other user research methods must be conducted at regular intervals throughout the development process (not just at the beginning or end)

A QASP is the plan the government follows to verify that the goods or services provided by a contractor satisfy the government's needs by meeting specific performance targets. The QASP should be included in contracts to make failure to meet the standards of the QASP enforceable under the terms and conditions of the contract. Including the QASP also ensures that the contractor builds and maintains the same quality level throughout the life of the contract. The QASP should be tailored for each acquisition but we recommend, at a minimum, that QASPs for custom software development include elements like deployability, documentation, security, accessibility (i.e., 508 compliant), and based on user research.

While adding QASPs to the AO's contracts for software development services is a step in the right direction, it is still just a tool and it requires someone with relevant skills to enforce it. The AO should find a person (preferably a government employee) with the technical skills to review the contractor's code to ensure it has good code hygiene and is in compliance with the performance standards of the QASP. We call this person the technical lead (or tech lead) and they would work closely with the Product Owner (PO) and Contracting Officer's Representative (COR) to inspect and accept the contractor's deliverable (i.e., working code that complies with the QASP). The tech lead, PO, and

COR could be the same person or three different people, but it is very important that all the relevant skills and responsibilities are represented on the team.

T&M contracts are the preferred contract type for agile software development.

Adding code as a deliverable and adding a QASP will strengthen the AO's T&M contracts and increase its ability to hold contractors responsible for the code they deliver, while maintaining flexibility for the AO to make in-scope project pivots without the need for contract renegotiations or modifications, which can take a long time.[24]

By definition, agile software development does not use detailed specifications or plans to develop software. The objective of the development team is known at the outset (i.e., develop software) but the *how* (i.e., what is the priority order of features), *when* (i.e., how long will each feature take to develop), and *who* (i.e., which labor categories will be needed to develop the features) is largely unknown at the beginning of the contract. T&M contracts also enable the AO to immediately end a contract by removing all work from the backlog, which in turn removes all work that the vendor team has to do. With T&M contracts, contractors can only invoice for actual time worked and if the team has no work to perform, then the contractor cannot bill the AO. This is different from the formal termination for cause or convenience clauses included in all government contracts. 18F offers acquisition consulting services that include methods for awarding contracts in 90 days or less that we have used at various agencies across the government.

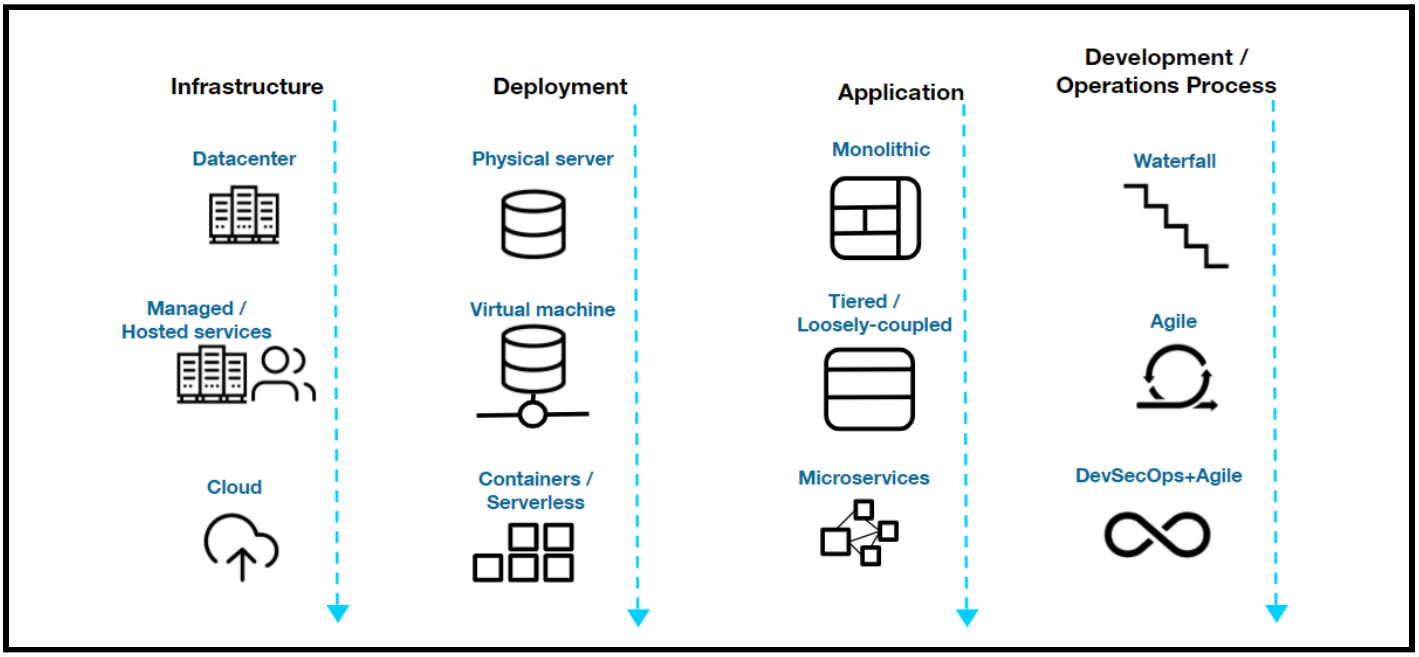
[24] Firm-fixed-price contracts do not offer this flexibility, especially when building a new system. Rather, firm-fixed-price contracts are ideal for "acquiring other supplies or services on the basis of reasonably definite functional or detailed specifications" (FAR 16.202).

6. Summary

Continue to evolve in line with industry trends.

The graphic below shows the direction of current industry trends, and many of our tactical recommendations focus on how the judiciary can better align itself with these trends and continue to evolve in the same direction as industry offerings. This approach also aligns with Issue 5 of the Federal Judiciary Strategic Plan:

> *"How can the judiciary develop, operate, and secure cost-effective national and local systems and infrastructure that meet the needs of court users and the public for information, service and access to the courts?"*



Meeting with broader stakeholders.

There are many technical and experienced AO and court staff, and we recommend that the judiciary leverage their domain expertise during the upcoming planning and building phases.

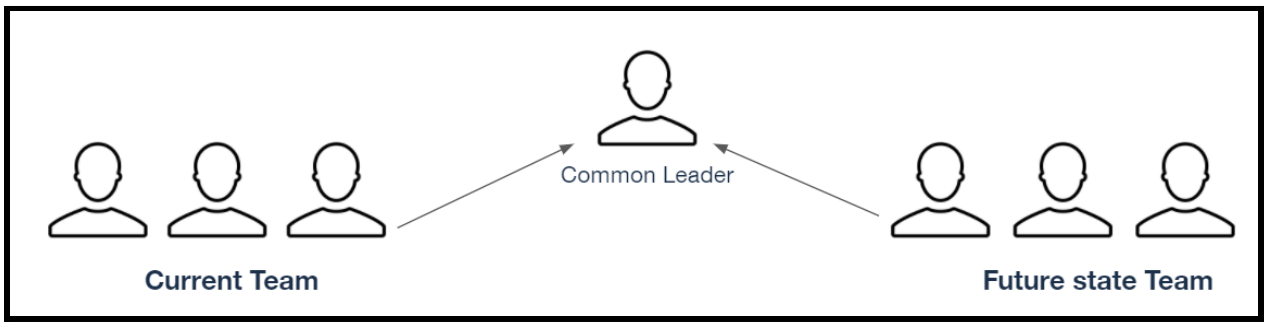
- Host information gathering sessions where the recommendations in this report are presented and feedback is openly solicited.

- Encourage IT stakeholders to make recommendations on modularizing components and to offer modernization and migration options.
- Conduct technical architecture visioning sessions with key CM/ECF stakeholders.
- Review key local modifications and third-party integrations and determine if and how they can be ported into future state solutions. We found a few programs, such as CHAP, ACMS and CEO, that have already adopted some of the methodologies and technologies we are recommending to produce features that courts need. These local modifications can be a source of inspiration, and could possibly serve as a starting point, for the future state solution.
- Using our recommendations, 18F can also help the judiciary conduct more market research and develop solicitation(s) for commercially available solutions and services.

The AO should build two teams with a common overarching identity.

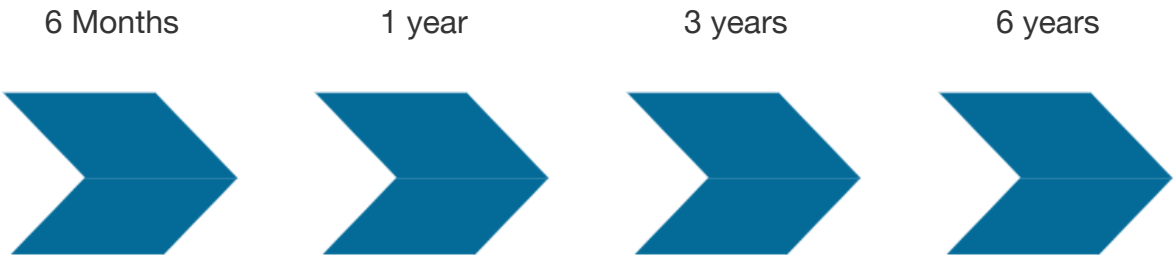
We recommend that the AO have two teams that report to a common leader. The Current Team should continue to support and improve the current system, as it is crucial to keep the current system operational, to support court business, during the new build phase. One priority for this team should be to fix the failover problem, as it affects the servers of all 200+ courts.

The Future State team should be empowered to adopt the new methodologies discussed above to build, test and iterate, while focusing on user needs. The Future State team would need some domain-knowledge support but should be independent in building the new system using DevSecOps, CI/CD, and agile methodologies.



7. Conclusion and roadmap

The roadmap below gives an overview of what could happen in the next 6 months to 6 years; however, it is highly dependent on many factors over which 18F and the judiciary have no control. It is up to the judiciary to adopt and pivot constantly. We are confident the new system can be delivered much faster than NextGen, if the judiciary acts soon, follows our recommendations, and continues to evolve with industry trends. 18F can continue working with the judiciary on this journey.



- 18F E&I: one DevSecOps team to prototype a use case applying a user-centric, agile methodology.
- Hold information sessions with various stakeholders and court staff for feedback and suggestions.
- Review the different options presented. The judiciary may need more market research & assistance with issuing an award.
- Review organizational and cultural changes needed to support building new initiatives.
- Present findings to other stakeholders.
- Make funding arrangements.

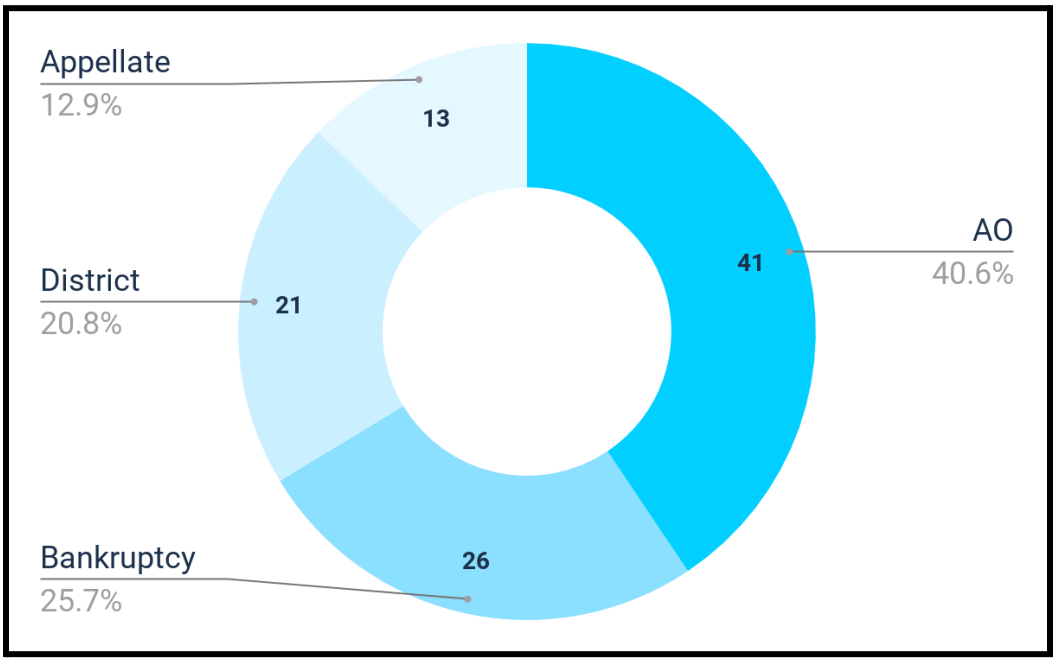
- Decide how to build the new system and gain support.
- Hire a new team and continue to align the organization.
- Decide which parts of the system to buy and which to custom build, specifically the UI and business logic and database layers.
- Solicit 2-3 pilot courts.
- Start to build the new system.
- DevSecOps team works closely with courts during the build-test-repeat cycle.
- Decide whether to build a solution for all court types, or a solution for each of the 3 court types.
- Application and database deployment strategy, e.g., cloud or not.

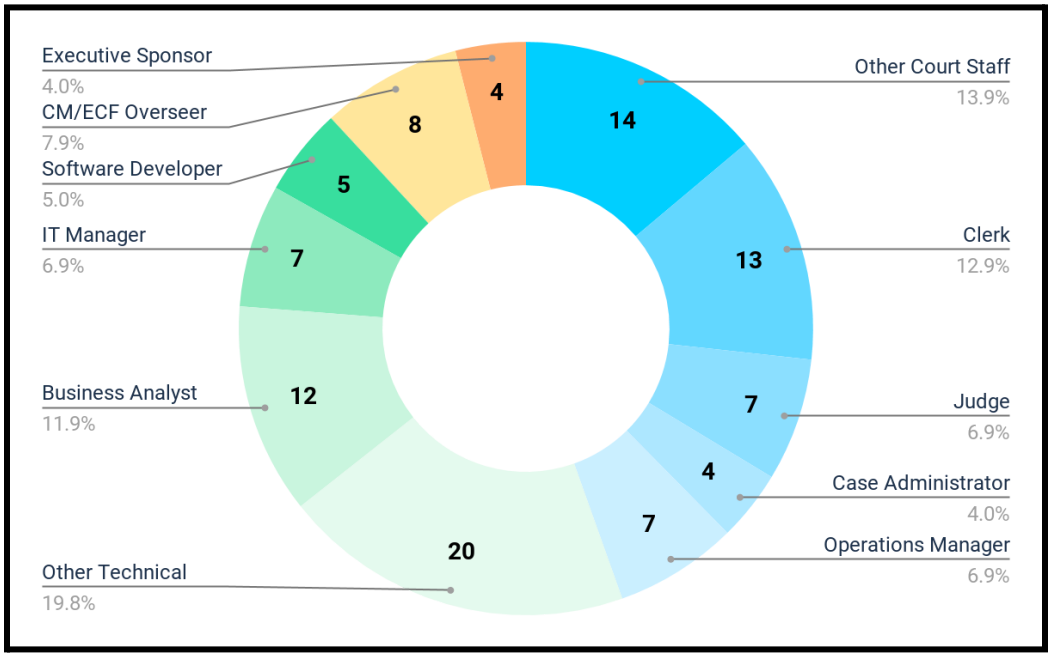
- Complete a system/solution for the pilot courts.
- Formulate a migration and training plan for the new system.
- Start to install the solution to the other courts of the same type, court by court.
- If needed, start to customize/update the solution to other court types.
- Establish new requests for enhancement (RFE), prioritization, deployment, and support organization/process.

- Aim to have at least 50% of courts migrated to the new solution.
- Continue to migrate courts to new solutions.
- Continue to evaluate user needs, new technologies, and organizational processes, and to pivot and adapt constantly.

Appendix A: Interview group statistics

The two charts show the group statistics of our 77+ interviews with 100+ participants.





Appendix B: the AO's two data centers

The following diagram shows the AO's existing data centers. Each data center acts as a backup to each other.

Appendix C: SWOT Analyses

1. CurrentGen/NextGen

Strengths ● **Business as usual.** ● It has been running for a long time (20+ yea that **people are comfortable.** ● No new development costs. ● "If it ain't broken, why fix it."	**Weaknesses** ● **User needs are not being met.** ● **High security risk factors** ● **No new features will be delivered easily.** ● **Just delaying the inevitable.** ● 20+ year old system & technology is difficult to maintain. ● High support costs. ● All efforts are focused on keeping the system stable and running (security patches and critical bugs). ● Low productivities, wasting time with too much manual work. ● System architecture does not encourage innovation.
Opportunities ● **Current staff have the technical skills and domain knowledge necessary to identify and fix issues,** continue the implementation of, and develop new modules for the existing system. ● AO can stabilize and improve the current infrastructure and utilize the virtual machine and high availability technology offered by VMware to fix the failover issue.	**Threats** ● **Nearly impossible for the Current/NextGen system to meet the mandates of the proposed Open Courts Act.** ● **Security threat is real and it will be damaging if it is breached.** ● **Looming staff/skill succession problem.** ● Current technology, coding language, database, and browsers are being deprecated. ● Huge technical debt.

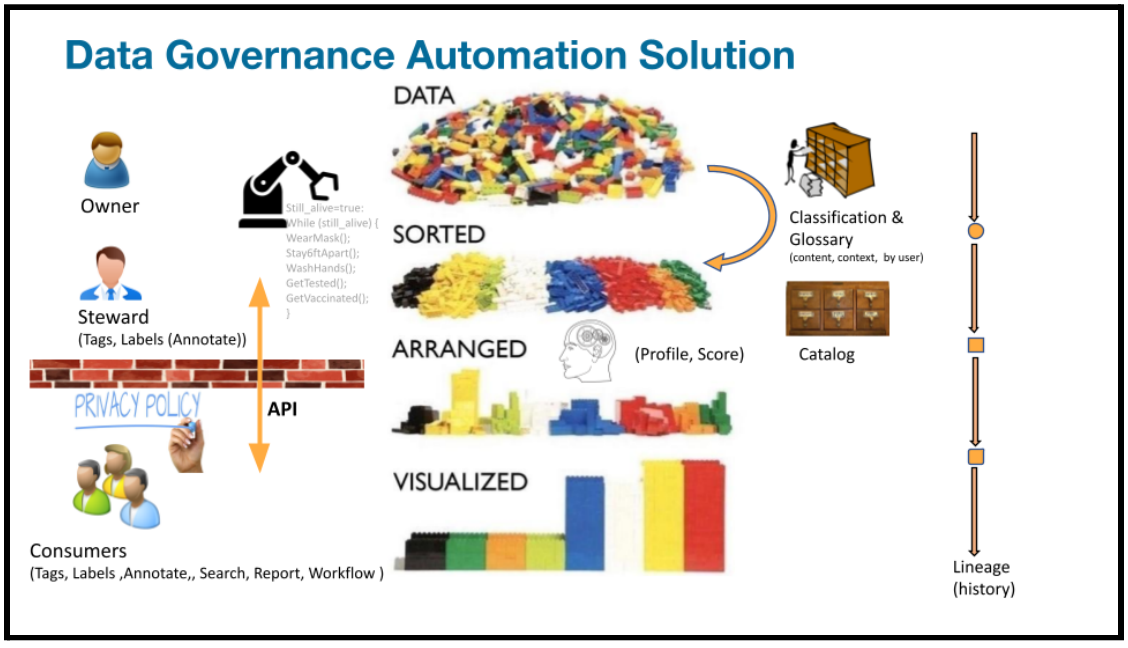
2. Build New

Strengths	Weaknesses
• **User-centered design meets user needs**. • **Feature built, delivered to user quickly and iteratively.** • **Security is built into the development process (DevSecOps).** • **Data governance is built in.** • New solution, technology, infrastructure, methodologies, processes and aligned organizationally. • High-performance, highly available & intuitive solution. • Seamless upgrades, far less support costs.	• **Need mindset and culture changes on many fronts.** • **Need initial investment (CapEx) and a new set of people with different technical skills.** • Need to run two systems concurrently during development and migration. • Need to evaluate the impact of other systems accessing the database.
Opportunities	**Threats**
• **Designed to meet the Open Courts Act proposed mandates & security compliance and policies.** • **Necessary solutions, frameworks and skill sets exist and are readily available in the commercial marketplace.** • Opportunity to innovate by taking risks and making mistakes, without impacting court operations. • Involves court experts in the journey. • Taking advantage of current federated installation, the migration risk will be minimal (court-by-court roll out).	• **Court users may have high expectations, not every court will have the new system at once.** • **Public scrutiny on the delivery of a new system.** • Culture change that encourages innovation – allows an autonomous team to innovate by taking small bets, sometimes succeeding, sometimes making mistakes, and always failing forward. • Need appropriate risk management.

Appendix D: Automated data catalog/ governance framework

Listed below are some of the characteristics of the recommended automated solution.

a. Data Owner: each data asset can assign the owner who can decide what to share and with whom to share it.
b. Data Steward: data governance role within an organization that is responsible for ensuring the quality and fitness for the purpose of the organization's data assets, including the metadata for those data assets.
c. Privacy Policy: Role Based Access Control (RBAC) can be achieved by setting policies for individuals and groups.
d. Automation (Robot arm): Background automation functions.
e. API: information can be accessed through API.
f. Cybersecurity: Built in security model.
g. Analytics (Human head): It analyzes the data and provides quality information to the user quickly and accurately.
h. Classification/Glossary/Catalog: It provides fast content and context search to the users.
i. Lineage: It shows the history of data asset movement, from point A to B to C. In other words, it shows the data lifecycle.



Appendix E: Business value drivers

This table shows the value drivers to increase business agility before and after building a new solution.

Before	After
• New features takes too long to dev and roll out. • Poor alignment of IT services to user/business needs • Lack of IT flexibility (technical debts, stale architecture) • Unable to leverage new technology in a timely manner. • Lack of standards for IT processes and technology • Unable to react in a timely manner to user needs	• IT organized and managed as a shared service, seen as partner to the business and users. • Define and agreed upon SLAs • Culture of innovation • Ability to integrate and adopt new technologies • Increase productivity through prioritized project identification and implementation • Flexible IT processes and systems that enable collaboration between IT and business
Negative Consequences	**Positive Business Outcome**
• Conflict between business/users and IT • No SLA or poor SLA achievement • Rogus use of external IT resources • Organizational malaise-" why bother, it will take so long" • High attribution of IT staff • Expensive, purpose built applications (local mods)	• IT spending in alignment with industry benchmarks • Achieved SLAs • Increase employee engagement and job satisfaction-lower attrition • Improve time to provision • Shorter development lifecycle (Agile) • Increase in business satisfaction with IT • High cost avoidance

Appendix F: Sample quality assurance surveillance plan (QASP)

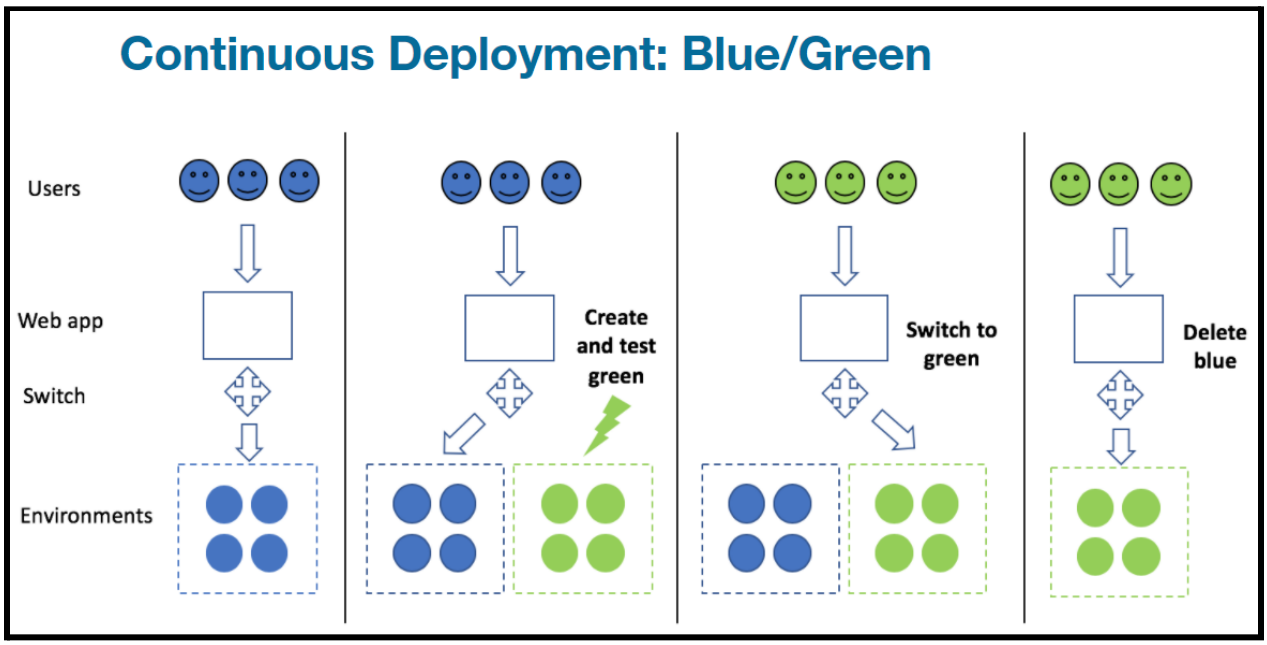
A QASP is the plan the government follows to verify that the goods or services provided by a contractor satisfy the government's needs by meeting specific performance targets.

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 Version-controlled, public repository of code comprising the product, which will remain in the government domain	Minimum of 90% test coverage of all code	Combination of manual review and automated testing
Properly Styled Code	GSA 18F Front-End Guide https://engineering.18f.gov/frontend/	0 linting errors and 0 warnings	Combination of manual review and automated testing
Accessibility	Web Content Accessibility Guidelines 2.1 AA standards	0 errors reported using an automated scanner, and 0 errors reported in manual testing	Pa11y https://github.com/pa11y/pa11y
Deployed	Code must successfully build and deploy into staging environment	Successful build with a single command	Combination of manual review and automated testing

Documented	All dependencies are listed and the licenses are documented. Major functionality in the software/source code is documented. Individual methods are documented inline using comments that permit the use of documentation-generation tools such as JSDoc . A system diagram is provided https://jsdoc.app/	Combination of manual review and automated testing, if available	Manual review
Security	OWASP Application Security Verification Standard 4.0, Level 2 https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf	Code submitted must be free of medium- and high-level static and dynamic security vulnerabilities	Clean tests from a static testing SaaS (e.g npm audit) and from OWASP ZAP, along with documentation explaining any false positives https://docs.npmjs.com/cli/v7/commands/npm-audit https://owasp.org/www-project-zap
User research	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 contractor's research plan	Manual review

Appendix G: Visualization of Blue/Green and Canary Deployments

Blue green deployment is an application release model that gradually transfers user traffic from a previous version of an app or microservice to a nearly identical new release—both of which are running in production.



Canary release is a technique to reduce the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users before rolling it out to the entire infrastructure and making it available to everybody.