# Civil Servants as Builders: Rethinking Digital Transformation Beyond Vendors and Centralized Agencies

Prashant Sharma*
prashant.sharma@berkeley.edu
Independent Researcher
Sacramento, California, USA

## Abstract

Government agencies work to serve the public, guided by necessary safeguards such as detailed procurement processes and strict IT security standards. These constraints, while essential for safeguarding public resources and data, can hinder civil servants outside IT teams from deploying low-cost, lightweight tools. This paper proposes a limits-aware platform that enables civil servants to develop and securely share small, domain-specific applications using Jupyter Notebooks and preapproved open-source libraries. Hosted entirely within internal government networks and governed through peer review and sandboxed execution, the system balances innovation with accountability and information security. Unlike centralized digital teams or vendor-led approaches, this model empowers public servants to build and maintain tools directly informed by frontline realities. By working within existing institutional constraints rather than bypassing them, the platform supports a sustainable, bottom-up model of digital transformation aligned with the values of the Computing within Limits community.

## Keywords

Digital Government, Information Systems, Public Administration, Technology Policy, Digital Transformation, Information Security

## 1 Introduction

Government agencies operate under important mandates: to safeguard public resources, ensure procedural fairness, and uphold high standards of security. To meet these obligations, digital tools in the public sector are typically subjected to rigorous procurement procedures and strict IT oversight. These institutional safeguards exist for good reason—they are designed to prevent waste, ensure equitable access to contracting opportunities, and reduce risks to sensitive data and infrastructure. However, they also have a cost: civil servants who identify clear opportunities for internal automation or lightweight tooling often face long procurement cycles or compete for limited IT bandwidth already stretched thin.

Procurement departments, for instance, are responsible for ensuring that every expenditure serves the public interest and that projects do not bypasses fair competition. Similarly, IT teams act as custodians of security and stability, ensuring that all deployed tools conform to internal standards and do not introduce cybersecurity vulnerabilities. In environments where time, staffing, and attention are chronically limited, both functions tend toward caution. This tendency, while understandable, can unintentionally suppress creativity, especially when civil servants propose novel solutions that fall outside conventional procurement or IT service delivery pipelines.

Today, many civil servants possess technical skills, such as scripting in Python or R, querying internal databases, or creating dashboards using open source libraries. However, these skills often remain siloed within individuals or teams. When a civil servant builds a script that automates a complex task or provides interactive visualizations, there is often no sanctioned pathway to scale that solution. Formalizing even a simple application can require navigating approvals, documentation, security reviews, and infrastructure constraints that are disproportionate to the complexity of the tool. As a result, practical innovations built by domain experts often remain trapped in local files or are discarded when the staff member leaves. In addition, some government IT departments may interpret security controls, such as those outlined in NIST SP 800-53 [8], as requiring strict oversight of programming environments, resulting in outright bans on tools like Jupyter Notebook, Python, or R for non-IT teams.

This paper proposes a speculative but feasible system that enables civil servants to safely build and deploy internal tools within institutional limits. In doing so, it gestures toward a broader shift in government digital transformation, moving beyond reliance on external vendors or centralized digital teams. The proposed platform allows civil servants to create applications in Jupyter Notebooks and deploy them as secure, interactive web applications accessible exclusively to authenticated users on the government's internal network, whether on-site or connecting remotely through a secure VPN gateway. Key features include a curated list of preapproved Python, R, or other relevant language libraries, automated sandbox execution, lightweight peer review, and internal-only web deployment. These mechanisms are intended not to replace institutional oversight, but to make it more scalable, allowing frontline staff to contribute low-risk digital tools while maintaining accountability and information security.

Rather than presenting digital transformation as a process that must be outsourced to vendors or specialized IT teams, this paper argues for a shift in perspective: one in which civil servants already inside the system are empowered to create sustainable, domain-specific applications under constrained conditions. This shift reflects the broader values of the Computing within Limits

*The author is a Research Data Specialist at the California Department of Food and Agriculture (CDFA) at the time of writing. This paper represents independent research conducted outside the scope of official duties and does not reflect the views, policies, or endorsements of the State of California or the CDFA. The author's prior experience includes work at a startup that collaborated with government agencies in Canada and the United Arab Emirates. All scenarios and observations presented are hypothetical, based on general knowledge of public sector processes, and do not reflect internal practices of any specific department or agency, including those the author has been affiliated with.

*LIMITS '25,*
2025.

community, working within institutional, infrastructural, and ecological boundaries while prioritizing resilience, adaptability, and the reuse of local knowledge. The sections that follow articulate the structural barriers to such an approach and how the proposed system works with, rather than around, those constraints.

### 1.1 Structural Barriers to Digital Innovation

Despite growing technical fluency among civil servants, systemic and organizational barriers often prevent frontline innovation from taking root. These barriers are not primarily technological; they are deeply embedded in the structures of public administration, which tend to prioritize procedural integrity, risk minimization, and stability over agility and experimentation.

Digital transformation in the public sector frequently proceeds through external procurement. While necessary for ensuring fairness and fiscal accountability, procurement processes are poorly suited to iterative or exploratory work. Edler et al. [3] note that procurement regimes emphasize compliance and predictability rather than creativity or impact. This orientation often makes it difficult to contract for lightweight, open-ended tools or small-scale internal prototypes. Proposals that do not fit neatly into formal scopes of work or budget categories may be filtered out early, regardless of their practical utility.

At the same time, government IT departments face pressures to maintain operational stability across critical infrastructure. Bozeman and Kingsley [1] describe how public sector organizations develop strong cultures of risk aversion—where the political and reputational consequences of software failure far outweigh the rewards of incremental success. As a result, even technically feasible proposals involving new platforms, packages, or data access methods are often denied. This caution systematically excludes open-source or experimental tools, even when they are popular, well-documented, and widely used in academia or industry.

Furthermore, digital innovation is not analogous to constructing a physical asset, such as a bridge or a building, that can be completed and then handed over for maintenance. Mergel [5] emphasizes that digital systems require continual refinement, adaptation, and support. When digital services are delivered entirely through third-party contracts, agencies often lack the institutional memory or technical fluency to modify them in response to shifting policy or operational needs. This dependency leads to brittle systems and long procurement delays for even minor updates—an unsustainable model under conditions of resource constraint.

Legacy systems further intensify the challenge. As noted by the UK House of Commons Public Accounts Committee [9, 14], much of government infrastructure relies on aging software that is difficult to modernize and expensive to maintain. These systems consume the bulk of IT budgets, leaving little room for experimentation or small-scale innovation. Moreover, their rigidity often requires new tools to conform to outdated standards, thereby limiting what frontline developers can feasibly build.

These constraints are not temporary; they are structural. Yet as the Computing within Limits community has emphasized, working within limits does not mean abandoning innovation—it means redirecting it [10, 11]. Digital infrastructure should be made resilient by embedding it in local practice, enabling small teams to operate and adapt their own tools without continuous reliance on external vendors or high-overhead processes. While much of the Computing within Limits community has focused on ecological and material resource constraints of future, some scholarship argues that the scenarios anticipated by limits community are not just future concerns but present realities in many domains. Chen [2] proposes that limits-aware computing should increasingly engage with real, immediate problems to make a difference today while preparing for potential future societal crisis, a perspective complemented by Tomlinson et al.'s [12] work on *collapse informatics*. Addressing structural barriers to small-scale digital innovation in government embodies many characteristics of a limits context. In fact, solving today's deeply embedded limits problems may offer a better foundation for resilience than designing for hypothetical futures—because the latter are difficult to predict. The system proposed in this paper thus contributes to the growing body of work [2, 7] that views collapse not as a singular future event, but as an ongoing condition that demands adaptive, embedded responses from within existing structures.

## 2 System Goals and Assumptions

The proposed system is designed with several key assumptions and goals:

- **No Vendor Lock-In** A foundational assumption of the proposed platform is the elimination of vendor lock-in and reliance on proprietary low-code environments, which pose significant risks to government IT sustainability. By exclusively using open-source tools like Jupyter Notebooks and deploying applications within secure internal networks, the platform ensures long-term institutional control, modifiability, and resilience without dependency on commercial vendors. Unlike proprietary low-code platforms, which often suffer from opaque licensing, limited long-term support, and inferior documentation and quality compared to open-source ecosystems, this approach leverages the robust, community-driven standards of open-source software.
- **Empower Domain Experts:** Civil servants possess deep, situated knowledge of government workflows, policies, and public needs—insights that external vendors often lack. Many already write analysis scripts in languages like Python or R to meet their day-to-day needs, yet systemic barriers prevent this technical ability from being channeled into broader digital innovation. By enabling civil servants to build and deploy tools themselves, the platform recognizes that domain expertise is a critical asset, not something to be outsourced. Unlike traditional vendor-driven development, which often struggles with costly requirement gathering and miscommunication cycles, empowering internal experts allows for iterative, agile development directly informed by frontline realities. In the spirit of Computing within Limits, this approach values local, embedded knowledge over external consultancy models, fostering systems that are better adapted to the environments in which they must operate.
- **Interactive Applications:** The platform supports the creation of rich, interactive web applications that allow users

to input parameters, explore data, visualize results, and export outputs—all without requiring them to engage with underlying code. This system makes civil servants' technical work accessible to non-programmer colleagues across departments. By transforming notebooks into interactive applications, the system bridges the gap between code and operational tools, allowing for responsive feedback loops and rapid iteration. In constrained environments where formal IT resources are scarce, such lightweight applications offer a practical path to increasing internal digital capacity without the need for heavy, vendor-led IT infrastructure investments.

- **Privacy and Security:** All applications created on the platform are securely hosted within the government's internal network, ensuring that sensitive data remains protected and accessible only to authenticated users. To minimize security risks, developers are restricted to a vetted set of preapproved libraries, with a streamlined workflow for requesting and approving new libraries when necessary. This structure acknowledges the legitimate concerns of IT security teams while still promoting flexibility and innovation. Furthermore, cross-departmental collaboration on maintaining the shared library list reduces redundant overhead and accelerates safe experimentation. For example, multiple departments within a state government might collaborate to maintain a shared list of approved libraries. From a LIMITS perspective, this security model accepts the inevitability of constraints—legal, ethical, and infrastructural—and designs innovation pathways that work within them, rather than ignoring them.
- **Lightweight Governance:** Rather than imposing heavy approval processes typical of traditional IT deployments, the platform emphasizes lightweight, efficient governance mechanisms. Each application undergoes a quick but meaningful peer review, modeled after common software engineering practices like GitHub pull requests, to catch obvious issues and ensure basic quality standards. Version control ensures that any changes are tracked and recoverable, and audit trails log who uploaded, reviewed, and approved each application, ensuring accountability without bureaucratic friction. This approach aligns with LIMITS values by creating governance structures that are resilient and sustainable under conditions of limited administrative and technical capacity. It demonstrates that good governance need not be synonymous with delay.
- **Cultural Transformation:** By enabling civil servants to build, deploy, and maintain their own digital tools, the platform challenges the entrenched notion that public sector innovation must always flow through formal IT procurement or external vendors. It reframes civil servants as capable technologists who, given the right platforms and safeguards, can drive meaningful change from within. This reorientation not only increases organizational agility but also fosters a culture of continuous learning, creativity, and ownership among public employees. Consistent with LIMITS thinking, the platform recognizes that systemic resilience is best achieved not by relying on external interventions but by cultivating internal capacities that can adapt and evolve under constraint. Civil servants are not passive users of technology; they are, and must be, its stewards.

## 3 Inspiration

This system is conceptually inspired by existing tools such as:

– Mercury [6] : an open-source framework for converting Python Notebooks into shareable web apps without learning frontend frameworks.
– Voilà [13]: a tool that turns Jupyter notebooks into standalone web applications.

Unlike these tools, the proposed platform is explicitly designed for a government context and limits: focusing on internal network-only deployment, preapproved libraries, sandbox security, audit trails, and peer governance.

## 4 System Overview

The core philosophy of the system is to work with, rather than against, the institutional limits present in government settings—recognizing that procurement rules, IT restrictions, and limited technical support are enduring realities, not temporary obstacles. The platform's use of Jupyter Notebooks reflects this ethos: they are already widely used for internal data work, are open-source and auditable, and avoid the risk of vendor lock-in—a critical concern for sustainability in public institutions. Rather than relying on proprietary low-code solutions, the platform prioritizes transparency, modifiability, and long-term institutional control. Its architecture distributes responsibility through peer review and versioning, aligning with open-source development norms and reducing burden on central IT. The system can be built using open-source technologies, such as Docker and Kubernetes, which can be shared as a reusable template across government agencies.

Figure 1 illustrates the full pipeline from notebook authoring to live application deployment.

### 4.1 Authoring the Notebook

Civil servants initiate the process by writing Jupyter Notebooks that encapsulate their data analysis, internal API interactions, visualizations, and workflows. The focus remains on domain logic and usability rather than web development. To make notebooks interactive, authors can add lightweight configuration metadata, such as a YAML header specifying which variables should be exposed as user inputs (e.g., dropdown menus, sliders, or file upload fields).

This approach builds on lessons from open-source frameworks like Mercury [6] and Voilà [13], which demonstrate that notebooks can serve as viable, low-code front-end interfaces. By lowering the barrier to building interactive applications, the platform empowers civil servants to convert internal analyses into reusable tools for broader teams, without needing specialized web development skills.

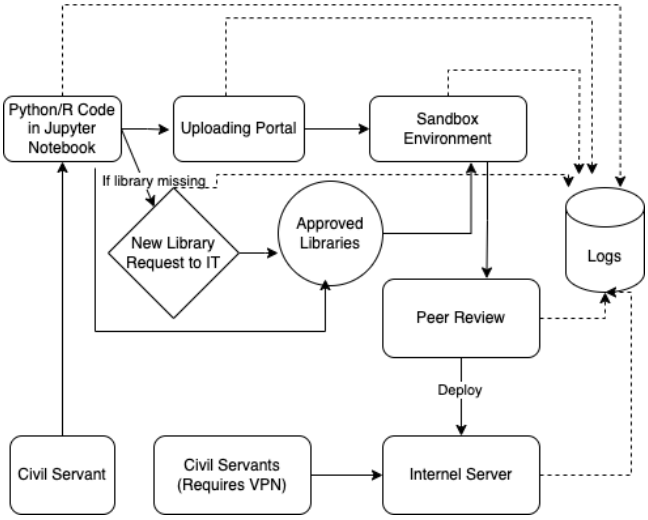


**Figure 1: System Architecture for Notebook to WebApp**

## 4.2 Upload to Portal

Once the notebook is complete, the civil servant uploads it to the platform's secure internal portal. This portal acts as the primary interface for managing application submissions, status updates, version history, and peer reviews.

Upon upload, the platform performs an automated validation process:

- `Checks Package Whitelist`: Authors must submit a dependency file (e.g., `requirements.txt` for Python) file listing all external libraries their notebook depends on. The platform cross-checks these against a curated registry of preapproved libraries. If any unapproved packages are detected, the author is prompted to submit a streamlined request for IT security review.
- `Commits Version Control`: Each notebook upload or revision is committed to an internal Git-based version control system. This ensures that all changes are tracked, historical versions are preserved, and any previous version can be restored if necessary.

By frontloading dependency checks and automating version management, the platform reduces the administrative burden on IT departments while maintaining security and traceability standards.

## 4.3 Automated Sandbox Build and Test

After successful upload, the platform automatically spins up a secure sandbox environment to validate the notebook execution:

- `Execution and Rendering`: The notebook runs end-to-end in a containerized environment to generate outputs such as plots, tables, and downloadable data files. Raw code cells are hidden from the final interface, ensuring users interact only with the intended front-end elements.
- `UI Generation`: Based on notebook metadata or YAML configurations, the system dynamically generates user interface controls. For example, a variable annotated as a dropdown choice will automatically appear as a selection menu in the resulting app. This feature transforms notebooks from static documents into parameter-driven, interactive applications.
- `Sandbox Restrictions`: The execution environment strictly limits external internet access and enforces read-only permissions for database credentials. Only connections to vetted internal APIs are permitted. These security measures minimize the risk of data leakage or unauthorized external communication, adhering to public sector IT security best practices.

If any failures occur, such as missing dependencies, runtime errors, or unauthorized operations, the platform halts the process and provides detailed feedback to the author for revision.

## 4.4 Peer Review and Approval

Following a successful sandbox build, the system initiates a lightweight peer review process:

- A designated peer reviewer, typically a colleague within the same program area, is notified.
- Reviewers are asked to validate basic functionality (e.g., does the app run correctly and produce expected outputs?) rather than performing deep technical security audits.
- Review actions—such as approval, request for changes, or rejection—are logged in the platform, creating an auditable trail for internal governance and compliance purposes.

This peer review model draws inspiration from collaborative software engineering practices, balancing quality assurance with efficiency. It supports rapid deployment while ensuring that at least two people assess each tool before it becomes widely accessible.

## 4.5 Deployment to Internal Server

Upon approval, the notebook is automatically deployed as a standalone web application on the government's internal servers. Key deployment features include:

- Applications are assigned stable internal URLs (e.g., https://apps.department.gov/internal/<app-name>), easily shareable within teams.
- Access is restricted to authenticated users connected through the intranet or VPN.
- Each application runs inside its own isolated container, ensuring that resource consumption, crashes, or security vulnerabilities in one app do not affect others.
- Applications can be scaled horizontally if usage grows, allowing for basic load balancing without requiring civil servants to manage infrastructure directly.

Because the deployed applications are fundamentally Jupyter-based backends rendered through secure web frontends, they can evolve easily over time. Civil servants can update their notebooks, submit new versions through the portal, and roll out iterative improvements without requiring full redeployments or external vendor involvement.

This deployment strategy minimizes the operational burden on central IT while enabling frontline teams to maintain, adapt, and enhance their digital tools autonomously—a critical capability for sustainability within the practical limits of government environments.

## 5 User Scenario Example

To illustrate how the platform might function, consider the following hypothetical cases:

### 5.1 Spreadsheets Generator – Binita's Tool

Binita, a Transportation Engineer at a government department, develops a Jupyter Notebook to automate the generation of Excel spreadsheets. Her tool extracts data from an internal SQL Server and an ArcGIS geodatabase, processes it, and outputs multiple structured Excel files. Previously, preparing these spreadsheets required approximately eight hours of manual work every week; her script reduces the task to just a few seconds.

With the proposed platform, Binita uploads her notebook. She specifies two configurable inputs—*month* and *county name*—using a YAML header. Her notebook relies mostly on preapproved libraries (*pandas*, *numpy*, *geopandas*) but also requires *spacy*, which is not yet on the approved list. The platform detects the use of an unapproved package and notifies Binita. Binita submits a package approval request for *spacy* through the platform. The IT security team reviews and approves the *spacy* library. The platform sandbox executes her notebook, generates the expected outputs, and configures UI widgets for the specified parameters. Binita is notified that the tool has passed automated checks and is ready for peer review.

During peer review, Yaw accesses the deployed app through a private preview link. Yaw suggests clarifying the title of one of the output charts for better interpretability. Binita updates her notebook accordingly, resubmits the new version, and passes the second review. The platform redeploys the finalized app and assigns it an internal URL.

Binita shares the link with her team. Team members, including those without coding skills, can now instantly generate customized spreadsheets for different months and counties, significantly improving efficiency.

### 5.2 Text Analysis Tool – Sirak's Tool

Sirak, a Program Data Specialist at the same government department as Binita but in a different team, develops a Jupyter Notebook that processes textual data and generates outputs based on user input. His tool utilizes the *pandas* and *spacy* libraries, both of which are already included in the agency's preapproved list. *spacy* was added after Binita's request while she was building Spreadsheet Generator tool, so it is already available to Sirak.

With the proposed platform, Sirak uploads his notebook. He specifies one parameter, *day*, as a configurable input using a YAML header. His notebook uses only preapproved libraries (*pandas*, *spacy*), so it passes the package validation automatically. The platform sandbox runs his notebook, generates an interface, and sets up a basic text input field for the parameter. Sirak is notified that the tool has passed initial checks and is ready for peer review.

During peer review, Marina reviews the deployed app via a private preview link. Marina suggests that instead of a free-text input, a dropdown list with days of the week (Monday, Tuesday, etc.) would standardize user input. She also recommends clarifying the input label to "Day of Week" to avoid confusion with specific calendar dates. Sirak updates his notebook's YAML configuration to replace the text input with a dropdown list and clarifies the label. He reuploads the revised notebook, and the platform sandbox re-executes it successfully. The app passes the second peer review without further issues. The platform deploys the updated app with an internal URL.

Sirak shares the link with his colleagues. Staff across the department can now easily select the day of the week from a dropdown menu and run the analysis consistently, without risk of input errors or needing any technical assistance.

The entire process, from code upload to internal deployment, can be completed in one to three days, rather than requiring months of procurement, contracting, or IT development time. These scenarios demonstrate how civil servants can deliver rapid, secure innovation within existing structural limits, freeing IT departments from routine application requests while maintaining public sector standards of security, accountability, and fairness.

## 6 Limitations and Future Work

### 6.1 Setup Complexity

Setting up such a system may require formal approvals, IT coordination, and, in some cases, vendor procurement. This raises a valid concern: does the platform merely shift complexity rather than reduce it? To some extent, yes—the initial setup is not trivial. However, unlike bespoke tools or one-off vendor systems, the platform is designed as a reusable template. Once piloted in a few government contexts, it gains institutional legitimacy, enabling other agencies to reference prior deployments and streamline their own approval processes. Early adopters thus pave the way for risk-averse institutions to adopt the platform with greater confidence. Future work could explore shared deployment kits, interagency collaborations, or public documentation to formalize this pattern.

### 6.2 Small Scale Tools

The platform is not intended for large-scale software development. It focuses on small, domain-specific tools, such as data cleaning scripts, report generators, or internal dashboards, that many civil servants already create informally or are capable of building but are restricted by IT security policies. By lowering barriers for these lightweight applications, the platform addresses a specific need but does not replace

the need for enterprise-grade systems or formally procured software contracts.

### 6.3 Accessibility for Non-Programmers

In its current form, the platform may exclude non-programmer innovators. Many frontline public servants have valuable ideas for improving workflows but lack coding experience. While Jupyter Notebooks reduce the barrier compared to traditional web development, they still require Python or R fluency. However, this mirrors the status quo rather than creating new obstacles. The platform may even widen participation by fostering collaboration between domain experts and technically skilled colleagues. Future versions could incorporate simplified tool creation, such as visual parameter configuration, template reuse, or assisted notebook generation, while avoiding proprietary platforms that risk vendor lock-in or rely on commercial low-code environments with uncertain long-term support.

### 6.4 Library Approval Delays

Although the platform simplifies security governance through sandboxing and package whitelisting, approving new libraries can still cause delays. IT departments, especially in sensitive policy or regulatory environments, are understandably cautious. No system can ensure complete safety, and there remains a risk of introducing vulnerabilities through third-party packages.

### 6.5 Feasibility for Small Governments

While designed for broad use across agencies, the platform may be challenging for smaller local governments with limited IT resources. Maintaining the infrastructure for sandboxed execution, peer review governance, and package registry curation demands sustained organizational capacity. Collaboration across multiple small governments could help pool resources, but further research is needed to assess such models in practice.

## 7 Discussion

Government digital transformation has followed several dominant approaches, each shaped by the institutional and technical constraints of its time. Early systems were often developed in-house by government IT departments as large, centralized projects—highly customized but expensive, inflexible, and difficult to evolve. Over time, many governments shifted toward vendor-driven procurement, outsourcing software development to external contractors. While this model promised efficiency and risk reduction, it often introduced long delays, rigid contracts, and deep dependency on outside vendors for even minor changes.

To address the limitations of these models, many governments introduced centralized digital service teams, such as the UK's Government Digital Service (GDS), the U.S. Digital Service (USDS), and similar units elsewhere. [4] These teams brought technical talent inside the public sector and demonstrated the value of agile, user-centered design. However, their capacity is often concentrated on selected high impact projects, not the small, domain specific tools needed by front line staff in everyday operations. More recently, low-code and no-code platforms have been promoted as a way to decentralize development, but they often come with risks of vendor lock-in, limited transparency, and unclear long-term support.

By contrast, in the private sector, many of these challenges are less pronounced. Teams across industries often have the flexibility to adopt freely available tools, low-cost software-as-a-service (SaaS) products, or internal scripts without the same legal, procurement, or security hurdles. For example, a marketing team might install a browser plugin, automate a spreadsheet, or connect a database to a visualization tool in just a few hours. In government, those same actions could require weeks or months of legal review, procurement paperwork, IT security vetting, or approvals from multiple departments. This stark difference highlights the need for systems that are specifically designed to work within public-sector constraints, rather than assuming the ease and flexibility that private-sector teams often take for granted.

The system proposed in this paper offers a complementary model to existing approaches, not a replacement. It does not reject centralized teams or vendor partnerships. This model recognizes a gap that neither vendors nor specialized digital units can bridge: deep, situated knowledge of frontline workflows and operational nuance. While centralized teams may have advanced engineering or data science skills, they often lack the granular, often tacit knowledge possessed by civil servants working directly on the problem. Empowering those workers to build tools does not just increase efficiency, it improves relevance, adaptability, and ownership.

By standardizing a reusable architecture, the platform lowers the barrier for risk-averse agencies to adopt it over time. Early implementations generate institutional precedent, allowing others to follow with less friction.

From a LIMITS perspective, this approach reflects the need to build within real constraints. The system uses what institutions already have: public servants with technical skills, open-source tooling, and secure internal networks. Designing for institutional resilience today, within bureaucratic and infrastructural limits, may offer a more grounded foundation for navigating future scenarios in other sectors that presently enjoys more flexibility on using low cost SaaS tools, open source packages, or internal scripts with minimal oversight. As LIMITS scholars have noted [2, 10, 11], future scenarios shaped by potential ecological instability, economic degrowth, supply chain fragility, or geopolitical disruptions could impose new limits similar to those faced by governmental organizations in other sectors. In that context, innovating within these constraints now may offer practical patterns and infrastructure that prove valuable far beyond their original use case.

## 8 Conclusion

This paper proposes a speculative but feasible platform that empowers civil servants to build and deploy internal tools safely within the structural limits of government work. By providing a secure, auditable pipeline from notebook to web application, the system upholds core public sector values of accountability, security, and fairness, while dramatically increasing agility at the frontline. Rather than bypassing procurement or IT processes, the platform complements and supports them, freeing critical resources for higher-risk projects and enabling routine innovation to flourish. It repositions civil servants not just as users of technology, but as active builders of digital solutions.

## References


[1] Barry Bozeman and Gordon Kingsley. 1998. Risk culture in public and private organizations. *Public Administration Review* 58, 2 (1998), 109–118.

[2] Jay Chen. 2016. A strategy for limits-aware computing. In *Proceedings of the Second Workshop on Computing within Limits*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/2926676.2926692

[3] Jakob Edler, Luke Georghiou, Elvira Uyarra, and Jillian Yeow. 2015. The meaning and limitations of public procurement for innovation: a supplier's experience. In *Public Procurement for Innovation*, Charles Edquist et al. (Eds.). Edward Elgar Publishing, 35–64.

[4] Ines Mergel. 2016. Agile innovation management in government: A research agenda. *Government Information Quarterly* (2016).

[5] Ines Mergel. 2017. *Digital Service Teams: Challenges and Recommendations for Government.* Technical Report. IBM Center for The Business of Government. https://www.businessofgovernment.org/sites/default/files/Digital%20Service%20Teams%20-%20Challenges%20and%20Recommendations%20for%20Government.pdf

[6] mljar. 2025. Mercury: Convert Jupyter Notebooks to Web Apps. https://github.com/mljar/mercury. Accessed: 2025-04-29.

[7] Bonnie Nardi, Bill Tomlinson, Donald J. Patterson, Jay Chen, Daniel Pargman, Barath Raghavan, and Birgit Penzenstadler. 2018. Computing within Limits. *Commun. ACM* (2018).

[8] National Institute of Standards and Technology. 2020. *Security and Privacy Controls for Information Systems and Organizations.* Technical Report NIST SP 800-53 Rev. 5. U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-53r5

[9] Committee of Public Accounts. 2025. *Use of AI in Government.* Eighteenth Report of Session 2024–25 HC 356. House of Commons, London, UK. https://committees.parliament.uk/publications/47199/documents/244683/default/

[10] Birgit Penzenstadler, Ankita Raturi, Debra J. Richardson, M. Six Silberman, and Bill Tomlinson. 2015. Collapse (and Other Futures) Software Engineering. *First Monday* (2015).

[11] Bill Tomlinson and Benoit A. Aubert. 2017. Information Systems in a Future of Decreased and Redistributed Global Growth. In *Proceedings of the 2017 Workshop on Computing Within Limits (LIMITS '17)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3080556.3080561

[12] Bill Tomlinson, M. Six Silberman, Donald J. Patterson, Yue Pan, and Eli Blevis. 2012. Collapse informatics: Augmenting the sustainability and ICT4D discourse in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM. doi:10.1145/2207676.2207770

[13] voila dashboards. 2025. Voilà: turn Jupyter notebooks into standalone web applications. https://github.com/voila-dashboards/voila. Accessed: 2025-04-29.

[14] Peter Walker. 2025. Government AI roll-outs threatened by outdated IT systems. *The Guardian* (2025). https://www.theguardian.com/technology/2025/mar/26/government-ai-roll-outs-threatened-by-outdated-it-systems Accessed: 2025-06-23.