SOFTWARE & SAAS

Guide to Evaluating or Building Public Safety Software In-House

An internal application may fit the department perfectly, but it still needs production security, support, documentation, ownership and a plan for the day its creator leaves.

13 MIN READ | AUGUST 2026

IN SHORT

A framework for comparing internally built public safety software with commercial services across cost, security, infrastructure, documentation and continuity.

Internal software can solve a department-specific problem quickly and economically. It should face the same operational scrutiny as a commercial vendor.

AI-assisted development and modern cloud platforms have made it possible for one technically capable employee or volunteer to create useful applications in days or weeks. That can be a tremendous advantage. The person building the tool may understand the department's workflow better than any outside vendor.

The risk begins when a helpful project quietly becomes operational infrastructure. More members begin using it, sensitive information is added, other systems connect to it and leadership starts depending on reports or notifications it produces. What began as a small internal tool is now a service the agency must secure, support and preserve.

The question is not whether an employee can build the application. The question is whether the agency can responsibly operate it for as long as people depend on it.

Define the Scope

Begin by defining what the system will do and what it will not do. Identify the users, workflows, data, integrations, administrators and departments affected.

Classify the application according to the consequences of failure:

- **Convenience tool:** Helpful, but work can continue manually if it is unavailable.
- **Administrative system:** Supports important records or processes, but short outages are manageable.
- **Operational system:** Affects staffing, readiness, incident information, notifications or time-sensitive decisions.
- **System of record:** Holds authoritative member, certification, incident, financial or other required information.

The higher the consequence, the stronger the security, support, recovery and continuity requirements must be.

Document who approved the project, who owns the operational outcome and who has authority to decide whether it enters production. An enthusiastic builder should not be the only person deciding that the tool is ready.

Also define the limits. If the first version is intended only to track equipment checks, do not allow it to become the unofficial personnel-record system without a new review.

Map the Workflow Before Writing Code

An internal builder may know the department well, but familiarity can hide assumptions.

Document the current workflow from beginning to end. Identify approvals, exceptions, sensitive fields, retention requirements and the people who use the information later. Talk with ordinary users, not only leadership.

Ask:

- Q Which problem is being solved?
- Q What happens today?
- Q Where does information originate?
- Q Which process depends on it next?
- Q What happens when information is missing or wrong?
- Q Which decisions require a human approval?
- Q How will success be measured?

Building the wrong workflow quickly is not an improvement.

Calculate the Complete Cost

Internal software is rarely free. The cost may be hidden inside payroll, volunteer time, existing infrastructure or work that is delayed elsewhere.

Calculate the full lifecycle cost, including:

- Initial research, design and development
- Project management and user review
- Hosting, databases, storage and network services
- Security tools and independent review

- Monitoring, logs, alerting and incident response
- Backups and recovery testing
- Software licenses and third-party APIs
- Data import and cleanup
- Documentation and administrator training
- User training and support
- Accessibility evaluation and remediation
- Updates required by operating systems, browsers and devices
- Changes to integrations and external APIs
- Compliance, legal and cyber-insurance requirements
- Continued development and technical debt
- Eventual replacement, migration or shutdown

Estimate the cost over three to five years, not only the time required to produce the first working version.

Consider opportunity cost. If an officer, IT employee or administrator spends hundreds of hours maintaining the application, what important work is not being completed?

Compare the internal cost with the complete commercial alternative, including implementation, licensing, support and migration. Neither option should receive artificially favorable accounting.

Establish Ownership

The agency should control every critical component of the system:

- Source-code repositories
- Cloud and hosting accounts

- Domains and DNS
- Databases and storage
- Email and messaging services
- Monitoring and analytics
- Security and firewall tools
- API accounts and integration credentials
- Backup and recovery services
- Application-store or distribution accounts

Use organizational accounts and agency-controlled billing. Do not build critical infrastructure around one person's personal email, credit card, GitHub account or cloud subscription.

Document intellectual-property ownership. Employment status does not always answer every ownership question, especially when volunteers, contractors, personal devices, preexisting code or open-source components are involved.

If a contractor or employee leaves, the agency should not have to negotiate for access to its own application.

Establish Development and Change Controls

A production system should not be edited directly whenever the builder has an idea.

Use a source-code repository with version history, review and protected access. Separate development, testing and production environments. Test changes before release and maintain a practical method to return to the previous working version.

Document:

- Who may change code or configuration
- Who reviews changes

- How releases are approved
- How urgent fixes are handled
- How database changes are tested
- How failed releases are reversed
- How users are notified of meaningful changes

AI-generated code requires the same review. AI can accelerate development, but it can also produce insecure, inefficient or incorrect logic that appears convincing.

Require Security Review

Security review should occur before production and continue throughout the application's life.

At minimum, evaluate:

- Multifactor authentication for administrative access
- Least-privilege roles and field-level protection for sensitive data
- Encryption during transmission and at rest
- Secure handling of passwords, tokens, keys and secrets
- Logging of administrative and sensitive actions
- Dependency and vulnerability scanning
- Patch and update procedures
- Protection against common application attacks
- Session management and account removal
- Rate limiting, firewall and denial-of-service protection
- Secure data import and export
- Incident detection, escalation and notification

Have someone other than the primary builder review the architecture and code. The reviewer should have relevant security experience and enough independence to challenge design decisions.

Determine what information the system will hold. Personnel records, dates of birth, license numbers, health information and operational data create different legal and security responsibilities.

If the agency would require a commercial vendor to complete a security questionnaire, provide insurance or demonstrate specific controls, the internal project should address the same underlying risks.

Build Reliable Infrastructure

The infrastructure should reflect the importance of the service.

Plan for:

- Availability and expected maintenance windows
- Capacity and sudden usage increases
- Database performance and integrity
- Application and infrastructure monitoring
- Automated alerts sent to more than one person
- Dependency failures
- Certificate, domain and credential expiration
- Hardware, provider or regional failures
- Disaster recovery and service restoration

Define the recovery-time objective and recovery-point objective. In plain language, determine how long the system can be unavailable and how much recent data the agency can afford to lose.

Backups should be encrypted, retained according to policy and protected from the same account compromise that could damage production. Test a complete restoration on a schedule. A successful backup notification does not prove that the agency can rebuild the service.

Avoid unnecessary complexity. A small application does not need the architecture of a national platform, but it does need infrastructure that matches its actual risk.

Document the System

Documentation must be detailed enough for another qualified person to operate, troubleshoot and recover the system.

Create and maintain:

- A plain-language purpose and scope
- Architecture and data-flow diagrams
- Inventory of services, accounts and owners
- Setup and deployment instructions
- Database structure and retention rules
- Integration details and API dependencies
- Role and permission definitions
- Backup and restoration procedures
- Monitoring and alert procedures
- Common support issues
- Release and rollback procedures
- Incident-response contacts
- Shutdown and data-export procedures

Documentation should not live only on the builder's laptop or inside the system it describes. Store protected copies where authorized leaders and technical staff can reach them during an outage.

Schedule documentation review. Instructions written during launch become unreliable as the system changes.

Plan for Personnel Change

The department must plan for ordinary and difficult personnel changes.

What happens when the builder retires, transfers, gets promoted, becomes ill, burns out or takes another job? What happens if the person is terminated and leaves angry?

More than one authorized and trained person should be able to:

- Access the source code and production environment
- Review monitoring and security alerts
- Deploy an approved change
- Restore data and service
- Rotate credentials and remove access
- Contact third-party providers
- Export the agency's information

Use individual accounts, not shared credentials, so access can be removed immediately and actions remain attributable. Critical production deletion should require additional safeguards, approval or recovery protection.

Conduct a continuity exercise. Ask the backup administrator to operate and restore the system without help from the original builder. That test will reveal whether the agency has a continuity plan or only a folder of incomplete notes.

Define Support and Service Expectations

Internal users will need support just as customers do.

Determine:

- How users report problems
- Who decides severity and priority
- Who responds after hours
- What happens during vacations or leave
- How outages are communicated
- How support requests are documented
- Which issues require leadership notification
- How enhancement requests are evaluated

If the system supports a 24-hour operation, relying on one employee who is available only during normal business hours may not be sustainable.

Separate maintenance from new development. Fixing dependencies, reviewing alerts, helping users and maintaining integrations consume time even when no new feature is being built.

Test Usability and Accessibility

Internal tools often receive less usability testing because the builder can personally explain confusing screens.

Test the application with members who were not involved in development. Include different roles, technical abilities, devices and accessibility needs.

Evaluate mobile use, keyboard navigation, contrast, readable text, form labels, error messages and screen-reader compatibility where applicable. Public employees do not lose accessibility needs because the application is internal.

Track common mistakes. Repeated user error may indicate a design problem rather than a training failure.

Address Compliance, Records and Insurance

Determine which requirements apply to the information and process involved. Potential considerations include PII, HIPAA, criminal justice information, public records, retention schedules, audit requirements, labor agreements and local IT policies.

Confirm whether the agency's cyber-insurance coverage includes internally developed applications and whether the insurer requires specific controls or

notification.

Work with legal, records, IT and security personnel before sensitive data enters the system, not after a request, audit or incident occurs.

Preserve Data Portability

The agency must be able to retrieve its information without the original builder manually reconstructing it.

Provide documented export of:

- Core records and relationships
- Attachments and uploaded documents
- Historical changes
- Audit information
- Configuration needed to interpret the data

Use common, usable formats and test the export. A raw database backup may be useful for recovery but insufficient for migration or public-record response.

Define retention and deletion, including backup copies. Determine how the system handles legal holds and records that must remain available after a member leaves.

Create a Formal Production Decision

Before the system becomes authoritative, conduct a documented go-live review.

Confirm that scope, ownership, security, testing, documentation, backups, support, training and recovery meet the approved standard. Record known risks and who accepted them.

Set a review date after launch. Internal software should not remain in production indefinitely merely because nobody scheduled a decision about its future.

Compare In-House and Commercial Options Fairly

An internal build may be the right answer when the workflow is unique, scope is controlled, capable staff are available and the agency can sustain operations.

A commercial service may provide stronger value when the need is common across many agencies, continued innovation matters or security, uptime, compliance and support exceed internal capacity.

Consider a hybrid approach. The department may use a supported commercial platform for core records while building limited integrations, dashboards or workflow extensions internally.

Evaluate both options using the same categories:

AREA	QUESTIONS TO COMPARE
Operational fit	Does it solve the real workflow and exceptions?
Total cost	What will development, licensing, support and maintenance cost over several years?
Security	Who reviews, monitors and updates the system?
Reliability	How is uptime measured, monitored and restored?
Support	Who responds when users need help or the system fails?
Continuity	What happens when key people or companies become unavailable?
Data control	Who owns the data, accounts, code and exports?
Development	Who maintains dependencies and delivers improvements?
Compliance	Can the option meet applicable requirements and audits?

AREA	QUESTIONS TO COMPARE
Exit	How does the agency migrate, replace or retire it?

Decide When to Buy Instead

A commercial service deserves strong consideration when:

- The required workflow is common and already well supported
- Sensitive information creates substantial security or compliance responsibility
- The service must operate continuously
- The department lacks multiple qualified technical administrators
- Integrations require ongoing vendor relationships
- Users expect regular improvements across devices
- Internal maintenance would depend on overtime or volunteer availability
- Leadership cannot fund the full lifecycle of internal ownership

Buying software does not eliminate risk. It transfers some responsibilities to a vendor that must be evaluated carefully. Building internally does not eliminate cost. It transfers responsibility to the agency.

The right decision is the one the department can sustain responsibly, not simply the option that reaches a demonstration first.

Frequently asked questions

Is in-house software really free?

No. Staff time, infrastructure, security, maintenance and institutional risk remain costs even without a license invoice.

Who should own the code and accounts?

The agency should control code, domains, hosting, databases and recovery through organizational accounts.

How many people need access?

At least two authorized and trained people should be able to administer and recover critical systems.

What documentation is essential?

Document architecture, deployment, dependencies, integrations, access, backups, recovery and support procedures.

When should a department choose a vendor?

Choose a vendor when the required reliability, security, compliance or development exceeds sustainable internal capacity.

Read AI Is Creating a New Generation of Public Safety Entrepreneurs. Ask the Hard Questions. and the Guide to Selecting a Public Safety SaaS Vendor for related evaluation questions.

Last reviewed August 2026

Dave Iannone

FOUNDER & CHIEF PRODUCT OFFICER, FIRST ARRIVING

He co-founded Firehouse.com and now leads First Arriving, a public safety SaaS company. A fire service veteran and former journalist, he writes on technology, marketing, recruitment and leadership in and around public safety.

Davelannone.com | LinkedIn | Contact | Schedule a Meeting

This guide is free to share. © 2026 Dave Iannone · Davelannone.com