The Acquisition that Helped the Department of Veterans Affairs Modernize Its Claims Appeals System

SUMMARY

This case study provides the following information:

- How a coding exercise was used to select a vendor
- How a digital service team met the need for additional development support services
- Lessons learned from conducting this type of acquisition
- Documents used in the solicitation

Level of Acquisition: **Expert**

The acquisition team conducting this was trained in digital service acquisition techniques and the agency's technical evaluation and implementation team is comprised of digital service experts.

By Clair Koroma and Brent Maravilla

U.S. Digital Service, Executive Office of the President

The VA competitively awarded a task order for digital services off of the agency's Indefinite Delivery Indefinite Quantity (IDIQ) contract vehicle called Transformation Twenty-One Total Technology Next Generation (T4NG). A coding exercise was leveraged as part of the evaluation.

Background

Every day, Veterans apply for health benefits from the Department of Veterans Affairs (VA). Those whose claims are denied can file an appeal to the Court of Veteran Appeals by using the VA's Enterprise Appeals Process. However, customers using the system encountered several issues that prevented them from completing the appeals process online. These issues, coupled with other systemic inefficiencies, meant that hundreds of thousands of Veterans endured waiting for long periods of time for a claim to be resolved.

To help resolve the technological inefficiencies in the claims process, the U.S. Digital Service embedded a team at the VA. Their work focused on modernizing the VA's Enterprise Appeals Process to enable efficient adjudication of appeals. On arrival, the team began to quickly develop remedies to fix the system. In order to continue the pace of work, they realized they needed more developer support. They decided to use a vendor to provide the additional developer support services to execute the project on the scale needed.

To ensure they got the right team to assist on this project, they evaluated the offerors with a coding exercise. This case study tells the story.

Lessons Learned

1. In a coding exercise, ensure you have the technical talent on the evaluation team that knows how to evaluate quality of code submission.
2. Technical team should have ability to understand and replicate an ideal submission.
3. Determine if it is necessary to have a written submission when doing a exercise. The written portion can present an additional financial burden to the company that may not be necessary if evaluating the code submission will provide sufficient insight into technical ability of company.
4. Consider use of a 2-step (down-select) process in the solicitation to decrease vendors' bid and proposal costs
5. If requiring a written portion, consider when is the best time to require it. In the case of this acquisition, vendor feedback indicated that upfront submission of the written portion was preferred since the code submission was more expensive.
6. Within the evaluation, the Government should be cautious about imposing go/no go wording. For example, stating within Code Quality a requirement that "there are no flagrant misspellings or typos" can become very binary, and lead to unwanted deficiencies.
7. Consider whether your evaluation criteria allows for sufficient strengths and weaknesses.
8. Prior to the solicitation, ensure that your team clearly understands potential deficiencies. For example, must the vendor complete all user stories or are there specific stories that are key? During the evaluation phase, the team realized that ***theoretically*** a vendor could have made an excellent submission even if they had not completed all user stories."

9. Much of the coding evaluation made for a very objective analysis (e.g. Specific security vulnerability identified), which is helpful to both parties for debriefing purposes.

The Problem

The Digital Service team at the VA had two developers on arrival. Those two developers began work on modernizing the VA's Enterprise Appeals Claims system. On delivery, their work would help alleviate the backlog in claims appeals that prevented veterans from getting benefits. As work progressed, it became clear that in order to continue the rapid pace of development, additional developer support was required. Using a vendor to maintain the dev pace was the fastest way to get high-quality support on the project. The vendor that could best meet the needs of the product would have developers with expertise in agile development, user-centered design, User Experience(UX) research, modern technology stacks, and DevOps. The question was how would they find this vendor? More importantly, how could they verify developer acumen in the aforementioned areas **before** contract award? Simple, they needed an innovative evaluation approach.

Finding A Solution

A hybrid VA-Digital Service team, including Contracting Officer Mark Junda, Contract Specialist Brandon Caltabilota (both graduates of USDS' Digital IT Acquisitions Professional(DITAP) Contracting Officer training), Digital Service Expert (design), Kavi Harshawat, and Digital Service Expert (engineering), Shane Russell worked together to devise an innovative evaluation for submissions and unique acquisition strategy that were uncommon in their approach. Leveraging the use an IDIQ vehicle for the VA, the team hypothesized that they could use a code exercise to reach the right vendors. Although unconventional, a code Wil offered the best opportunity for a vendor to demonstrate how they would build a modern digital tool. It also presented the VA team with substantive code upon which the vendor could be evaluated. In theory this would help the VA to successfully identify a vendor who could help continue the pace and scale of the modernization work on the appeals system.

A Unique Contracting Approach

Separate contractual requirements from functional requirements

The VA anticipated that functional requirements will change, which is normal, and therefore separated contractual requirements from functional requirements. Since their contractual requirements indicate a repeatable process that results in working software code, this allows functional requirements to change as user needs change. The functional requirements can change without a modification because the changes to the features are within the general scope of the contract. For example, instead of adding contractual requirements that dictated the specifics of a user story like "the screen shall be blue" the requirement was "backlog of user stories shall be completed by the vendor." This enabled the VA to reference the user stories as a requirement and allow for changes to the backlog as needed without contract modification.

Market intelligence

VA heavily leverages its T4NG IDIQ for software development services.

Next the team had to develop a solicitation for the support services that evaluated vendors based on a coding exercise. Why a coding exercise? This is a common method used in commercial companies to help them selecting the right talent and vendors. It is also a common practice used by established software firms during their hiring processes. Therefore, it made sense to use this method to select the right vendor to provide digital services to the government.

> Show Don't Tell
>
> Imagine it's time for show and tell. Two kids show toys that day. The first, showed the class how to build a toy while continuously asking classmates what they wanted the toy to do. She modified it based on their feedback. She even had different classmates try the toy as she was building it and made changes for better functionality. The second, built an amazing toy step by step while the class watched. He never gathered feedback from the class during the build. After several kids tried each toy, most preferred the custom toy. Conversely, the toy built without any mods or changes based on class feedback, was seldom used because many of the kids found that it was difficult to maneuver use its features.

Sadly, the latter experience in this story is very similar to what normally happens when the Government buys IT. A company gives a shiny description of how they will deliver a digital service or tool, but absent a demo of their skillset to deliver it, or incorporation of user feedback as they work on the end product, we often end up with things that do not work for users.

In order to avoid that experience in their procurement, the VA team worked with the Technology Acquisition Center (TAC) to design a RFQ that included a coding exercise. While bidders had to submit a traditional written response that included management approach, technical approach, etc., they also had to demonstrate how they would get the work done.

The following is how the acquisition was executed:

1. The VA Digital Service team designed the coding exercise parameters they wanted tested and then validated that the exercise could actually be run in the environment and timeframe they wanted to include in the solicitation. They used digital service team members from other teams as well as ran the tests themselves to validate.
2. The Request for Task Execution Plan (RTEP) detailing the full requirement was released as a Service-Disabled Veteran-Owned Small Business set aside on the T4NG. It included the coding exercise submission instructions and evaluation approach. At this point the vendors could begin working on the response to the price and the written technical portion. The RTEP also provided guidance on the coding submission and detailed how the submission would be evaluated.
3. After release of the RTEP, VA provided notice that the coding submission information would be posted at a certain date/time.

4. At the appointed time, nine user stories (*See Appendix 1*) were released on the T4NG repository where RTEPs are posted and proposals are submitted. Upon release, vendors had 4 hours to submit questions and the VA had 2 hours to respond to all questions.
5. Vendors had 72 hours, from time of release, to provide a code submission.
6. A detailed evaluation criteria (*See Appendix 2*) was developed to guide offerors on the requirements for successful completion of the exercise
7. The exercise required the offerors to build an online payment system (venmo clone). The clone required extensive security considerations, so there were many opportunities for excellence OR mistakes!
8. The source code for the coding submission and all relevant design assets and documentation were submitted via github repository with a clearly viewable commit history of the entire development process.
9. Offerors formally certified that "the team who developed and designed the coding submission are proposed as key personnel to perform work under the resulting task order barring any unforeseen circumstances"

Evaluate the Right Things

The typical evaluation criteria wasn't going to work. The team consulted with GSA's 18F team about their experiences running a coding exercise to select vendors for their agile Blanket Purchase Agreement (BPA). Some key suggestions from 18F included using automated testing to quickly show weaknesses or strengths in a vendors' code base AND specifying the coding language for the submissions. The team then developed criteria that properly evaluated the quality of the submitted code and design.

The team used a combination of:

1. Evaluation criteria in the RFQ that was written to allow for innovative approaches to the exercise, and
2. Specific validation that enabled the team to evaluate submissions uniformly against the criteria.

An example of how this approach was used to evaluate submissions is in the following table:

Criteria	Validation
Tests fail when functionality is broken	"(I'd) go into every code base, break something, and then run the (vendor's) test and make sure the test failed. And I did that with 5 separate things in every (vendor's) code base" – Shane Russell, USDS
Accounts for high volumes and heavy loads	"(we wrote) a script to load 10,000 transactions into each vendor's (application?), then load (the vendor's) webpage. And if it took more than 20 seconds (to load) then they failed and it was a weakness" – Shane Russell, USDS

This combined evaluation approach was only possible because of the visibility that a coding exercise provides. Ultimately, that approach proved to be effective in identifying the best vendor. Submissions were evaluated against seven development categories and four design categories, each with subcomponents (*See Appendix 2: Evaluation Criteria)*.

Results

The team received six submissions that each took 4-6 hours to evaluate. The rubric and evaluation criteria allowed the technical team to evaluate and select the vendor offering the best value and expertise for the project requirements. This submission style also gave way to well-informed debriefs that likely contributed to zero protest actions on the award. Ability to review code prior to contract award allowed:

1. Test of the submitted code for errors that would be fatal to project success
2. Determination of best code based on hard evidence of work completed in direct response to the RFQ, not just a narrative of how the vendor would accomplish requested tasks.
3. Ability to identify the vendor with the developers that met the VA needs

Task Order Award Details

The period of performance is for a base plus 2 option years for a total awarded amount of $13,946,873.

Contract Type: This order was awarded as a hybrid Time & Materials (T&M) and Firm Fixed Price (FFP) task order. The development support services are primarily T&M since they meet the main need: providing dev support for the blended government/contractor development and design team implementing the new Appeals system.

Solicitation Time Frame: From identification of need to award took 7 months. This included the entirety of creating and revising all acquisition documents, market research, selection of most suitable contract vehicle, solicitation, evaluation, and award. The evaluation timeframe took about 1.5 months, but some of this was impacted by coordinating team schedules and the newness of writing evaluations based on evaluating software code.

Milestones in Modernizing the Appeals System Since Contract Award

1. Product, Caseflow Dispatch moved from idea (before requirements gathering) to implementation in 5 months. This product ensures that all decisions made at the Board are uploaded to the proper VA systems, and that the claims are properly adjusted. This will close a gap through which thousands of appeals could get lost. The product also standardizes and correctly routes decisions of over 20 categories to over 70 VA offices around the country that use differing mechanisms for managing their work.
2. Prototyped a document reader that can greatly speed attorney workflows at the board, which would have a direct effect on the speed at which the backlog appeals are processed.
3. Automating Form 8, which is a part of the process for transferring appeals from regional offices to the Board of Veterans' Appeals (BVA). This streamlines the process and speed up the rate at which appeals can be transferred to the BVA.

Conclusion

Incorporating a coding exercise into a digital services acquisitions better positions a team to select the right vendor to meet their needs. It allows teams the opportunity to work with a potential vendor in real time to learn about their capacity to deliver the quality of digital services needed. Additionally, it allows vendors an opportunity to learn the maximum amount of details to best respond for an RFQ.

Appendix 1. User Stories

Cashflow is a desktop web application where people can send money to each other, along with a message. It allows for users that use different currencies.

Login

- As a user,
- When I visit the login page,
- Then there will be a form where I can enter my username and password.
- When I enter a valid password for a given username, And submit the form, Then I should be redirected to the account page, logged in.
- Additional Information:
- There should be a database of users that are capable of being authenticated with passwords.
- Users should be creatable by a database administrator.

Failed Login

- As a user,
- When I visit the login page,
- And I enter an unknown username, or a password that does not match the username I entered, Then I should see an error, stating that my username and password did not match, and I should not be logged in.

Account Page - Not Logged In

- As a user,
- When I attempt to visit the account page, And I'm not logged in, Then I should be redirected to the login page.

Account Page - Show basic Information As a user, When I'm logged in and visit the account page, I should see

- My username
- My account balance - the total amount of money I have, listed in my home currency.
- Additional info:
- There are three currencies that can be used to store money: Purple Dollars, Gold Dollars, and Silver Dollars.
- Each user should have a home currency, which is the currency that their balance is stored in.

Bank Transactions

- As a database administrator,
- I should be able to create a transaction to give any user any amount of money
- in any currency from the "International World Bank". The bank has an unlimited amount of money.
- This can be done in the database or via a script, and does not require a web UI.

Account Page - Create Transaction

- As a user,
- When I'm logged in and visit the account page,
- I should see a transaction form with a "Recipient Username" field,
- an "Amount" field, and a "Message" field.
- When I enter the username of a user who has the same home currency as I do,
- And I enter a numeric value into the amount field,
- And I optionally enter a string into the message field,
- And I submit the transaction form,
- Then a transaction will be created for the specified amount from me to the user matching the username field. The entered message should also be associated to the transaction.
- And my account balance should be immediately updated to reflect the lost money that I sent, without reloading the entire page.
- And the next time the recipient visits their account page, their account balance should reflect the money that they received.

Account Page - Show Transactions

- As a user,
- When I'm logged in and I visit the account page,
- I should see a list of all transactions where I have given or received money, along with the usernames, and messages for those messages.
- And transactions should be listed in reverse chronological order, with the most recent transaction on top.
- And the amount listed for transactions where I gave money should be red, and be a negative number. (i.e. -20)
- And the amount listed for transactions where I received money should be green, and have a positive value.
- All amounts should be shown in my home currency.
- When I successfully submit a transaction,
- The transaction list should update without reloading the entire page.
- Additional Info: Transaction amounts should be truncated at 2 decimal places.

Account Page - Validate transaction

- As a user,

- When I'm logged in and I visit the account page,
- When I try and submit a transaction with a username that does not match any user in the database,
- Then I should see an error, telling me that the user was not found.
- And a transaction is not created.
- When I try and submit a transaction with an amount greater than my account balance,
- Then I should see an error, telling me that I do not have enough money to complete that transaction.
- And a transaction is not created.
- When I try and submit a transaction with an amount with no username or amount filled in,
- Then I should see an error, telling me that I need to fill in the required fields.
- And a transaction is not created.
- **Additional Info:** Unsuccessful transactions should not update the displayed account balance.

Account Page - Create Transaction Requiring Exchange

- As a user,
- When I successfully submit the transaction form from the account page,
- And the recipient has a different home currency than me,
- Then two transactions should be created,
- 1) From me to the bank in my home currency
- 2) From the bank to the recipient in the recipient's home currency
- The amount of 2) should be based on an exchange rate from my home currency to the recipient's home currency (see additional information below).
- And I should see just one transaction from myself to the recipient in the transaction list. The amount should show up in my home currency.
- As a recipient of the transaction,
- I should see one transaction in my transaction list, with the amount displayed in my own currency.
- **Additional information:**
    - The exchange rates are as follows:
    - 7.25 Silver Dollars = 1 Purple Dollar
    - 2 Gold Dollars = 1 Purple Dollar
- The bank, as a nonprofit, will exchange money without taking a percentage of the transaction.
- Transaction amounts should continue to be truncated at 2 decimal places. The system should round values such that the bank does not lose fractions of money on transactions.
- **Security Concerns**

- Users should not be able to steal money from the bank or each other.
- All transaction data is considered private between the receiver and the sender.
- Users should not have access to transactions where they are neither the sender
- or receiver.

Appendix 2. Coding Exercise Evaluation Criteria

Traditional Requirements

A detailed description and diagram of the Offeror's approach to architecting and implementing a secure, scalable, automated, fault-tolerant system that supports the ongoing integration of developed appeals-centric features and products in accordance with PWS paragraph 5.2.1.

A description of the Offeror's approach to applying the design and development principles that will be used for the coding submission across the life of the task order to meet the continuous deployment requirements defined in PWS paragraph 5.2.2.2 for the products defined in PWS paragraph 5.2.1.

The Offeror's detailed solution of the planned management methodology for executing the effort, including a proposed staffing approach, key personnel retention, security clearance considerations, and details of how the proposed effort will be assigned within the Offeror's corporate entity and among proposed subcontractors.

Innovative Requirements

Coding Submission: The Offeror shall ***develop*** an application adhering to the requirements defined in Section B(3)(a)(3) Coding Submission and, in addition, shall be evaluated by the Government in seven categories as follows:

Security

- No blatantly exploitable vulnerability or failed automated security checks
- No other vulnerabilities discovered
- Threat mitigation techniques implemented

Testing

- Automated tests written
- All tests pass
- Not missing an entire category of tests (feature or unit)
- Code coverage meets standards set by SimpleCov
- There are JavaScript tests, if appropriate
- The tests are easy to read/understand
- The tests are well organized
- There is no duplicated/unnecessary testing
- Tests fail when functionality is broken

Database/Data Modeling

- Data is organized into logical objects
- Data is not stored in multiple places, without reason
- Indexes are present for primary operations so that tables can handle 100,000 or more rows
- Data is pragmatically normalized
- Data is easy to query/understand

- Naming is clear with consistent conventions
- Correct data types are used
- Proper constraints exist
- Seed data exists, when applicable

DevOps

- Deployed solution
- Deployed solution does not require repeated manual configuration
- Deployment automation is testable without a host machine (e.g., using Vagrant, etc.)
- Development Environment setup is repeatable and not error prone.
- Deployment automation is well written and organized

Code Quality

- There is no duplication of logic
- Controllers split based on restful resources
- JavaScript is encapsulated
- JavaScript is written in its own file, in-markup JavaScript is limited
- There are no flagrant misspellings or typos
- Makes use of SASS variables
- Uses bourbon to reduce code involved in advanced CSS, if relevant
- Classes and methods are terse and do one thing
- The code is easy to understand (comments can help)
- The code is easy to change and maintain.
- Commit messages are clear. Why changes are made is explained, if relevant.
- Uses rails features and conventions when relevant
- Keeps business logic outside of controllers, in models and service objects
- There is limited logic in ERB templates, formatting logic is delegated to helpers
- Avoids creating unnecessary boilerplate files and code
- Passes all automated linters

Application Quality

- Edge cases are properly handled
- Uses restful routes with short, readable URLs
- The root goes to account page
- Accounts for high volumes and heavy loads
- All interactive elements are in a sensible tab order (for accessibility)
- Page is accessible and 508 compliant
- Incorporates CSS animations
- Follows visual design documents
- Appropriate HTTP response codes are used
- Does not generate HTTP 500 status

Documentation

- Assumptions are documented
- Includes development environment setup documentation
- Technical decision making and architecture documented, including a diagram of major infrastructure components.

Coding Submission: The Offeror shall ***design*** an application adhering to the requirements defined in Section B(3)(a)(3) Coding Submission and, in addition, shall be evaluated by the Government in four categories as follows:

Visual Design

- Color palette and contrast
- Consistency in fonts, colors, weights, spacing
- Hierarchy of information
- Creation, delivery, and quality of visual assets such as icons or images
- Creation of a visual system, style guide, or re-usable visual patterns
- Minimalism
- Use of USWDS visual style (typography, color palette, grid relevant interface components)
- Extension of USWDS where appropriate

User Research

- Spoke with one or more users
- Prepared and conducted formal usability studies
- Documentation of findings
- Presentation of findings
- Used feedback to improve the application
- Choice and appropriateness of research methods

Interaction Design

- Designed using wireframes, sketches or high fidelity mockups
- Demonstration of design iteration
- Clear errors messaging and follow up actions
- Affordance and discoverability
- Progressive disclosure
- Indication of system status
- Mapping of interaction inside and between pages
- Consideration of alternate design directions
- Organization of information
- Fidelity of design assets delivered to developers
- Use of common interaction design patterns

Thoroughness

- Includes interaction, functionality, and design not specified in the coding submission instructions
- Handling of edge cases
- Page titles and section headings
- Pagination
- Motion design and animation
- Design coherence
- Error prevention
- Contextual help and documentation
- Ease of implementation