JPL Summer Internship Final Report

James [redacted] †, Cristopher [redacted] *
† *2024 Summer Intern*
Jet Propulsion Laboratory, California Institute of Technology
Bunker Hill Community College - Charlestown, MA
james[redacted]@jpl.nasa.gov
* *Software Quality Assurance Engineer*
Jet Propulsion Laboratory, California Institute of Technology
cristophe[redacted]@jpl.nasa.gov

***Abstract*—Cyber Security Assurance (CSA) assessors are tasked with reviewing cyber security for flight missions and ensuring that the processes and systems implemented are following all guidelines that have been defined for it to follow. While these guidelines and requirements have been well-defined within the NIST SP800-53 and NIST SP800-37 documents, there is still much to be desired with the process of conducting these assessments. Today, when an CSA assessor conducts an assessment, the primary tool they use to complete their work is Microsoft Excel. This document presents CyberCAT, a web-based tool based on Cristopher [redacted]'s original concept and idea with the goal of helping Cyber Security Assurance Assessors conduct and complete assessments.**

I. Introduction

CyberCAT is a web-based assessment tool that can be used to help cyber security assurance assessors conduct and complete assessments

During my internship, I had the opportunity to apply the skills I have learned in web development to build an enterprise web application that will allow Cyber Security Assurance assessors to conduct and complete their assessments with a tool that complements the institutional tool, specifically the Risk Information Security Compliance System. This application aims to provide assessors with a streamlined experience in conducting and tracking the progress of assessments, enabling them to simultaneously work on the same assessment and stay up-to-date on progress. Before I joined this project, it was decided that the application would be built with Django, Django Rest Framework (DRF), Vue.js, and Vuetify. My mentor, Cristopher [redacted], also brought on User Interface/User Experience designer Steven [redacted], who has a wealth of experience in designing user interfaces to ensure they are easy to understand and use, regardless of whether the user has conducted several assessments in the past or if this is the first assessment they would ever perform. Cristopher has also been instrumental in understanding the tools and libraries used to build this application. While I had significant experience with Python, the language used to write Django and Django Rest Framework, I had yet to work with these specific frameworks. However, I quickly learned they offer a single solution that can help acknowledge fully implement a web-based application programming interface (API).

When initially designing this application, five major goals were decided on to be a priority, all of which are features which would be required at a for the application to be usable by stakeholders.

II. Background

With respect to flight projects, CSA conducts assessments to ensure that any security controls in scope are implemented appropriately and well understood. NASA has chosen to follow the NIST framework when conducting its assessments. The framework is followed through NASA's Assessment and Authorization (AA), which is defined by NIST 800-53. This document defines a yearly process where a system is continually assessed to ensure its reliability and compliance. The assessments are conducted yearly, with the first year

having an external assessor and the following two years with an internal assessor. This repeats. Every third year, an external assessor must come in to ensure compliance. The method used to evaluate and assess compliance is through evaluating controls. Within cybersecurity projects, each control defines a particular requirement for how an aspect of an application must be managed. For example, one control might be ”Unsuccessful Logon Attempts,” which defines how a system should respond when a user attempts and fails to log in. NIST SP800-53 also makes accommodations for different organizational preferences through ”organizational defined values” (ODVs). ODVs allow for each organization following the NIST SP800 framework to respond differently to this control. Using variables means that, for example, one organization can only allow ten login attempts, whereas others may only want to allow two attempts before being locked out. Within NASA, this A&A process must be adhered to if a project would like to receive an Authority to Operate (ATO). An ATO is a statement stating that the assessment process has been followed, that the risks of using the assessed project have been accepted, and that the project is able to be operated. Only authorizing Officers (sent to a project by NASA) have the authority to issue ATOs.

III. Frameworks

A. Vue.js

Web application development is typically split into two separate programs: the front end and the back end. The front end is the part of the program with which the user interacts, and for this application, Vue.js and Vuetify were chosen. Vue.js is a JavaScript framework that helps build dynamic and interactive user interfaces for web applications and typically interfaces with the backend. Generally, no pertinent data is stored on the front end. It acts as a gateway to the backend and is the primary way users interact with stored data. As web development has evolved, users generally want more responsive websites that are performant, dynamic, visually appealing, and provide intuitive, user-friendly interfaces. Vue.js solves the question of performance and keeping websites dynamic so new information can be presented in an existing window without reloading the screen. It starts by defining ”views,” a single file containing HTML, cascading Style Sheet (CSS), and JavaScript. HTML defines how code is laid out on the screen, CSS defines what the HTML will look like, and JavaScript defines how different HTML elements will function when interacting with them, such as by clicking or putting your mouse over them. These views are generally composed of other views. If we imagine Google, the search bar and search button can be individual components, composed together in a parent view that keeps everything together.

B. Vue Components

These ”components” are isolated files containing all necessary code for a single visual element. For example, within CyberCAT, there is a page that lists all existing assessments. The content shown in their visual elements is generally the same among all these assessments. We present the name of the assessment, who the primary assessor is, and when the assessment must be completed. While we could include the code required for this visual element several times programmatically, based on how many assessments we want to show the user (which could be different every time a user accesses the page), it would be more organized to write this code a single time inside a component. This component would define all variables needed to present this visual element to the user. This development form significantly simplifies a codebase, especially when a website grows more extensive than a few web pages.

1) Vue-Router: The architecture of Vue.js is that of a single-page application (SPA). Typically, when a user navigates a website, clicking on links and interacting with the backend, their web browser handles things like tracking history. They can use the back button to go to the last page they visited and communicate with the server so that new information can be retrieved and displayed to the user. With Vue.js, this model is changed significantly. When the user connects to a website using this SPA architecture, that first attempt to connect is the only attempt the web browser makes to connect to the website. From here, Vue.js takes control of changing the website’s state and ensuring that the web browser still behaves as the user expects. SPA applications do not have different HTML for the

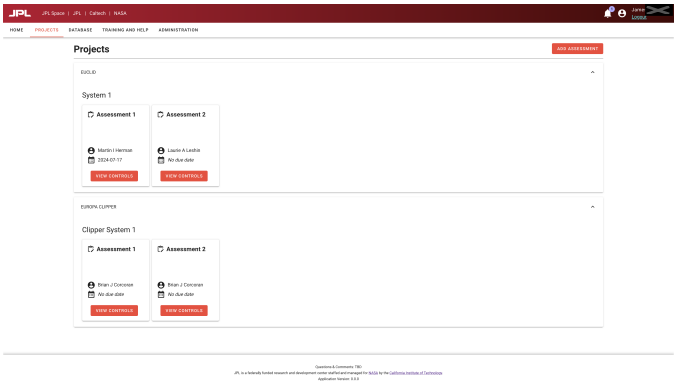

Fig. 1. CyberCAT Project/Active Assessment View

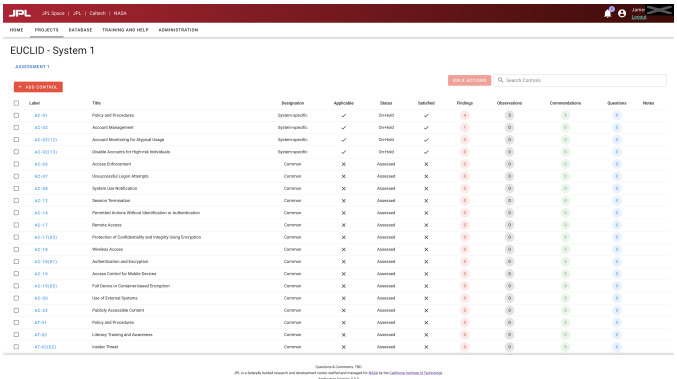

Fig. 2. CyberCAT Control Table View

web browser to request as the user navigates the web-page. They are all defined in the previously mentioned ”view” files, though it is still essential for the ”back” and ”forward” buttons to function as the user expects. This means that the back and forward buttons need to be re-implemented to allow an SPA to function correctly without losing the data on the page. Vue-router is a library that helps accomplish this desired behavior within a SPA.

2) Vuetify: With a framework that simplifies creating dynamic and generated content, the next question is how to create an interface that people enjoy using and that is pleasing to the eye. The roles of the person who can build a website and the person who can design a website are two entirely different disciplines, similar to how an engineer who can construct a bridge is different from the architect who can design the bridge. This library, along with the help from Steven [redacted] (319H), satisfies this application's 'architecture' role. Before I joined, Steven had been working with Cristopher to help develop an interface catered to quality assurance engineers who conduct assessments and had laid out several critical pages of this application necessary for building what is considered a minimum viable product (MVP). With these interfaces designed, I could take advantage of Steven's layout and the pre-build components from the Vuetify library to add functionality to a pre-made. For instance, Vuetify's grid system is used throughout the SPA to match the layout of visual elements as they have been designed. With Vuetify being named eponymously to Vue.js, its implementation is directed explicitly towards using Vue.js, meaning it takes advantage of the Vue.js core features that make it easy to implement interactive user interfaces. Two notable features include:

- Reactive Bindings: When ‘v-model‘ is applied to almost any Vuetify component, it behaves like any Vue.js component and lets the developer access the pertinent data within the component effortlessly. ‘v-model‘ is a built-in attribute that exists when the ‘defineModel‘ function is called from within the component. It is a two-way binding that allows data modified inside the component to be easily accessed from outside.
- Vue Router: Components within Vuetify intended to link to other pages are often given a ‘to‘ attribute, specifically designed to interact with Vue Router and link to pages defined within Vue Router.

C. Django

With the features provided by the libraries noted here, we are given more than enough features to develop a fully-features frontend application that can satisfy the needs of this enterprise web application. However, it is only half of what is required to complete this application. Without the backend, there would be no data to present in the application's front end. Django is the framework that would be used to develop the backend of this application. While this framework was chosen before I arrived at JPL, there are several aspects to the framework that make it well-suited to what we're trying to develop. Those aspects include:

- Admin Interface: Django adds an admin interface that makes it easy for any user, even those not intimately familiar with web development, to make changes to the application that can be seen on the front end. This interface is valuable as development is focused on features that will make CyberCAT usable from the start. An application can still be usable without an admin interface. With this interface, we can focus on application-critical features without being distracted by the management side of the application.
- Object-Relational Mapping (ORM): CyberCAT will store its application data within a PostgreSQL database. When interacting with this database, programmers must manually type out Structure Query Language (SQL) commands to create, read, update, and delete data. While the language is easy to learn and understand, it can quickly become exceedingly difficult to understand when large queries need to be created. Django's ORM helps keep the developer from writing raw SQL by providing an equivalent interface that is accessible through a familiar Python object interface.
- Authentication System: On the backend, Django has a complete authentication system that exists within Django, which includes a login system, password management, and user permissions. This feature generally has to be written into every web application, and having that already integrated into Django, along with extensions that allow it to integrate with the JPL LDAP, greatly simplifies adding and implementing this feature. While it's still a feature that needs to be integrated with the front end, a large portion of the backend functionality for this system is already complete for us.

These core features simplify essential backend functionality but do not offer the front end a standardized way to communicate with the implemented data, which is the responsibility of a REST API that will be used for this application.

IV. Django Rest Framework

While Django can render HTML directly to the client, using that functionality when using a frontend application like frontend Vue.js is not ideal, as it already handles all visual aspects. All we need from the front end is for the data to be serialized in a format that the front end can process. Django Rest Framework (DRF) is built to act as this interface. The primary features that DRF provide that allow for easily building API's include:

- ViewSets: When the user is querying to a API endpoint, that usually is done through a ViewSet, most commonly in the from of a ModelViewSet. This type of ViewSet provide create, read, update, and delete (CRUD) functions for the associated model. For CyberCAT, there are many instances where the user will want to be able to do a CRUD operation on a single model, such as a comment, or updating an evaluation. ModelViewsSets make this easy to implement.
- Serializers: In order for data to be sent to the user, it needs to be queried from the database and transformed in to a format that can be read and understood by the client. In this case, since we're interfacing with a web-application, the go-to choice is usually JavaScript Object Notation (JSON) serialization, as JavaScript can easily take advantage of an object formatted in JSON.
- Validators: When the user is submitting data to be stored in the database, it's a common use-case where the backend will want to validate that the data passed in by the user is acceptable data. "Acceptable" can be considered a value is not a number, when it's intended to be a boolean value, or ensuring that a given value matches a value that is stored in the database as a foreign-key. The goal of the validator is to ensure that the data follows the model correctly, or raises an exception before it reaches the point where the data gets stored in the database.
- Permissions: At the moment CyberCAT's only permission model is that of either being allowed to access the tool, or not being allowed to access the tool. Though DRF implements a system where each endpoint can have a different required set of permissions, for who is allowed to do what actions.

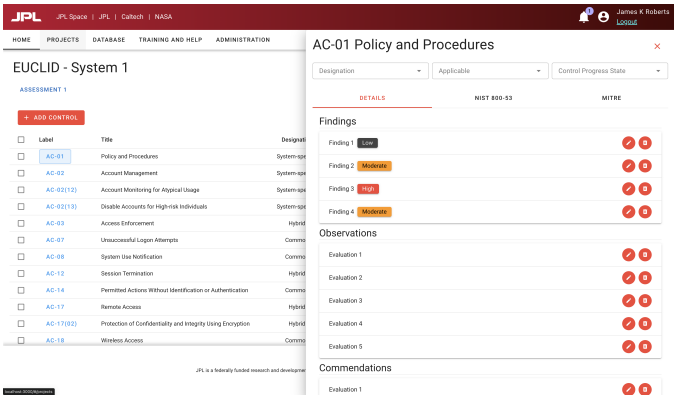

Fig. 3. CyberCAT Control Details View

V. CyberCAT Development

CyberCAT has received most it's development in the following areas deemed necessary for a minimum viable version of this application.

A. User Authentication

Users are able to login with their JPL credentials, and access assessments which they are given permission to access. This system interfaces with a Lightweight Directory Access Protocol server which JPL hosts that stores all users login information. Once authenticated, that users pertinent information (name, badge number, section and division number, and their supervisor), are all pulled into the database used to store user information. From here, when the user attempts to login, this information can be sent to the frontend, letting the application keep track of the current user who is accessing the database.

B. Control Table and Control Details

These two parts of the application compose where the assessor will be spending the majority of their time conducting their assessments. It provides a view where they can search through existing controls, add new controls which were not originally apart of the assessment, and a drawer menu that appears when a control is selected to modify the details of each individual control. In the control details view, there are three kind of evaluations that can be made within a control.

- Findings: These are concerns that are seen when an assessor is evaluating the control at hand. There is usually a risk level associated with a finding (low, moderate, high) and serve as documentation for those who implement the controls to update/resolve for future upgrades.
- Observations: The serve as records that have no inherit positive or negative bearing, and act as a note which the assessor deemed noteworthy for recording. These can be associated with the assessment overall as-well, and will usually be associated with a control when it has a direct relationship with it.
- Commendations: These are similar to observations, but generally with a positive bearing, and serves as a way for an assessor to say an implementation was done well. These can be associated with assessments in the same way which observations are.

C. Bulk Modification

Inside this table, assessors can also complete and modify a large number of assessments all at once, as assessors frequently find that many assessments can be related to each other (such as assessments all with the same label but different enhancements on top of them). They would find it helpful to mark multiple assessments as complete, pending, or on hold (if their completion is being blocked). This feature is designed to simplify the assessment process by allowing assessors to make changes to multiple controls simultaneously.

Glossary

A&A Assessment and Authorization. 2
ATO Authority to Operate. 2
CSA Cyber Security Assurance. 1
NIST National Institute of Standards and Technology. 1, 2
ODV Organization Defined Values. 2

Acknowledgements

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by Maximizing Student Potential and the National Aeronautics and Space Administration (80NM0018D0004).