USSF GRC Controls

Project Summary

The **USSF GRC Controls** project is a SaaS application built using Ruby on Rails that helps USSF representatives in the cybersecurity space ensure their applications comply with Governance, Risk, and Compliance (GRC) controls as defined by NIST SP800-53. The application scans Docker images—vital components in cloud-native, microservices-based architectures—for vulnerabilities, providing detailed reports on whether these images meet the relevant NIST controls. With the growing adoption of scalable, containerized environments, Docker images play a critical role in maintaining consistent, isolated environments across development, testing, and production. By focusing on Docker images as the runtime environment objects, the application addresses a specific need in the cybersecurity industry to assess and maintain compliance with GRC controls in a streamlined, efficient manner.

The main customer need is to help USSF representatives quickly assess the security of their Docker images and ensure that they comply with the required GRC controls. Our application meets this need by offering a user-friendly interface that scans both public and private Docker images, identifying vulnerabilities and mapping them to the corresponding NIST controls. In addition, the application supports version control for Docker image reports, allowing users to track changes over time using tags, and it includes robust access control features to manage different user types and control permissions for viewing reports. The primary stakeholders are USSF representatives responsible for deploying applications in secure, compliance-driven environments. By providing an easy-to-use platform to evaluate and maintain GRC compliance, our application empowers these stakeholders to ensure the security and integrity of their Docker images before deployment.

Sprint Details

Sprint 1

Team Roles:

- **Product Owner:** Shravan Bhat
- **Scrum Master:** Aditya Gourishetty
- **Developers:** Duy Vu, Maitreya Niranjan, Medha Kaushika Podipireddi, Sahil Fayaz, Vasudha Devarakond

Summary: We were successfully able to achieve the sprint goal. We have a working prototype of the GRC scan application deployed on Heroku and the login feature through SSO enabled for the application users. We have also finalized using Trivy - an open-source security scanner as our backend tool to run the scan against the provided images. On the deployed application, currently, we have a home page, new image page, main image page, and report page, and can pull any public image and run the GRC scan for it. We have also achieved good code quality having used rubocop, rubycritic, and code coverage for rspec and cucumber test cases. All the committed stories for the sprint have been marked as done.

Total Points Completed: 15

User Stories

Feature: Login to the application

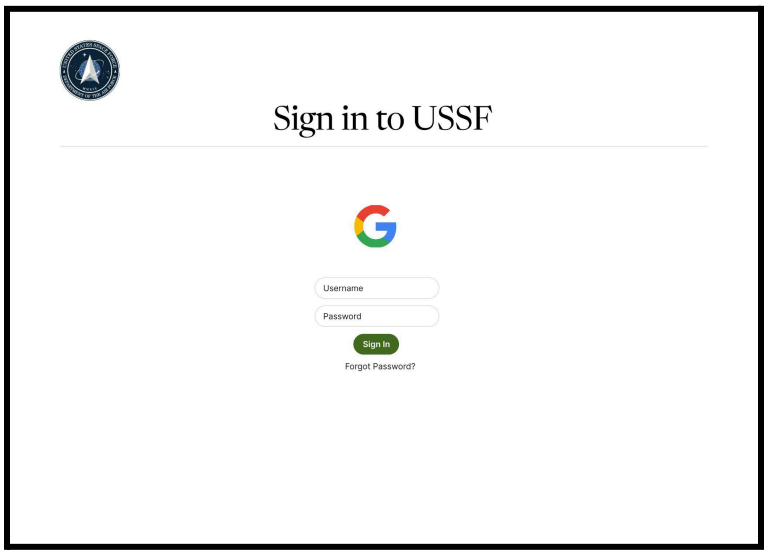
- **As a user** of the application
- **So that** I can log in to the application
- **I want** a login page with a login button through which I can log in via SSO.

Sub Tasks:

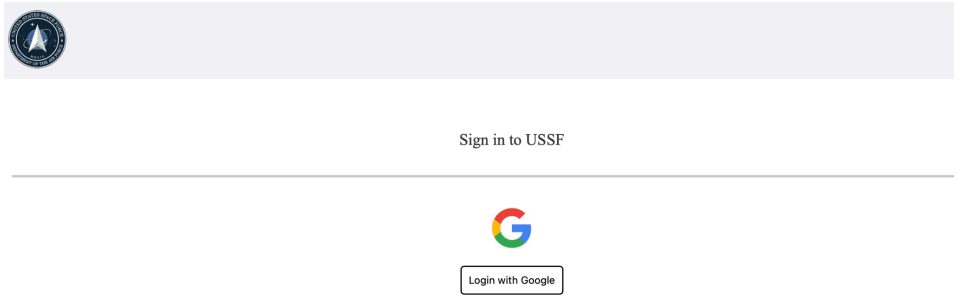
- Integrate Google OAuth SSO Provider by creating a new project in Google Developer Console and Implement SSO Login & Session Management.
- Test and Validate SSO Functionality

Implementation Status: Completed
Changes to Story during the Sprint: None

UI Mockup:



Final Implementation:



Feature: Home Page

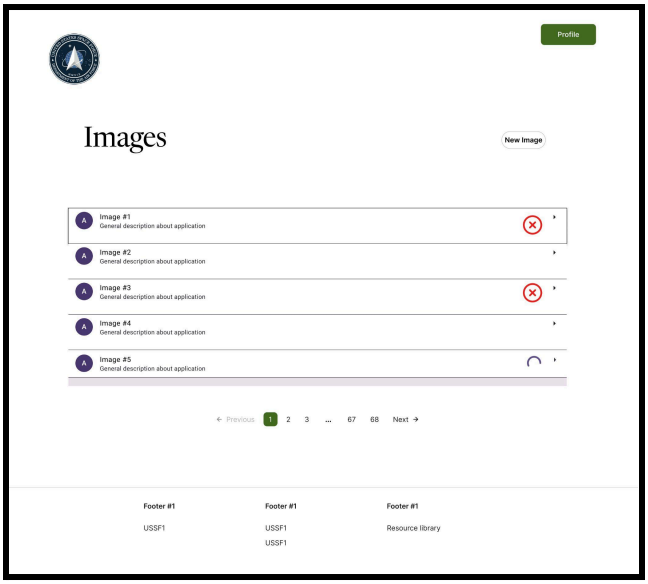
- **As a user** of the application
- **So that** I can view all my activities with all my images.
- **I want** a home page with a well-formatted list of all the images.

Sub Tasks:

- Display the images already uploaded by the user along with a button to go to the “Main Image page” and add a button that takes you to the “New Image page for GRC”

Implementation Status: Completed
Changes to Story during the Sprint: None

UI Mockup:



Final Implementation:

Feature: New Image Page for GRC

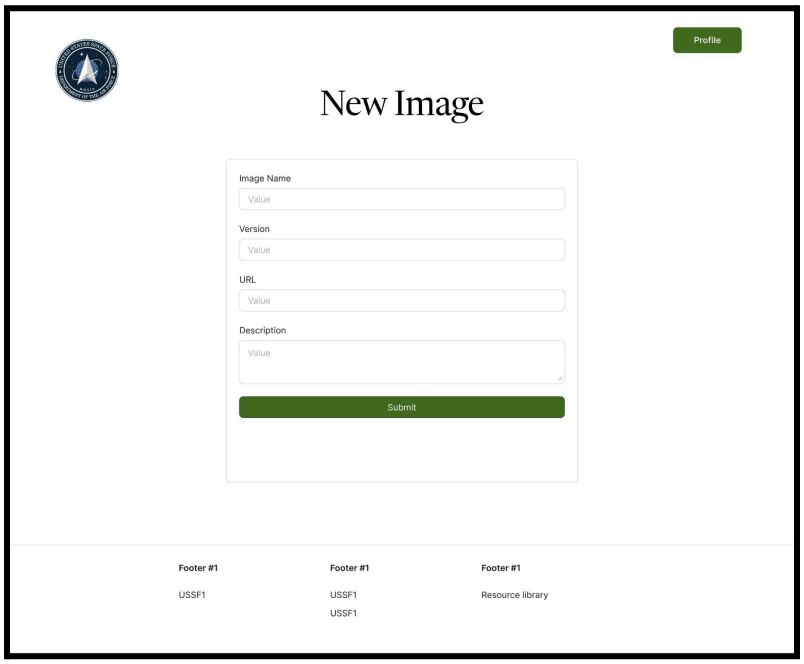
- **As a user** of the application
- **So that** I can add/upload a new image.
- **I want** a form where I can add details of the image.

Sub Tasks:

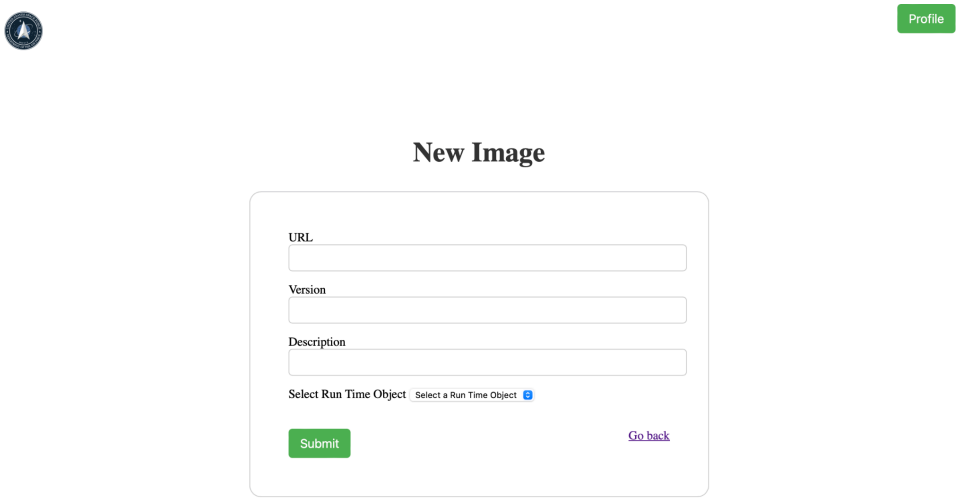
- Create a form for image upload with necessary fields (e.g., image name, tags) and add a dummy upload button that will finally take you to the “Main Image Page”.

Implementation Status: Completed
Changes to Story during the Sprint: Slight color scheme change for the buttons

UI Mockup:



Final Implementation:



Feature: Main Image Page

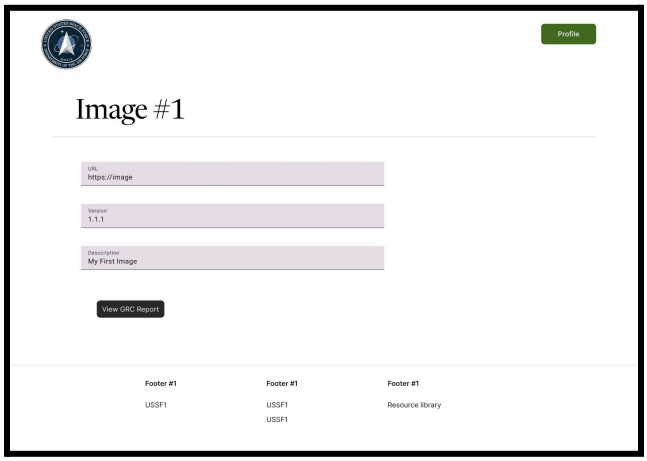
- **As a user** of the application
- **So that** I can view details of a created image
- **I want** a page where I can see the details of the image that I have created

Sub Tasks:

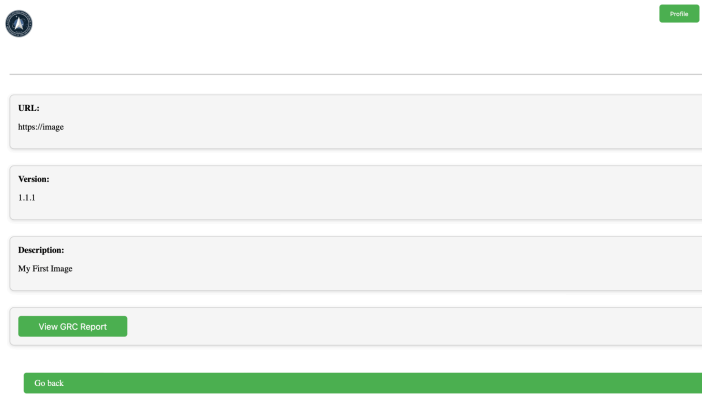
- Display uploaded image details on the main image page. Add a button (currently a dummy button) that links to the “Report Page”.

Implementation Status: Completed
Changes to Story during the Sprint: Slight color scheme change for the buttons

UI Mockup:



Final Implementation:



Feature: Report page

- **As a user** of the application
- **So that** I can view the compliance report of the scan
- **I want** a report page that shows the results of the GRC scan(NIST).

Sub Tasks:

- Integrate Google OAuth SSO Provider by creating a new project in Google Developer Console and Implement SSO Login & Session Management.
- Test and Validate SSO Functionality

Implementation Status: Completed
Changes to Story during the Sprint: None

Story Point Distribution:

Story	Story Points
Login to the application	4
Home Page	2
New Image Page for GRC	1
Main Image page	2
Report page	6

Client Sprint 1 Meeting Information

Date: 09/24/24
Time: 18:30 - 20:00
Location: Peterson Building
Attendees: Prof. Shreyas Kumar, Aditya Gourishetty, Maitreya Niranjan, Duy Vu, Medha Kaushika Podipireddi, Sahil Fayaz, Shravan Bhat, Vasudha Devarakonda

- The team demonstrated the deployed application to the client including the basic UI workflow and the different pages - Home Page, New Image Page, Main Image Page and Report Page
- The Login SSO mechanism and the use of Trivy - an open-source tool to run the GRC scan was communicated to the client.
- The client gave the following feedback:
    - A security analysis document for the project, including the external tools being used.
    - A product documentation including the limitations and the scope of the project.
    - The constraints and limitations should include the NIST800 controls we are committing to incorporate into the project. The team must analyze the controls Trivy is accommodating.
    - The client expects better aesthetics of the UI - A bigger Logo, better colors, and appearance overall.
    - The client also wants to have inline comments and documentation within the code for better code quality.
    - A Developers' documentation or a more comprehensive README was also suggested.

Sprint 2

Team Roles:

- **Scrum Master** Medha Kaushika Podipireddi
- **Product Owner**: Sahil Fayaz
- **Developers:** Duy Vu, Maitreya Niranjan, Shravan Bhat, Vasudha Devarakonda, Aditya Gourishetty

Summary: In Sprint 2, we have enhanced the UI and navigation for better user experiences and also derived inferences from the generated output by the third-party security scanner, Trivy. Additionally, we also performed security due diligence on the Trivy to ensure the inputs that we have are securely getting scanned. We presented the same to the USSF clients and they have agreed upon the design and features to be in alignment with their idea of the solution to the problem statement. The application is currently hosted on Heroku. We have also achieved good code quality having used rubocop, rubycritic, and code coverage for rspec and cucumber test cases. All the committed stories for the sprint have been marked as done.

Total Points Completed: 15

User Stories

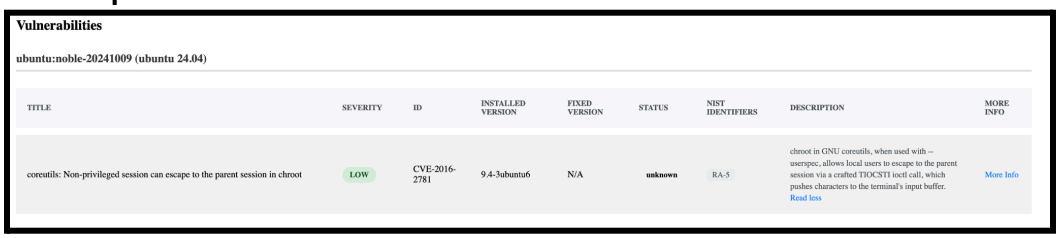
Feature: Derive & Show Inferences from GRC Report

- **As a user** of the application
- **So that** I need to be able to view the security analysis of the run-time object on which the controls are getting evaluated and decide whether to go ahead with the deployment of the run-time object securely
- **I want** a page where I can see the details of the different vulnerabilities and their severity along with the details of the vulnerability.

Sub Tasks:

- Enhanced Vulnerability Report Page with Filters and Improved Layout
- Vulnerability Analysis and Deployment Decision with Trivy-NIST Mapping

Final Implementation:



Vulnerabilities

ubuntu:noble-20241009 (ubuntu 24.04)

TITLE	SEVERITY	ID	INSTALLED VERSION	FIXED VERSION	STATUS	NIST IDENTIFIERS	DESCRIPTION	MORE INFO
coreutils: Non-privileged session can escape to the parent session in chroot	LOW	CVE-2016-2781	9.4-3ubuntu6	N/A	unknown	RA-5	chroot in GNU coreutils, when used with --userspec, allows local users to escape to the parent session via a crafted TIOCSTI ioctl call, which pushes characters to the terminal's input buffer. Read less	More Info

Implementation Status: Completed
Changes to Story during the Sprint: None

Feature: Security Due Diligence of Trivy

- **As a stakeholder** of the application
- **So that** I can ensure the third-party library Trivy does not have any security vulnerabilities
- **I want** a thorough evaluation of Trivy including an assessment of potential vulnerabilities and risks

Documentation: Link

Implementation Status: Completed
Changes to Story during the Sprint: None

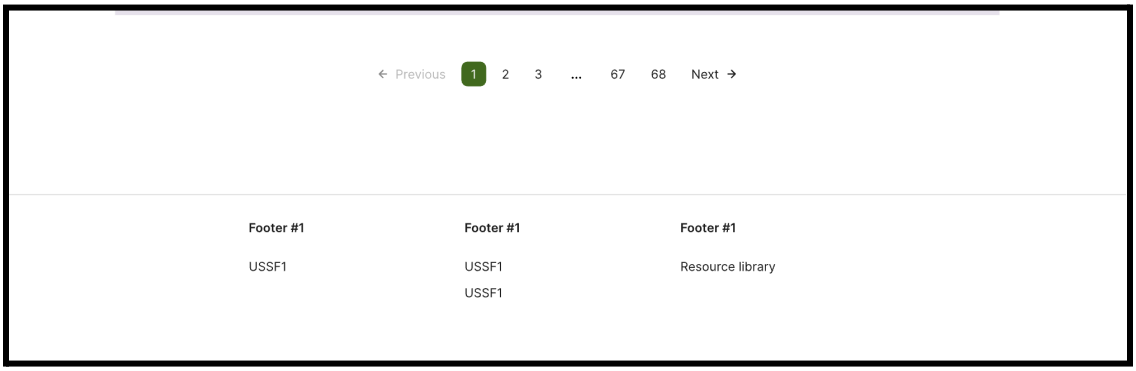
Feature: Pagination in Homepage

- **As a user** of the application
- **So that** I can easily navigate through large number of images in the homepage
- **I want** the homepage to include pagination allowing me to view images in smaller and manageable sections

Sub Tasks:

- Show the paginated images
- Implement the pagination controls

UI Mockup:



Final Implementation:

Implementation Status: Completed
Changes to Story during the Sprint: None

Feature: Enhancing UI of the application

- **As a user** of the application
- **So that** I have a better experience while using the application
- **I want** all the pages/views to be visually consistent with all logos and stylings

Implementation Status: Completed
Changes to Story during the Sprint: None

Feature: Rerun scan for the image

- **As a user** of the application
- **So that** I can ensure the latest security vulnerabilities are detected for that particular image
- **I want** the option to rerun the scan for the selected image and view updated scan results.

Final Implementation:

Implementation Status: Completed
Changes to Story during the Sprint: None

Story Point Distribution:

Story	Story Points
Derive & Show Inferences from GRC Report	6
Security Due Diligence of Trivy	2
Pagination in Homepage	3
Enhancing UI of the Application	3
Rerun scan for the image	1

Client Sprint 2 meeting information

Date: 10/24/2024
Time: 18:30
Location: Google Meet
Attendees: Aditya Gourishetty, Maitreya Niranjan, Duy Vu, Prof. Shreyas Kumar, Medha Kaushika Podipireddi, Sahil Fayaz, Shravan Bhat, Vasudha Devarakonda, Lt. Theresa Kopecky, Major Eric Griffin

- The team demonstrated the deployed application to the client including the UI and scanning workflow and the different pages of the application.
- Security due diligence of Trivy was also communicated to the client.
- The client gave the following feedback:
    - The USSF representatives agreed on the design and workflow stating that the solution we provided was in alignment with their idea.
    - Additional requirements:
        - One page overview about building a service like Trivy that includes details like estimated time of building such service, complexities involved in it, etc.,
        - How is our application different from any other automated vulnerability scanning applications available?
        - What is the scalability of the application and Trivy(third-party scanner)?
        - Can our application be integrated into the dev pipeline in its current stage or any other fine-tuning required for the purpose?

Sprint 3

Team Roles:

- **Product Owner:** Maitreya Niranjan
- **Scrum Master:** Duy Vu
- **Developers:** Aditya Gourishetty, Shravan Bhat, Maitreya Niranjan, Medha Kaushika Podipireddi, Sahil Fayaz, Vasudha Devarakond

Summary: In Sprint 3, our team successfully advanced the GRC Control application by implementing features that enhance report handling and runtime object management. A key achievement was enabling users to download the GRC report in CSV format, making compliance tracking easier and more accessible. We introduced a new home page where users can view all their runtime objects and detailed pages for editing, deleting, and managing associated images. Additionally, we created functionality to add new runtime objects and share them with other users in a read-only capacity, providing flexibility in object visibility and collaboration. By addressing critical features like editing and deleting both runtime objects and images, and fixing a reloading bug on the report page, we aimed to streamline the user experience, making the application more efficient for compliance monitoring and data

management. This sprint solidified the application's core functionality and enhanced user control over runtime objects and their associated images.

Total Points Completed: 15

User Stories

Feature: Download and view the GRC report as a CSV

- **As a user** of the application
- **So that** I can download the report to my local machine
- **I want** to convert the report to CSV format and download it

  Subtasks:

  - Implement download functionality
  - Integrate report

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  UI Mockup:

  Final Implementation:

Feature: New RunTime Object Page

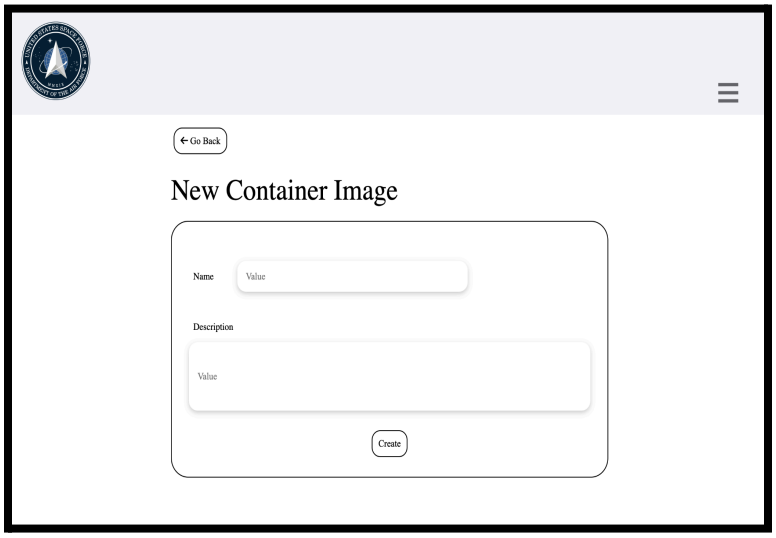
- **As a user** of the application
- **So that** I can create a new run-time object for scanning.
- **I want** a form where I can give details of the new object.

  Subtasks:

  - Show the form for the new run-time object
  - Save new object to the database

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  Final Implementation:



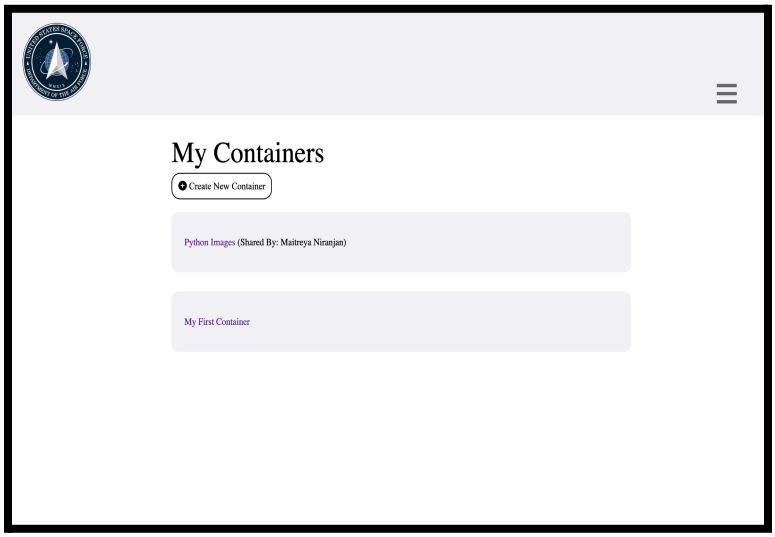
Feature: View RunTime Objects in Home Page

- **As a user** of the application
- **So that** I can view all my run-time objects
- **I want** a home page with a formatted list of objects I created.

Implementation Status: Completed
Changes to Story during the Sprint: None

UI Mockup:

Final Implementation:



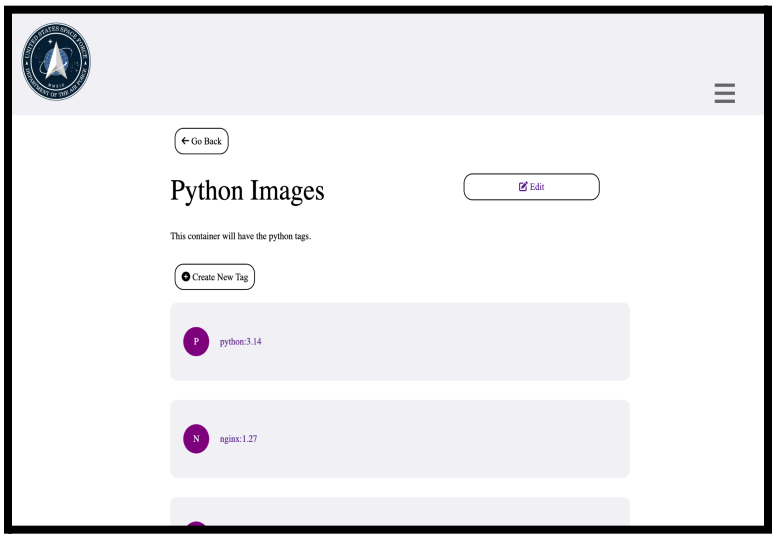
Feature: View Images of RunTime Object

- **As a user** of the application
- **So that** I can view different images associated with each run-time object created by me
- **I want** a page with a formatted list of images linked to a particular run-time object.

Implementation Status: Completed
Changes to Story during the Sprint: None

UI Mockup:

Final Implementation:



Feature: Edit the run-time object

- **As a user** of the application
- **So that** I can edit the existing run-time object
- **I want** a page where I can edit a particular run-time object

  Subtasks:

  - Create an edit form for a run-time object
  - Update the run-time object in the database

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  Final Implementation:

Feature: Delete the run-time object and all the images associated with it

- **As a user** of the application
- **So that** I can delete the existing run-time object
- **I want** a button where I can delete a particular run-time object and all its associated images.

  Subtasks:

  - Remove run-time objects and related images from the database.

  Implementation Status: Completed
  Changes to Story during the Sprint: None

Feature: Edit the image

- **As a user** of the application
- **So that** I can edit the existing image
- **I want** a page where I can edit a particular image

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  Final Implementation:

Feature: Delete the image

- **As a user** of the application
- **So that** I can delete the existing image
- **I want** a button where I can delete a particular image without affecting the other images.

  Implementation Status: Completed
  Changes to Story during the Sprint: None

Feature: Permission

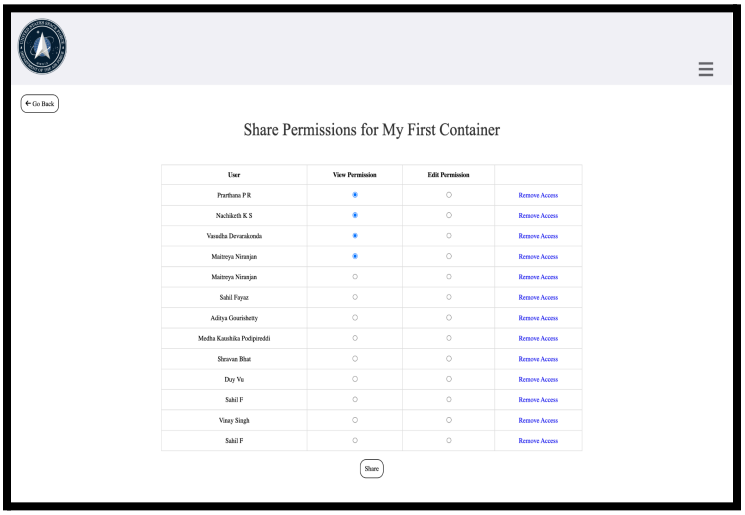
- **As a user** of the application
- **So that** I want to share the runtime objects with other (read-only)
- **I want** a button to share with other users (read-only)

  Subtasks:

  - On the runtime object detail page, there should be a button to share with other users. When tapping on the button, show a list of users that you want to share with.

Implementation Status: Completed
Changes to Story during the Sprint: None

Final Implementation:



Story Point Distribution:

Story	**Story Points**
Download and view the GRC report	3
Run-time object on the home page	1
Edit Run-time object	2
Delete Run-time object	1
Edit Image	2
Delete Image	1
Create a new run-time object	2

Permission	3

Client Sprint 3 meeting information:

Date: 11/07/2024
Time: 19:00 - 20:00
Location: Google Meet
Attendees: Aditya Gourishetty, Maitreya Niranjan, Duy Vu, Prof. Shreyas Kumar, Medha Kaushika Podipireddi, Sahil Fayaz, Shravan Bhat, Vasudha Devarakonda

The team presented the latest version of the application to Prof. Sheyras Kumar, showcasing the application navigation, scanning and downloading functionality, and container-sharing feature across users.

Feedback from Prof. Sheyras Kumar**:**

- Suggested interface updates for clarity and user experience:
    - Ensure the image URL and title are displayed in a single, horizontal line for consistency.
    - Add a descriptive "Help" section in the menu that explains the purpose of each feature.
    - Remove the display of email addresses in the info section.
    - Standardize button shapes and colors for visual consistency across the application.
- Requested functional improvements:
    - Fix the bug where the application reloads after a download is completed.
    - Provide more details on images, particularly default images.
    - Implement sorting functionality in tables, specifically by properties such as ID.
    - Modify "Unknown" or "Uncategorized" labels to something more descriptive.
- Access control and user roles:
    - Create user roles, such as Admin, with permission management (e.g., ability to delete users).
    - Rename "Access" to a more intuitive term like "Share" to avoid confusion.
- Accessibility:
    - Make the website accessible for users with disabilities, especially considering government ADA compliance. Suggested features for color blindness and screen-reader support were discussed.

Additional requirements**:**

- Adjust font sizes for better readability, ensure titles and colors are consistent, and enhance the user experience for visually impaired users if feasible.

Sprint 4

Team Roles:

- **Product Owner**: Medha Kaushika Podipireddi
- **Scrum Master** : Vasudha Devarakonda
- **Developers:** Sahil Fayaz, Shravan Bhat, Aditya Gourishetty, Maitreya Niranjan, Duy Vu, Medha Kaushika Podipireddi

Summary: In Sprint 4, our team successfully advanced the GRC Control application by implementing features that enhance the user management of the applications. Specifically three types of users were created; admin, image owners and general users. Further general users will be given permission to either view or edit the image based on the permissions given by the image owners. Further the application supports scanning of private registry images. With this feature users can scan the images in a private registry by adding their credentials(temporary and not saved). By adding this feature, the application supports all types of container images. Further, the view of the vulnerability report was improved where users can now filter and sort for an easy and structured view of the report. Additionally, considering the request from the client, the accessibility of the application was improved. Navigations were added to each element of the application and was further tested with NVDA to ensure accessibility to impaired users. Additionally, color schema and styling was selected to improve color blind accessibility.

Total Points Completed: 18

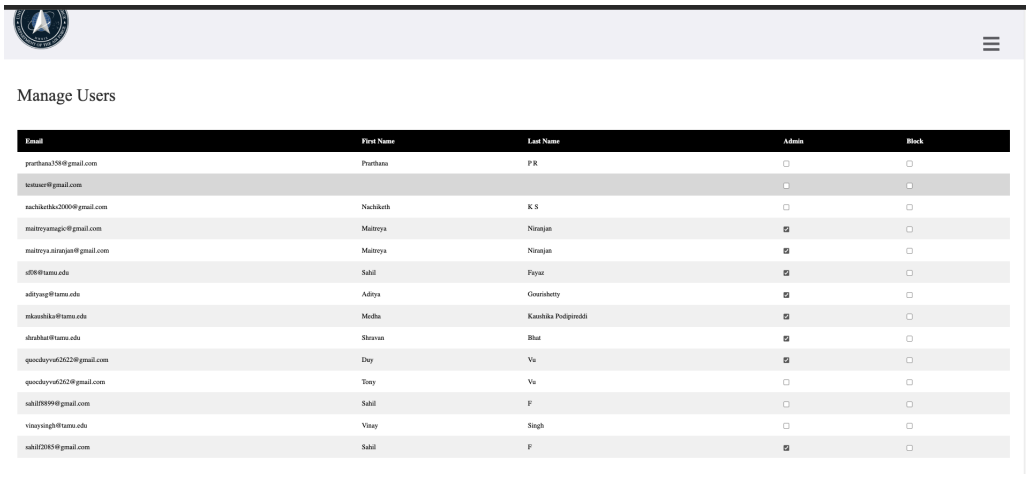
User Stories

Feature: Add User types

- **As a user** of the application
- **So that** I can have a centralized governance of the product
- **I want** an “admin” user and a “general” user who can share their applications with each other

  Implementation Status: Completed
  Changes to Story during the Sprint: None

Final Implementation:



Feature: Refine UIs

- **As a user** of the application
- **So that** I can have a better user interface
- **I want** consistency in my views irrespective of the device we are viewing it on.

  Subtasks:

  - Make buttons and texts more consistent and compatible with all types of devices
  - Add description of image tasks

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  Final Implementation:

Feature: Refine functionalities and actions

- **As a user** of the application
- **So that** I can see the report
- **I want** consistency in my actions .

  Subtasks:

- Bug:See the vulnerability report immediately without reloading
- Bug:Fix downloading report feature

Implementation Status: Completed
Changes to Story during the Sprint: Following the feedback from client(last sprint), a loading object was added to indicate that the scanning is in progress

Feature: Refine Access Control

- **As a user** of the application
- **So that** I can take action based on the permissions granted
- **I want** to edit or view the image page as per sharing options.

Subtasks:

- Include granular sharing options like edit and view for runtime objects
- Remove delete option in collaborator view, only owners should have the option

Implementation Status: Completed
Changes to Story during the Sprint: None

Final Implementation:

Feature: Make the application Accessible

- **As a user** of the application
- **So that** I can make the application more accessible to end users
- **I want** a help to navigate and accommodate multiple people

Subtasks:

- Show help: what a certain action or a field does while hovering over them
- Make the UI accessible to color lindess especially the report page [Reference]

Implementation Status: Completed
Changes to Story during the Sprint: None

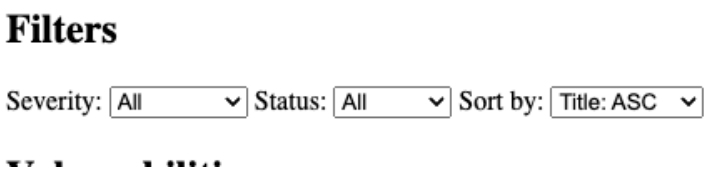
Final Implementation:

Feature: Add Sort feature based on some properties

- **As a user** of the application
- **So that** I can sort my vulnerability report based on various properties together instead of the existing severity-only filter
- **I want** ascending and descending sort ability per properties available in the report.

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  Final Implementation:

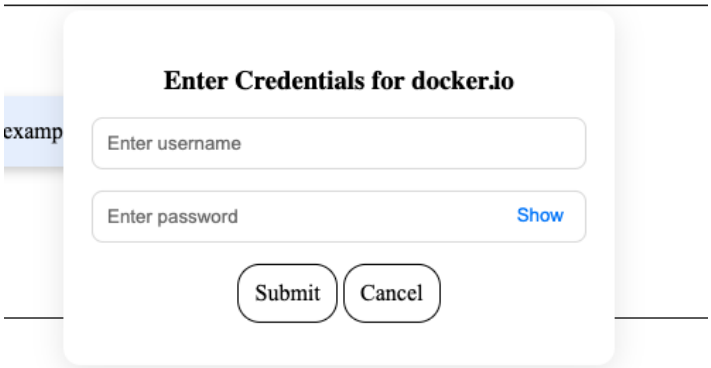


Feature: Add feature to scan private images

- **As a user** of the application
- **So that** I can scan image from private registry
- **I want** to able to fetch the image from private registry and check for vulnerabilities

  Implementation Status: Completed
  Changes to Story during the Sprint: None

  Final Implementation:



Story Point Distribution:

Story	**Story Points**
Add user types	3
Refine UIs	2
Refine functionalities and actions	3
Refine Access Control	3
Make the application Accessible	3
Add Sort feature based on some properties	1
Add feature to scan from private registry	3

Client Sprint 4 meeting information:

Date: 11/21/2024
Time: 18:15 - 19:00
Location: Google Meet
Attendees: Prof. Shreyas Kumar, Major Eric Griffin, Oslyn Marie, Aditya Gourishetty, Maitreya Niranjan, Duy Vu, Medha Kaushika Podipireddi, Sahil Fayaz, Shravan Bhat, Vasudha Devarakonda

The team presented the latest version of the application to Prof. Sheyras Kumar and the USSF team, showcasing the application navigation, scanning, and downloading functionality, different types of users(admin and regular), accessibility of the application for color blindness, user feasibility of usage of the application, and access control features across users. We also presented security due diligence of the application along with answers to the additional requirements raised by Lt. Theresa Kopecky in the previous sprint(Sprint 2).

The additional requirements were:

- One page overview about building a service like Trivy that includes details like estimated time of building such service, complexities involved in it, etc.,
- How is our application different from any other automated vulnerability scanning applications available?
- What is the scalability of the application and Trivy(third-party scanner)?

The USSF clients were impressed by the application and there were no further improvements suggested. They wanted us to present the application demo to their seniors on the day of the final presentation in class so that their team could draw up tasks for next semester.

Member Contribution

Team Member	Sprint 1		Sprint 2		Sprint 3		Sprint 4		Total Contribution	
	Stories	**Points**	**Stories**	**Points**	**Stories**	**Points**	**Stories**	**Points**	**Stories**	**Points**
Aditya Gourishetty	(Scrum Master)		1	3	2	3	2	3	**5**	**9**
Duy Vu	2	3	1	3	(Scrum Master)		2	3	**5**	**9**
Maitreya Niranja	2	3	2	3	(Product Owner)		1	3	**5**	**9**
Medha Kaushika Podipireddi	1	3	(Scrum Master)		1	3	1	3	**3**	**9**
Sahil Fayaz	1	3	(Product Owner)		1	3	1	3	**3**	**9**
Shravan Bhat	(Product Owner)		1	3	2	3	2	3	**5**	**9**
Vasudha Devarakonda	1	3	1	3	2	3	(Scrum Master)		**4**	**9**

BDD/TDD Process

Throughout the project, we followed a Behavior-Driven Development (BDD) and Test-Driven Development (TDD) process. This approach allowed us to catch bugs early, ensure code quality, and align features with customer needs through clear communication with stakeholders.

We used tools like Cucumber for BDD, RSpec for TDD, Capybara for browser-level behavior testing, and SimpleCov to ensure comprehensive coverage.

TDD Process

1. Approach:
    - Test-Driven Development (TDD) was used to guide low-level implementation details.
    - We followed the Red-Green-Refactor cycle:
        - Red: Write a failing test.
        - Green: Write code to pass the test.
        - Refactor: Improve code quality while keeping tests green.
2. Tools Used:
    - RSpec for unit tests and controller-level testing.
    - SimpleCov to ensure comprehensive test coverage for both backend and frontend logic.
3. Example Workflow: Validate that only users with the appropriate permissions can view a Docker image report.
    - Write a test that fails when unauthorized access occurs.
    - Implement role-based access control logic to pass the test.
    - Refactor the authorization code for readability and efficiency.
4. Benefits:
    - Increased Confidence: Writing tests first ensured all features worked as intended.
    - Bug Prevention: TDD helped catch issues early in the development cycle.
    - Maintainable Codebase: Tests acted as a safety net, enabling smooth refactoring without fear of introducing new bugs.
5. Challenges:
    - Time-Intensive: Writing tests before coding increased the upfront development time.
    - Edge Cases: Writing comprehensive tests for edge cases in Docker scanning and GRC mapping was complex.

BDD Process

1. Approach:
    - We adopted Behavior-Driven Development (BDD) to ensure that our application's features aligned with the primary customer needs of assessing GRC compliance for Docker images.
    - Feature behaviors were defined in plain language scenarios (e.g., "As a user-So that-I want"), ensuring clear communication between developers and stakeholders.
    - Example:
        - **As a user** of the application
        - **So that** I can have the vulnerability report generated, mapping findings to NIST SP800-53 controls
        - **I want** to input the docker image for scanning

2. Tools Used:
    - We used Cucumber for defining and automating BDD scenarios.
    - Capybara was integrated to test application behaviors in a simulated browser environment.
3. Benefits:
    - Improved Stakeholder Collaboration: Writing tests in plain language helped align development goals with customer expectations.
    - Enhanced Understanding of Requirements: Scenarios were living documentation for the app's expected behaviors.
    - Reduced Miscommunication: Using customer-friendly terms ensured we implemented features that directly addressed their needs.
4. Challenges:
    - Scenario Maintenance: As the application evolved, keeping BDD scenarios in sync with changing requirements required additional effort.
    - Initial Learning Curve: Team members needed time to adapt to writing Gherkin syntax and managing Cucumber workflows.

Configuration Management Approach

Our configuration management approach followed Git flow principles, leveraging Git for version control to maintain separate branches for development, features, and production. Throughout the project, we managed 56 branches and released five versions. This structured approach allowed for clear collaboration, streamlined code integration, and efficient management of releases.

Technical Spikes and Explorations

1. **Vulnerability Scanning Tool Evaluation**
   Objective: Evaluate and compare Trivy and Clair scanners to determine the best fit for our project's vulnerability scanning requirements.
   Process: Conducted a proof of concept to compare both tools across multiple criteria such as installation ease, performance, reporting capabilities, and community support.
   Outcome: Based on the evaluation, Trivy was selected for its:
    - Minimal setup (a single binary executable).
    - Faster scanning times and smaller database footprint.
    - Versatile platform support (e.g., Docker, Kubernetes, AWS Lambda).
    - Comprehensive reporting formats for CI/CD pipelines.
    - Robust community support and active development.

2. **Security Review for Trivy Usage**
   Objective: Assess the security of Trivy, given the high-security requirements of our client (USSF).

Process: Conducted an in-depth review of Trivy's operational and security practices to ensure compliance with stringent standards.
Outcome: Trivy was confirmed to meet the required security standards based on:

- Privacy by Design: Trivy operates locally, ensuring no user data, secrets, or scan results are collected or transmitted.
- Comprehensive Misconfiguration and Secret Detection: Protects against insecure practices and exposed credentials.
- Automated Updates: Maintains up-to-date vulnerability databases without manual intervention.
- Local Scanning: Ensures all sensitive data remains within the user's infrastructure.

3. **ADA Accessibility for Website Development**
   Objective: Ensure compliance with accessibility standards to make the website inclusive for all users, especially those with disabilities.

   Approach:
   - Screen Reader Compatibility: Tested using NVDA to ensure blind users could navigate the site seamlessly. This included using semantic HTML for structured content and enabling focusable interactive elements for keyboard navigation.
   - Color-Blind Accessibility: Incorporated a high-contrast color palette and avoided reliance on color alone to convey information. For example, error messages include both text and color for clarity.
   - General Accessibility Practices:
     - Semantic HTML elements ensure logical reading order.
     - Keyboard-friendly navigation supports non-mouse users.
     - Accessible forms feature clear labels and understandable error messages.
     - Exclusion of images reduces potential barriers and improves performance.

Production Release Issues to Heroku

During the production release process to Heroku, we encountered issues that have been listed in this section along with their respective solutions.

Issue 1:

- Ruby version blocks release to heroku: If the ruby version is not explicitly mentioned in the Gemfile, heroku might use use an older version of ruby which can be verified from

the logs as below

```
remote:
remote:        We could not determine the version of Ruby from your Gemfile.lock.
remote:
remote:          $ bundle platform --ruby
remote:          No ruby version specified
remote:
remote:          $ bundle -v
remote:          Bundler version 2.5.6
```

This might install an older version of ruby which might not be compatible with other gems.
Solution : Mention the version of ruby in the Gemfile and retry the deployment. The above solution can be verified from the logs

```
remote: -----> Building on the Heroku-24 stack
remote: -----> Using buildpacks:
remote:        1. heroku/ruby
remote:        2. https://github.com/tamu-edu-students/buildpack-trivy
remote: -----> Ruby app detected
remote: -----> Installing bundler 2.5.6
remote: -----> Removing BUNDLED WITH version in the Gemfile.lock
remote: -----> Compiling Ruby/Rails
remote: -----> Using Ruby version: ruby-3.3.4
```

Issue 2:

- Sometimes the db:migrate might not succeed. This can be verified by running command `heroku run rake db:migrate:status`

```
database: d45lvt1egct4ue

 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20240930152837  Create users
   up     20240930153226  Add oauth details to users
   up     20241001034223  Create run time objects
   up     20241001034250  Create images
   up     20241018213708  Create cve nist mapping
   up     20241026165257  Create run time objects permissions
  down    20241108233324  Add admin to users and make user admin
  down    20241113043707  Add block to users
```

Solution: To make the above migration successful, manually update the status of the migration by using the migration ID from the status. Run command `heroku run rake db:migrate:up VERSION=<version-number>` . Verify the migration by running the status command again

```
heroku run rake db:migrate:up VERSION=20241113043707
```

```
database: d451vt1egct4ue

 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20240930152837  Create users
   up     20240930153226  Add oauth details to users
   up     20241001034223  Create run time objects
   up     20241001034250  Create images
   up     20241018213708  Create cve nist mapping
   up     20241026165257  Create run time objects permissions
 down     20241108233324  Add admin to users and make user admin
   up     20241113043707  Add block to users
```

Issue3:

- `heroku buildpacks:add https://github.com/tamu-edu-students/buildpack-trivy --index 2` might sometimes fail. This can be verified using command `heroku buildpacks`

Solution: Please verify that `https://github.com/tamu-edu-students/buildpack-trivy` is accessible

Warning:

- `heroku run rails db:seed` takes upto 15 minutes

Tools/Gems Used

GitHub: Used for version control. It enabled efficient collaboration
GitHub Projects: Helped virtualize the Scrum Board.
CodeClimate: Assisted in maintaining code quality by identifying areas that needed improvement.
SimpleCov: Provided test coverage metrics to ensure we had adequate test coverage.
Pagy: This gem was used for pagination actions.
OmniAuth: This gem was used for configuring Google SSO authentication.

Repo Contents and Deployment Process

The repository contains everything necessary for deploying the application, including:

- README explaining the local setup and deployment process explicitly.
- Code for the application

- Test files (Cucumber, RSpec)

Project Links

- **GitHub Repo**: USSF-CSCE606
- **Project Management Page:** USSF-CSCE606-Project-Management
- **Slack Workspace**: Fall 2024 CSCE606: USSF GRC Control Kubernetes
- **Deployed App URL**: USSF-GRC

Presentation and Demo Links

- **Demo Video**: USSF-CSCE606
- **Presentation Slides:** USSF-CSCE606