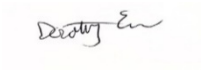
AutoDSO

by

Dorothy Eves, Vance Phu, and Chelsea Gantt

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

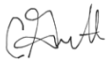


04/12/2020

Dorothy Eves — Date

04/12/2020

Vance Phu — Date



04/12/2019

Chelsea Gantt — Date

Bogdan Vykhovanyuk 04/12/2020

Bogdan Vykhovanyuk — Date

University of Cincinnati

College of Education, Criminal Justice, and Human Services

April 12, 2020

Table of Contents

List of Illustrations

Tables

Figures

AutoDSO

Abstract

Companies are generally hyper-focused on releasing software quickly to meet strict deadlines or to stay ahead of the competition. This generally results in implementing security once software is near release. In applying DevSecOps best practices early, companies can not only catch and fix security problems early but train their teams in what to avoid in the future. For those organizations that implement DevSecOps, it is found to be implemented during the beginning stages of their software development projects. Most of which is focused on identifying and displaying the source of security misconfigurations. AutoDSO will base their security best practice requirements based off the OWASP DevSecOps model. AutoDSO takes DevSecOps a step further by focusing on establishing security baseline requirements and allowing the users to select metrics specific to their DevSecOps policies and procedures. The application generates an automated DevSecOps policy document which can be used by security analyst and given to auditors to monitor a company's DevSecOps processes. This allows companies an efficient way to document their security best practices and ensuring that security becomes a part of continuous integration and continuous development (CI/CD) in their organization.

Problem Statement

Introduction

In today's rapidly evolving tech industry, companies are generally hyper focused on releasing software quickly to stay ahead of competition. This results in lack of implementation and integration of security at the beginning of the Software Development Lifecycle. In 2019, 65% of U.S. companies disclosed that they were victims of a data breach. Statistics shed light that a hacking event occurs every 39 seconds. Half of organization spend only around 6% – 15% of their security budget on data security and discovery time for 56% of data breaches can take months or even longer to identify. This demonstrates the need to not only implement security in the SDLC, but also document policies and procedures that will govern security in DevSecOps. AutoDSO was inspired by Dorothy Eves, Chelsea Gantt and Vance Phu. Dorothy initially proposed a Cybersecurity Education tool to promote user awareness and upon review, the team's advisor suggested a DevSecOps tool that will specifically benefit businesses by helping create documentation for DevSecOps processes in an agile environment. Chelsea proposed creating sections that would provide an overview of what DevSecOps is, best practices, and useful links that can be used as reference for the user.

Problem

Security implementation in the Software Development Lifecycle process is often left at the end of the development phase and not given a top priority. Due to the lack of automation for company specific security metrics and training for teams, DevSecOps processes aren't fully adopted into many organizations. In order to have a streamlined process, it requires cross-functional support in organizations for security. Security teams and development teams don't generally communicate in the beginning of the development phase and have separate functions. Security teams aren't usually involved until vulnerabilities are exposed in the applications or until testing may ensue. Many organizations work on time constraints which is a factor in the lag of security processes being fully integrated very early on in projects.

It is AutoDSO's mission to provide an efficient tool, that will not only automate the creation of company specific security metrics that can help lessen the time of implementing security into a development project early on, but to provide a knowledge base for DevSecOps processes that teams can adopt and use to fully integrate DevSecOps with cross-functional support in their organization.

Solution

AutoDSO is a web application that serves as an all in one guide for security integration into DevOps, also known as DevSecOps. The application will provide businesses with the ability to document what metrics they want to follow, by selecting and inputting timeframes, goals, and numbers they want them to be measured by. Using the information provided by the business, AutoDSO will then produce customized documentation to outline a security policy which can be implemented and given to auditors and security analyst for review. These reports are specifically designed to be implemented within the scrum and agile methodology in order to provide a more disciplined outline for their security processes. AutoDSO will also provide sections designed to give an informative overview of what the application does, an overview of what DevSecOps is and best practices, and finally links to useful resources.

While comparing other DevSecOps solutions to AutoDSO, we found that most of the solutions out there, for example Checkmarx and Continuum Security are two of the most currently used DevSecOps tools. These two companies are mainly focusing on implementing DevSecOps to search for security misconfigurations. AutoDSO will focus on providing security best practices in the beginning stages of their software development projects. AutoDSO will take DevSecOps one step further by allowing users to be able to select metrics provided by a template and document their security baseline requirements needed to implement into their DevSecOps policies and procedures.

AutoDSO will also generate an automated document based on the metrics selected and will be able to automatically take those metrics and create a DevSecOps policy document which can be used by security analysts and given to auditors.

Project Description

AutoDSO will allow users to set up their account and begin to view their DevSecOps security best practices. Once the user has been set up, they can create their security metrics by opening the template and inputting their security metrics. Then the application will automatically generate a security metric document that the user can view and export the document to store it into the company's resource to store documents. Once a user gets logged into their account, they will have the ability to view historical data that shows their previous documents and users will be able to update and delete them as needed.

The remainder of this final report outlines in detail how the project was completed. The report includes the following sections: Design objectives, budget, timeline, technical elements, project testing, testing scenarios, problems encountered, future recommendations, and conclusions.

User Profile

Table 1: User Profile displays the user profile used throughout the development of AutoDSO's web application. It highlights details of the market we are creating our web application for. After beginning the development of the application our team has determined the potential users for AutoDSO.

PROJECT: AutoDSO
POTENTIAL USERS: • Security Analysts • Developers • Project Managers • Business Stakeholders • Auditors -
SOFTWARE, INTERFACE, AND RELATED EXPERIENCE: AutoDSO users will be technology savvy and familiar with using desktop and web applications. They will be familiar with navigating various web browsers and efficient with word processing applications. All users will be using the same functions.
EXPERIENCE WITH SIMILAR APPLICATIONS: Users would be familiar with applications like Word, PowerPoint, Visio where they can use templates to create documents.
TASK EXPERIENCE: Initially, users will create an account with AutoDSO and view the information for the DevSecOps security best practices. They can create their security metrics by opening the template and entering in their metrics. The application will automatically create a security metric document and the user can then export the document and add to their own company resource to document their security practices. If a user would like to view historical data, they can login to AutoDSO and view their previous documents. Users can login to view, update and delete documents as needed.
FREQUENCY OF USE: The users of this application would use this on an as needed basis for creating security metric documents and as a point of reference for information on the DevSecOps process. Periodically they may use to modify metrics and view historical metrics. Dependent on auditing frequency, this will be used to reference security metrics for auditing purposes.
KEY PROJECT DESIGN REQUIREMENTS THAT THE PROFILE SUGGESTS: • Easy for the users to set up an account • User-friendly interface • Efficient processing of document creation and export • Fluid responsive design

Table 1: User Profile

Potential Users

Potential users for this application include Project Managers that may want to establish security metrics for the development team to follow when beginning their projects. Security analysts that are knowledgeable about security best practices that can analyze and suggest what metrics to implement. Development teams that can use this document in order to make sure they are following best practices as they are creating their projects. Business Stakeholders who may want to review how the development teams are following DevSecOps processes and Auditors who may want to audit a company's security best practices to ensure they are adhering to requirements.

Software and Interface Experience

Develop a web application that will allow developers to implement DevSecOps in the beginning stages of their software development projects and provide automated documentation outlining their specific security metrics, ensuring they are catching vulnerabilities and bugs early on. AutoDSO will also provide resources and a guideline for following DevSecOps that users can reference.

The features of AutoDSO will include:

- Clear guideline for the DevSecOps process in an agile environment
- Resource area that provides information in implementing DevSecOps during software development
- Users will be able to select metrics provided by a template to document their security baseline requirements needed to implement into their DevSecOps policies.
- Produces an automated document based on the metrics selected outlines a DevSecOps policy which can be used by security analysts and auditors.
- Resources area for additional DevSecOps information

Together these features will provide an automated solution for DevSecOps documentation specific to the business's security requirements in an agile environment.

Experience with Similar Applications

There aren't any current applications that automate a document for security metrics. Most applications focus upon automated security code review in order to establish if there are any vulnerabilities in the code. Most noteworthy is SonarQube. This application automates security code reviews and highlights to the users the area's that are not adhering to best security practices, it also gives them tips and advice on implementing these best practices. Organizations like OWASP may come closest to what AutoDSO wants to implement regarding companies establishing security metrics. This organization has documentation on suggested security metrics that teams can use in order to implement DevSecOps best practices. Yet, this is only a reference and the users must create their own document in order to have this information in their hands. Our application takes this documentation a step further, by helping organizations easily enter in their security metrics into an application and automating the creation of a document, they can use.

Task Experience

Users will first create an account and login into the application and create a new security metric. They can reference the baseline requirements and see documentation on suggestions for the security metrics they can implement that will adhere to the DevSecOps best security practices. Users will input each metric specific to their organizational needs and select the Create button, that will autogenerate a document with all of their metrics. They may export this document to share with their team and put with their organization's documentation area. Any user that has an account will be able to visit AutoDSO and view their historical documents, delete or update it.

Frequency of Use

Users may need to use this application annually or quarterly dependent on when they review their security metrics and processes. This can include for auditing purposes or for the team to measure their success in following their DevSecOps best practices. Users may occasionally visit AutoDSO in order to update their metrics or review current metrics and make any changes they may need.

Key Project Design Requirements That the Profile Suggests

Whenever users need to create, view, or print out and redistribute policy documents. This occurrence can happen daily, weekly, monthly, or yearly. Also, developers will be constantly using AutoDSO while developing code. This occurrence can happen daily or weekly depending on the organization.

- Easy for the users to set up an account
- User-friendly interface
- Efficient processing of document creation and export
- Fluid responsive design

Use Case Diagram

Figure 1: Use Case Diagram presents the project use case diagram. AutoDSO use cases consist of 5 parties in total. All parties will have the same functionality. These parties consist of the Project Manager, Auditor, Security Analyst, Business Stakeholder, and Developers. Created users will have the additional option to view historical documents and export them.

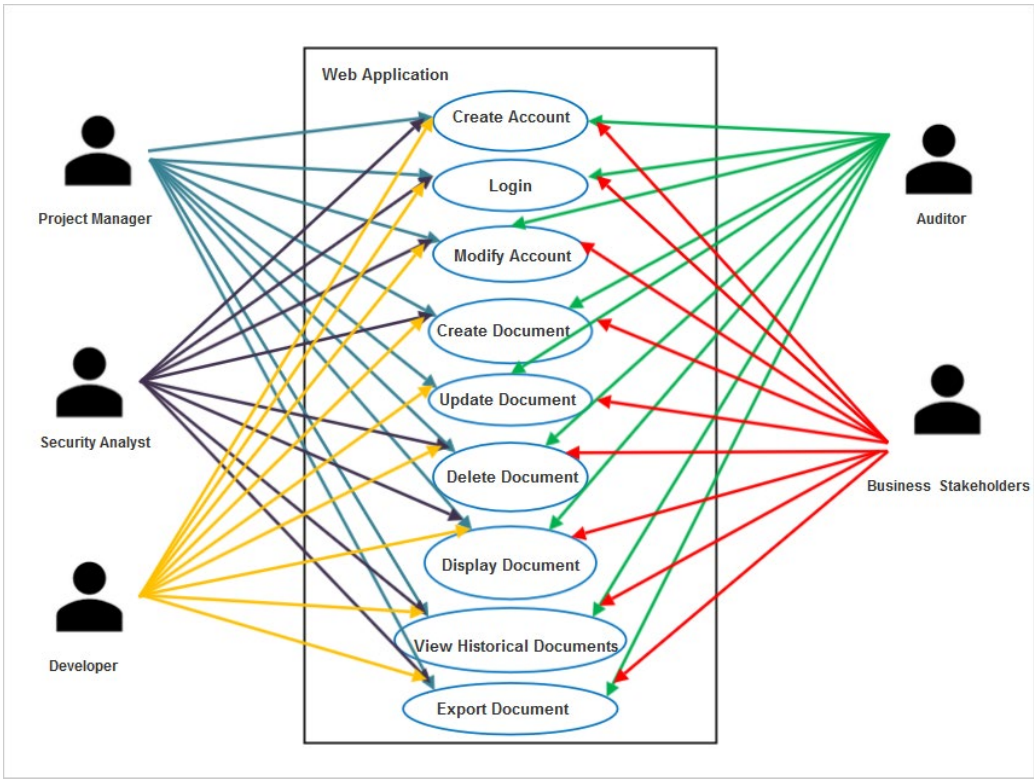


Figure 1: Use Case Diagram

Project Management

Budget

Table 2 represents our budget and reflects actual and estimated costs of labor wages. Actual wage costs are $0 as we are developing the application for no wages. Estimated wage costs reflect paid wages for a developer.

CATEGORY	ITEM	DESCRIPTION	EXPECTED COST	ACTUAL COST
Labor	Actual Wage Costs	The actual wages for the development of the project	$0	TBD
	Wage Costs Post-Production	Estimated wage costs if the project was paid labor. ($20/hr per developer, 3-member team)	$8,000	TBD
TOTALS			$8,000	TBD

Table 2: Budget

Objectives/Deliverables

Our objective is to develop a web application that will provide an automated solution for DevSecOps documentation specific to the business's security requirements in an agile environment. This will provide a clear guideline for the DevSecOps process in an agile environment and resources that provides information in implementing DevSecOps during software development. The user interface will be fluid, responsive and easy for the user to create an account and begin to work on their security metrics quickly. As any software project may have deadlines, users will only need to input their metrics based off a provided template and once complete, the application will autogenerate a document that the user can export to share with others.

Table 3: Major Project Milestones: This shows our major project milestones deliverables that include both Fall 2019 and Spring 2019 milestones that cover our development process.

MAJOR PROJECT MILESTONES (DELIVERABLES)			
FALL OF 2019 MILESTONES			
Initiation Phase	08/26/19	Team Contract Written	09/09/19
Research Phase	09/09/19	Project Abstract Drafted	09/23/19
Design Phase	09/20/19	User Profile Drafted	10/07/19
Implementation Phase	10/07/19	Elevator Speech	10/21/19
Security Testing	01/13/20	Fall Presentation	11/25/19
SPRING OF 2020 MILESTONES			
Alpha Test Phase	01/13/20	Spring Presentation	03/30/20
Deployment Phase	01/20/20	Tech Expo	04/13/20
Security Testing	02/03/20		
Beta Test Phase	02/03/19		
Redesign/Software Update Phase	03/2/19		

Table 3: Major Project Milestones

Project Schedule and Gantt Chart

Table 4 Project Schedule consists of our project schedule with major tasks and milestones listed.

WBS NUMBER	TASK NAME	START DATE	DUE DATE	DURATION
1	**Project Management & Deliverables**	**8/26/19**	**4/13/20**	**231**
1.1	Team Building	8/26/19	9/2/19	7
1.2	Ideas and Brainstorming	9/2/19	9/9/19	7
1.3	Fall Semester Assignment 0: Team Members & Project Name	9/2/19	9/9/19	7
1.4	Fall Semester Assignment 1: Team Contract	9/9/19	9/23/19	14
1.4.1	Project Approval	9/9/19	9/23/19	14
1.4.2	Gantt Chart & Work Breakdown Structure	9/9/19	9/23/19	14
1.5	Fall Semester Assignment 2: Project Abstract for Tech Expo	9/23/19	10/14/19	21
1.6	Fall Semester Assignment 3: Team Contract Resubmission	10/7/19	10/14/19	7
1.7	Fall Semester Assignment 4: User Profile	10/7/19	10/21/19	14
1.8	Fall Semester Assignment 5: Use Case Diagram	10/7/19	10/21/19	14

1.9	Fall Semester Assignment 6: Draft Report	10/21/19	11/4/19	14
1.1	Fall Semester Assignment 7: Final Fall Semester Report	11/4/19	11/25/19	21
1.11	Fall Semester Oral Presentation	11/4/19	11/18/19	14
1.11.1	Presentation Practice	11/4/19	11/18/19	14
1.12	Spring Semester Assignment 1: Testing Plan/Report	1/13/20	2/10/20	28
1.13	Spring Semester Assignment 2: Abstract	2/10/20	2/17/20	7
1.14	Spring Semester Assignment 3: Draft Tech Expo Poster	2/17/20	3/2/20	14
1.15	Spring Semester Assignment 4: Final Poster	3/2/20	3/16/20	14
1.16	Spring Semester Oral Presentation	3/16/20	3/30/20	14
1.16.1	Presentation Practice	3/16/20	3/30/20	14
1.17	IT Expo Preparations	3/30/20	4/13/20	14
1.17.1	IT Expo	3/30/20	4/14/20	15
1.18	Spring Semester Assignment 5: Final Report	3/30/20	4/13/20	14
1.19	Spring Semester Assignment 6: Safe Assign Final Report	3/30/20	4/13/20	14
1.2	Spring Semester Assignment 7: Final Library Copy	4/13/20	4/27/20	14
2	**Research**	**9/9/19**	**10/21/19**	**42**
2.1	Software Requirements	9/9/19	10/14/19	35
2.1.1	Determine Front End Development Languages	9/9/19	9/30/19	21
2.1.2	Determine Back End Development Languages	9/9/19	9/30/19	14
2.1.3	Notification/Alerting System	9/30/19	10/7/19	7
2.1.4	Determine Automation Development Languages	10/7/19	10/14/19	7
2.2	Network Requirements	9/9/19	10/14/19	35
2.2.1	Determine Hosting Environment	9/9/19	10/14/19	35
2.2.2	Determine Database Environment	9/9/19	10/14/19	35
2.2.3	Research IP Mapping	9/9/19	10/14/19	35
2.2.4	Research Load Balancing Systems	9/9/19	10/14/19	35
2.3	Security Requirements	9/9/19	10/21/19	42
2.3.1	Research Security Aspects of Source Code	9/9/19	10/21/19	42
2.3.2	Research Comprehensive Protection in Production	9/9/19	10/21/19	42
2.3.3	Research Security Aspects of Automation	9/9/19	10/21/19	42
2.3.4	Research Security Aspects of CI/CD	9/9/19	10/21/19	42
2.3.5	Research Legal Disclaimers	9/9/19	10/21/19	42
2.4	Miscellaneous Research	9/16/19	10/21/19	35
2.4.1	Determine Features and Mockup Lists	9/16/19	9/30/19	14

2.4.2	Budget Analysis	9/30/19	10/14/19	14
3	**System Design**	**9/16/19**	**10/28/19**	**42**
3.1	Create System Diagrams	9/16/19	10/21/19	35
3.1.1	Create Network Diagrams	9/16/19	9/23/19	7
3.1.2	Create Database Diagrams	9/16/19	9/30/19	14
3.1.3	Create Interaction Diagrams	9/16/19	10/7/19	21
3.1.4	Create Wireframe Diagrams	9/30/19	10/21/19	21
3.2	Create Legal Documentation	10/7/19	10/28/19	21
3.2.1	Draft Legal Disclaimers and Privacy Policy	10/7/19	10/28/19	21
4	**Environment Set-Up**	**9/23/19**	**2/3/20**	**133**
4.1	Import Libraries for Development	9/23/19	9/30/19	7
4.2	Setup GitHub	9/23/19	9/30/19	7
4.3	Setup CI/CD Environment	9/23/19	9/30/19	7
4.5	Setup Front End Framework	9/23/19	9/30/19	7
4.6	Setup Google Firestore	10/7/19	11/4/19	28
4.6.1	Write Database Creation Scripts	10/7/19	10/21/19	14
4.6.2	Link Database Tables	10/21/19	11/4/19	14
4.6.3	Configure User Groups and Access/Roles	10/21/19	11/4/19	14
4.6.4	Implement Audit Logging	10/21/19	11/4/19	14
4.7	Setup Web Application Servers	10/28/19	11/18/19	21
4.8	Setup Web Application Firewall	11/4/19	11/25/19	21
4.8.1	Create Firewall Rules	11/4/19	11/25/19	21
4.9	Setup File Share Server	11/4/19	11/25/19	21
4.1	Setup Load Balancer	1/13/19	1/27/19	14
4.11	Purchase Domain Name	1/20/19	2/3/19	14
4.11.1	Install Site Certificates	1/27/19	2/3/19	7
5	**Development**	**10/7/19**	**3/30/20**	**175**
5.1	Create Main Angular Module and Landing Pages	10/7/19	10/21/19	14
5.2	Create and Configure Angular Routing Module	10/7/19	10/21/19	14
5.3	Create Navigation Bar	10/14/19	10/28/19	14
5.4	Setup API Routers	10/14/19	11/4/19	21
5.5	Create Login and Setup Authentication	10/14/19	11/4/19	21
5.5.1	Create User Registration and Confirmation	10/14/19	11/4/19	21
5.5.2	Create Forgot Password	10/21/19	11/4/19	14
5.6	Develop Features	10/21/19	2/24/20	126
5.6.1	Design and Develop AutoDSO Overview Page	10/21/19	10/28/19	7
5.6.2	Design and Develop DevSecOps Overview Page	10/28/19	11/4/19	7

5.6.3	Design and Develop DevSecOps Best Practices Page	11/4/19	11/18/19	14
5.6.4	Design and Develop Helpful Links Page	11/18/19	11/25/19	7
5.6.5	Design and Develop Security Assessment	11/25/19	12/16/19	21
5.6.6	Design and Develop Security Document Automation	12/16/19	1/6/20	21
5.6.7	Design and Develop Document Management	1/6/20	1/27/20	21
5.6.8	QA Fix Application Bugs	1/27/20	2/24/20	28
5.7	Release Bugfixes	2/24/20	3/30/20	35
5.7.1	Patches for Regression Testing	2/24/20	3/9/20	14
5.7.2	Patches for User Acceptance Testing	3/9/20	3/30/20	21
6	**Testing**	**1/13/20**	**3/30/20**	**77**
6.1	Set Up Mock Clients	1/13/20	2/10/20	28
6.1.1	Setup Desktop Clients	1/13/20	1/27/20	14
6.1.2	Setup Mobile Clients	1/27/20	2/10/20	14
6.2	Full Regression Test	2/17/20	3/2/20	14
6.2.1	Regression Test for Client	2/17/20	3/2/20	14
6.2.2	Regression Test for Administrator	2/17/20	3/2/20	14
6.3	Security Test	2/3/20	3/16/20	42
6.3.1	Firewall Test	2/3/20	2/10/20	7
6.3.2	Penetration Test	2/10/20	2/17/20	7
6.3.3	SQL Injection Test	2/17/20	2/24/20	7
6.3.4	Source Code Test	3/2/20	3/9/20	7
6.3.5	Comprehensive Protection in Production Test	3/9/20	3/16/20	7
6.3.6	Automation Test	2/3/20	2/24/20	21
6.3.7	CI/CD Test	2/24/20	3/16/20	21
6.4	User Acceptance Test	3/9/20	3/23/20	14
6.4.1	User Acceptance Test for Client	3/9/20	3/23/20	14
6.4.2	User Acceptance Test for Administrator	3/9/20	3/23/20	14
6.5	Final Certification Test	3/23/20	3/30/20	7

Table 4: Project Schedule

Figure 2: Gantt Chart highlights the timeline from the project schedule.

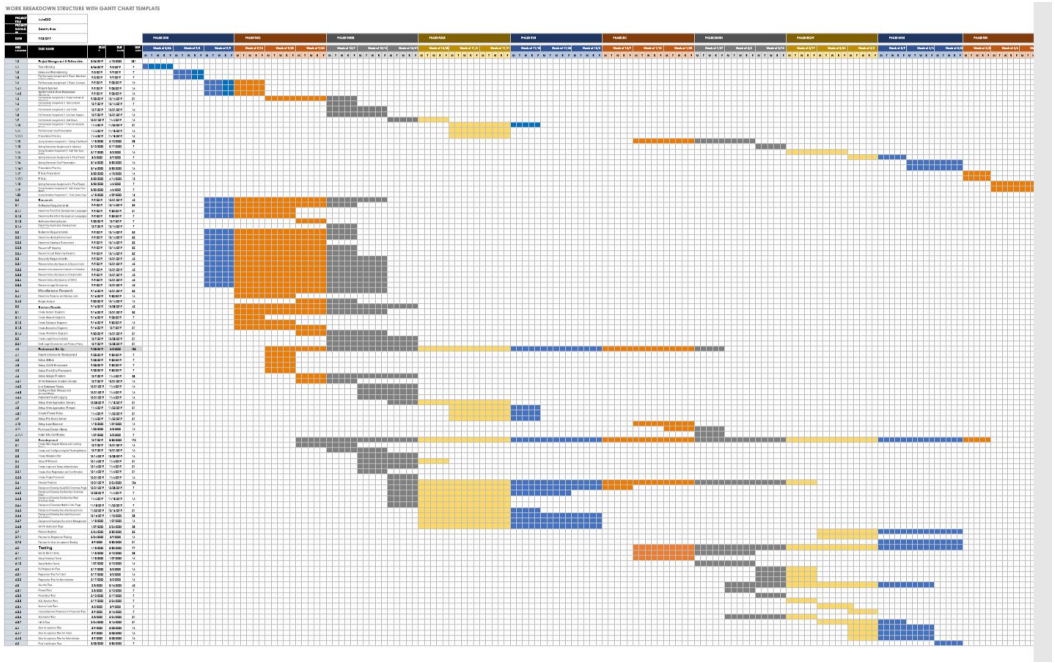


Figure 2: Gantt Chart

Technical Elements

Network

Using Google Cloud Computing Platform was an ideal solution due to its cost effectiveness and scalability. This account is mainly managed by the team's software developer, but admin access has been delegated out to the other two team members. By late January, our hope is to have purchased a domain name to create as this site should be active and available for businesses and students alike to use by the end of the spring semester. This will also help with professional appearance.

Application

AutoDSO will be build using a variation of the MEAN stack. This leverages two open source, lightweight NodeJS technologies to build a full stack application. For Front-End development we will be leveraging Angular8, a leading framework that uses a modular type build that allows for code reusability and maintainability. For our back end we have

decided to use ExpressJS. The decision to use ExpressJS over another NodeJS library is too due to its open availability, large user base, and ample documentation which is useful for adoption and maintainability. This also enables allows us to make this application extremely scalable as we do not have a need for multiple repositories or code management systems, we can create one Mono Repository using NX, ensuring that all our code stays in one place. This is beneficial because as the application grows bigger in the future there will be no need to try to connect the dots between two different applications, as you would if you had used a Java backend for instance. Overall, this applications technology will allow it to remain extremely maintainable over long period of time.

Database

AutoDSO will be using a NoSQL MongoDB database for its data storage. The nature of this applications data is non-relational, which pairs with a NoSQL Database. This means as our application grows and changes the database can grow, mutate, and scale with it, unlike with a traditional relational database where its structure is rigid and inflexible. Our database will be designed specifically to be able to take and store differing data types to accommodate each company's differing security metric requirements. Our goal is to have a database that allows for quick and seamless automated document generation.

Technical Architecture

Application Architecture

The previous section described the technical elements of AutoDSO and how the components of our application architecture focus upon scalability, performance and ease of use. Therefore, we chose to implement a variation of the MEAN stack. These technologies include: Angular8, NodeJS, ExpressJS, Typescript and MongoDB. Figure 3: Technical Architecture Diagram displays a visualization of how these technologies were combined to create AutoDSO web application.

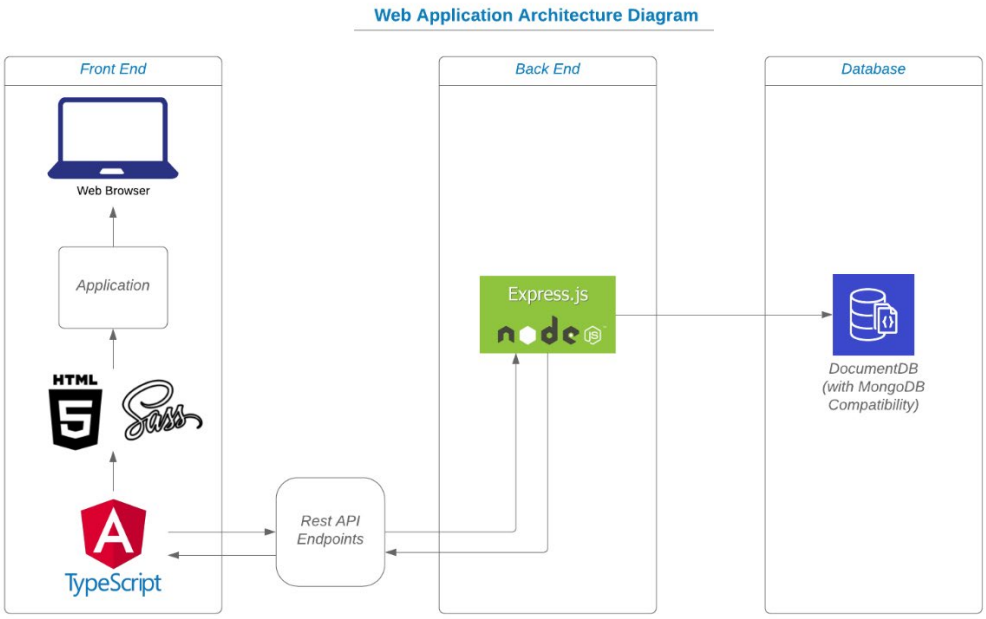


Figure 3: Technical Architecture Diagram

User Interface

Home View

Figure 4 Home View: The home view is being built with MDBootstrap components. It will maintain a simple and clean look and soon will have additions to it to provide a brief description of what the app does to entice users. There will be visuals available of documents to catch the users eye using svgs. Each proceeding page will have a similar design with blue, white, and grey color scheme.

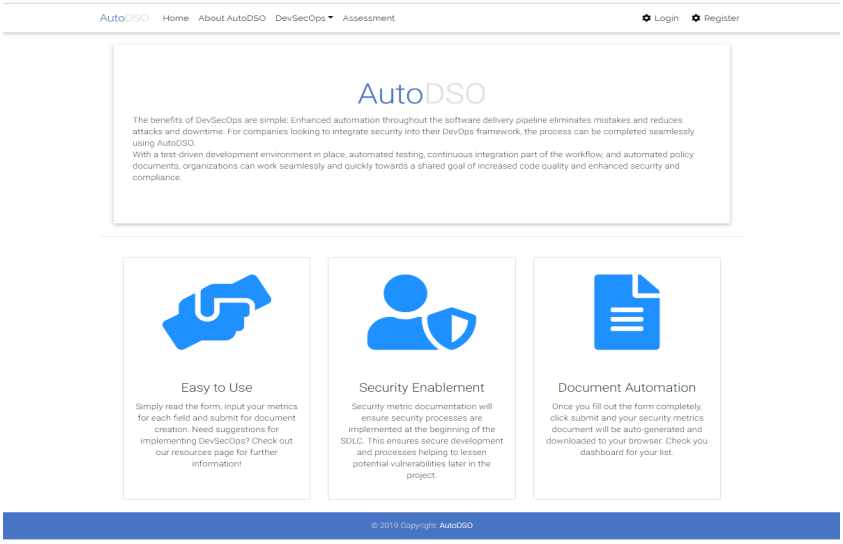


Figure 4: Home View

Login View

Figure 5 Login View: This shows where users can log in to manage their documents. This page is separate from most of the application and serves as a security gateway between the user and user protected information. It will remain simplistic and contemporary in design.

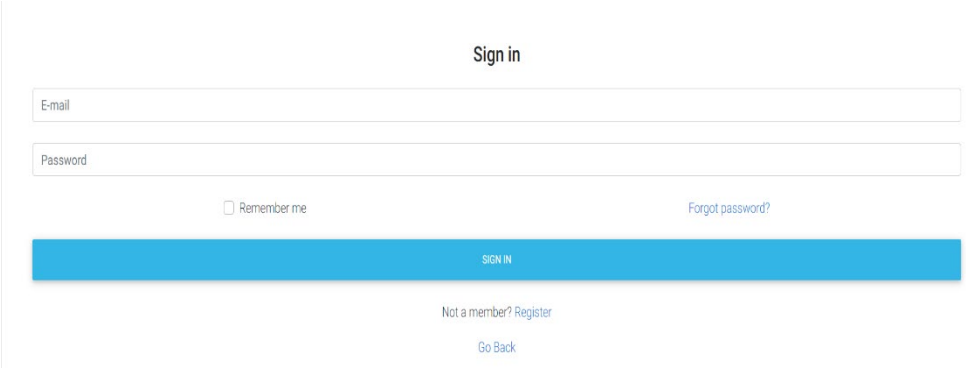


Figure 5: Login View

Registration View

Figure 6 Registration View: This shows where users can register for an account with AutoDSO. Registration is required to use our Assessment functionality.

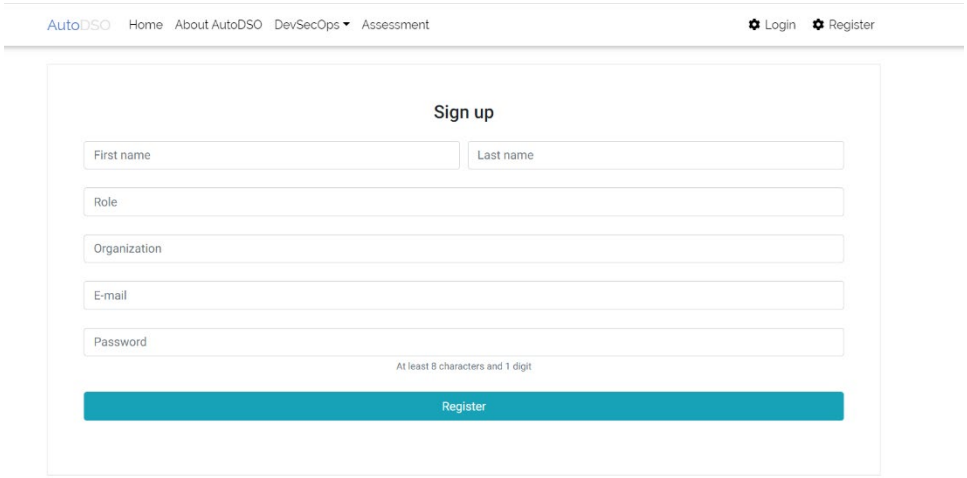


Figure 6: Registration View

Dashboard View

Figure 7 Dashboard View: Here you can find the initial page users are directed to upon logging in. This section will provide access to their historical assessment records as well as basic information provided at registration.

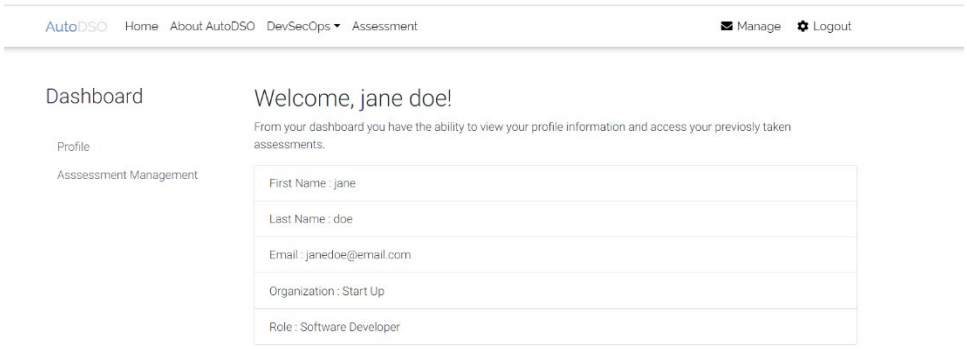


Figure 7: Dashboard View

Assessment Management View

Figure 8 Assessment Management View: This page will allow you to download previous assessments taken as well as get basic information on when the assessment was taken. You will also be redirected here upon finishing the assessment.

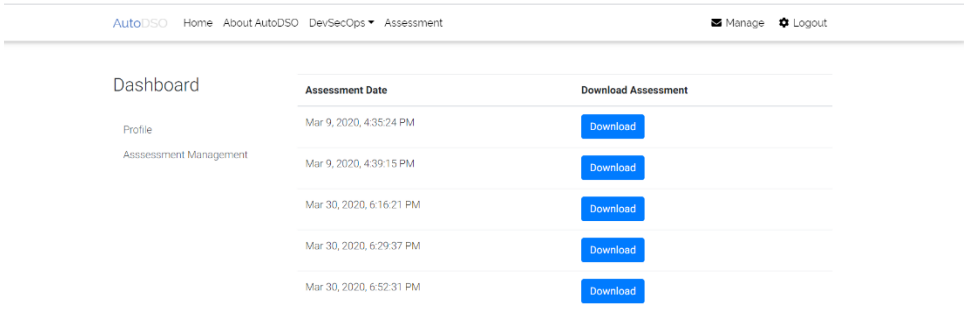


Figure 8: Assessment Management View

Assessment View

Figure 9 Assessment View 1 & Figure 10 Assessment View 2: Here in Figure 9 the use will find themselves on a page that provides a brief explanation of what our assessment hopes to accomplish. Once they click the “Take Assessment” button, they will be redirected to Figure 10 where they must complete all applicable questions and submit for the Security Metric Document to be generated and downloaded to their browsers.

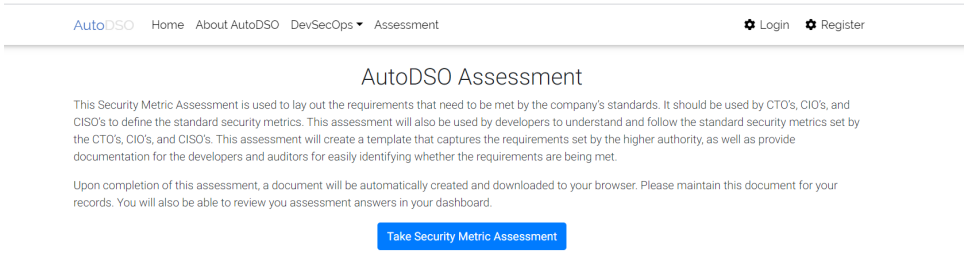


Figure 9: Assessment View 1

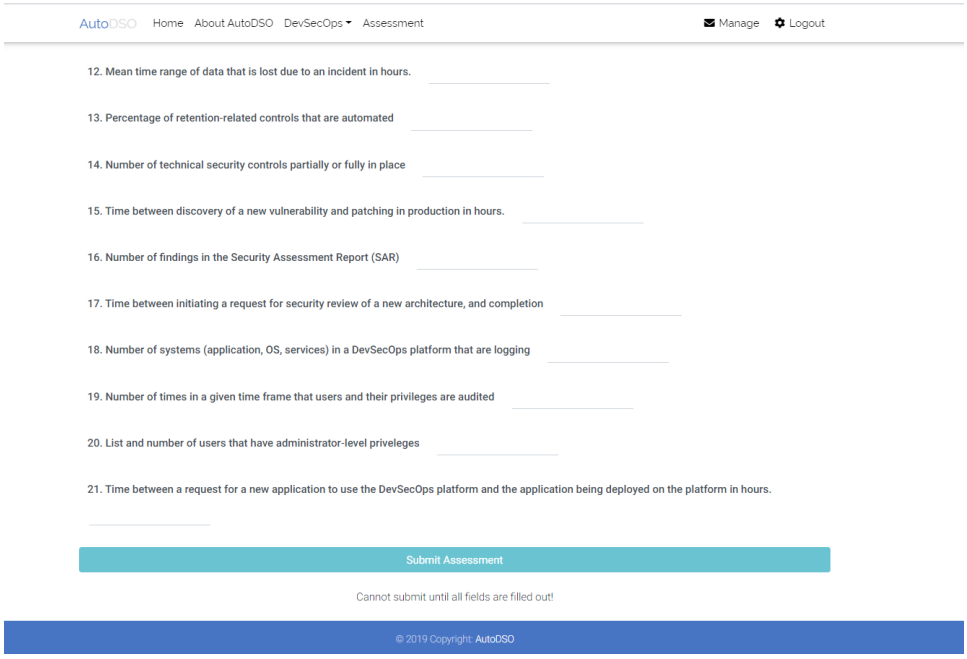


Figure 10: Assessment View 2

Project Testing

Testing Overview

This section covers in detail information regarding test methods used for the functionality of the AutoDSO web application in order to ensure the application works as intended. We chose to test each section of our web application individually in order to ensure that each part of our web application works as intended. If we test each section of the web application separately and each of those sections work correctly, we know that the web application will function as a whole.

Testing Objective

The goal of testing AutoDSO web application is to determine user interface functionality is running as predicted and the functionality of the web application is working correctly. During the testing phase, each section is tested individually. User interface and functionality testing needs to be tested because we want the end-users to be able to easily use our web application without running into any interface problems and should not run into delays.

Testing Scope

The scope of this testing will cover all the major features that are being built into the AutoDSO web application. Our test will include testing for user interaction with the application including navigating the menu's and the web application for content. It will also include functionality and performance testing to ensure the application is running as it should, and test input validation for input fields. This will ensure security of the web application.

Testing Procedures

The following is required to begin application testing

Establish pass/fail conditions

Create a table to record all test attempts and results

Test that will be performed:

Overview

User Interface Test

This test involves different interactions a user would follow when using the AutoDSO application

Functionality Test

This test involves testing that every section of the application functions correctly.

Testing Scenarios

The testing scenarios demonstrates how each test listed in the Testing procedures were conducted.

User Interface Test

The User Interface Test will check to ensure that the User Interface is free of bugs and errors, since the user of AutoDSO does not have any knowledge about the web application with their first use. This test will be conducted manually and help to see how easy it is for the user to understand the navigation of the web application. It will include the menus, links and performance of the web application. Questions that will be answered in this scenario include:

1. Are there any noticeable performance issues when navigating the menus?
2. Are the menu links working properly and linked correctly to the pages?
3. Are error messages displayed correctly?
4. Are the fonts used on the web application readable?
5. Do the images have good clarity?
6. How long does it take for a page to load?
    a. Anytime under 3 seconds is considered an acceptable parameter. If it is over 3 seconds, it'll be displayed in red.
7. Does it take a long time to go through the security assessment?
8. Does the import/export functionality work properly for the security metrics document?
9. Is the security assessment easy to follow along and fill out?

Functionality Test

The functionality test will check the main functions of the web application. This test will test usage techniques to check for error conditions and basic usability functionality of whether the user can login into the web application and navigate the application without any difficulties. Test scenarios include:

1. *Check Login Functionality*
    a. Check response on entering a valid Username and Password
    b. Check response on entering an invalid Username and Password
    c. Check response when Username field is empty and Login button is clicked
    d. Check response when Password field is empty and Login button is clicked
    e. Test response on entering valid length of characters entered for Username and Password
    f. Check redirect response on login
2. *Check New User Registration Login Functionality*
    a. Check response on entering a valid Username, Password, First Name, Last Name, Role, and Corporation
    b. Check response on entering an invalid Username, Password, First Name, Last Name, Role, and Corporation
    c. Check response when Username field is empty and registration button is clicked
    d. Check response when Password field is empty and registration button is clicked
    e. Check response when First Name field is empty and registration button is clicked
    f. Check response when Last Name field is empty and registration button is clicked
    g. Check response when Role field is empty and registration button is clicked
    h. Check response when Corporation field is empty and registration button is clicked
3. *Check Assessment Functionality*

a. Check assessment availability on login
b. Check redirect functionality if not logged in when attempting to take assessment
c. Check response when all mandatory assessment fields are not completed on submit
d. Check response when all mandatory assessment field are completed on submit

4. *Check User Input Functionality*

a. Test response on entering valid length of characters entered for Username and Password during login
b. Test response on entering invalid length of characters entered for Username and Password during login
c. Test response on entering invalid special characters for Username and Password during login
d. Test response on entering valid length of characters entered for Username, Password, First Name, Last Name, Role, and Corporation during registration
e. Test response on entering invalid length of characters entered for Username, Password, First Name, Last Name, Role, and Corporation during registration
f. Test response on entering invalid special characters for Username, Password, First Name, Last Name, Role, and Corporation during registration
g. Test response on entering valid character for assessment questions
h. Check response on entering valid assessment responses
i. Check response on entering invalid assessment response

Table 2: Functionality Test Results, displays the test scenario, test case description, test results, performance issues and the test data used for each Functionality test conducted. Test scenarios include the Login functionality. Under the Test Results column a green check mark means the test passed and a red X means it did not. **Note: Test Data below is a sample of how our test results will be displayed, for final Test Results**

Check Login Functionality			
Test Case	Test Results	Performance Issues	Test Data
Check response on entering a valid Username and Password	✓	None	Username: user1 Password: Anderson
Check response on entering an invalid Username and Password	✓	None	Username: AB Password: Mapper!
Check response when Username field is empty and Login button is clicked.	✓	Yes	Username: Password: Mapp##
Check response when Password field is empty and Login button is clicked	✓	Yes	Username: AndersonP Password:
Test response on entering valid length of characters entered for Username and Password	✓	None	Username: user1 Password: Anderson
Check redirect response on login	✓	None	Success
Check New User Registration Functionality			
Test Case	Test Results	Performance Issues	Test Data
Check response on entering a valid Username, Password, First Name, Last Name, Role, and Corporation	✓	None	Username: bjohnson Password: Mapper! First Name: Bill Last Name: Johnson Role: Senior Software Developer Corporation: XYZ Company
Check response on entering an invalid Username, Password, First Name, Last Name, Role, and Corporation	✓	None	Username: AB Password: Mapper! -- First Name: B Last Name: J Role: Dev Corporation: X

Check response when Username field is empty and registration button is clicked	✓	None	Username: Password: BobI$Great
Check response when Password field is empty and registration button is clicked	✓	None	Username: bjohnson Password:
Check response when First Name field is empty and registration button is clicked	✓	None	First Name:
Check response when Last Name field is empty and registration button is clicked	✓	None	Last Name:
Check response when Role field is empty and registration button is clicked	✓	None	Role:
Check response when Corporation field is empty and registration button is clicked	✓	None	Corporation:

Check Assessment Functionality			
Test Case	Test Results	Performance Issues	Test Data
Check assessment availability on login	✓	None	Success
Check redirect functionality if not logged in when attempting to take assessment	✓	None	Success
Check response when all mandatory assessment fields are not completed on submit	✓	None	Success
Check response when all mandatory assessment field are completed on submit	✓	None	Success

Check User Input Functionality			
Test Case	Test Results	Performance Issues	Test Data
Test response on entering valid length of characters entered for Username and Password during login	✓	None	Username: bjohnson Password: Mapper!
Test response on entering invalid length of	✓	None	Username: A Password: B

characters entered for Username and Password during login			
Test response on entering invalid special characters for Username and Password during login	✓	None	Username: (}\ PW i Password: += PW
Test response on entering valid length of characters entered for Username, Password, First Name, Last Name, Role, and Corporation during registration	✓	None	Username: bjohnson Password: Mapper! First Name: Bill Last Name: Johnson Role: Senior Software Developer Corporation: XYZ Company
Test response on entering invalid length of characters entered for Username, Password, First Name, Last Name, Role, and Corporation during registration	✓	None	Username: bj Password: M! First Name: B Last Name: J Role: Dev Corporation:
Test response on entering invalid special characters for Username, Password, First Name, Last Name, Role, and Corporation during registration	✓	None	Username: b+anders Password: ++Map! First Name: B+ Last Name: J= Role: =Senior Corporation: ((XYZ))
Test response on entering valid character for assessment questions	✓	None	Success
Check response on entering valid assessment responses	✓	None	Success
Check response on entering invalid assessment responses	✓	None	Success

Table 2: Functionality Test Results

Problems Encountered

The only problems encountered were needing to make technology changes. We switched from NestJS to ExpressJs. The reason being due to the lack of documentation available on NestJS which caused an unprecedented learning curve that wasn't accounted for initially. ExpressJS is a tried and tested compatible backend language that was capable of supporting all of our development needs and was highly compatible with Angular 8 despite its differences in architecture.

Future Recommendations

AutoDSO currently in our view a working product that is useful for any organization looking to get a start in documenting their security metrics. As we completed our project there are numerous areas we have learned that we could potentially add to enhance our web application. If there was more time, we would add the ability to provide a questionnaire for our users to determine their specific needs. Currently AutoDSO covers an extensive list of security metrics an organization may document, but it appears geared towards those organizations that are large in size. We envision in expanding AutoDSO, to provide security metrics that a mid to small size organization may use. This could be done by providing security metrics that cover smaller project sizes and certain barebone security metrics rather than an extensive list.

We've gotten suggestions from professionals that ideally it would be useful to implement a color code system tracking how organizations have either met or not met their goals in their security metric document. Their history list would show green when their security metrics document has continued to meet their initial metrics. For example, if an organization decided that it should take 1 day to fix code vulnerabilities found, and throughout the year, they continue to meet this goal upon updating their security metrics, it would stay green. If an organization has found that it takes 3 days to fix code vulnerabilities then their updated document would display a red arrow next to it, displaying they had to increase their time. The idea behind this is that a C-level executive could see at a high level whether their organization is meeting their goals or not.

Another addition for the future would be to allow the ability to modify an existing security metric document. Currently the user can only create one and would need to create a new one if they wanted to update. Due to time, this was the only function we could implement.

Lastly, we haven't made a concrete decision as to whether we would continue to enhance AutoDSO and promote it to organizations. Upon speaking with professionals this automation of a security metric is a needed application. I believe any of our team members could continue to work on enhancing the application and promoting it, if that was a goal.

Conclusion

Fall Semester 2019

The Homepage was completed and we finished the template for the security document will be at the end of the fall semester. During this semester a lot of research was completed regarding the DevSecOps process and concluding exactly how our application would be beneficial. Once we grasped our concept, the technical elements were determined that would best help our vision succeed.

We worked on completing the design for the features of AutoDSO, finishing the development of the security assessment, document automation, and providing testing for the Spring Semester.

Spring Semester 2020

Spring semester our team worked on the testing phase, this included a full regression test. We finalized the completion of the application, and fixed any issues found during testing in order to ensure it was corrected. Our team completed the full user-interface for the web application as well as fixed a bug found early on in our testing phase. Our team completed designing the features for AutoDSO, fully integrating the security assessment, and completed the export of the security metrics document into a Microsoft Word document.

Throughout this semester, our team learned that we must learn how to talk both technical for the people that understand and have background knowledge about DevSecOps. We also learned that we need to be able to speak to people not so tech savvy well enough to where they understand the concepts of what the application's used for as well as just to spread knowledge on DevSecOps. Some of the major things our team learned throughout completion of this project was understanding that DevSecOps is a continuous solution. It will continue to change and improve. This project we believe is a great start to allowing organizations to create security metric documentation for their projects. In our future recommendations section, we discussed how AutoDSO could dive deeper into providing various security metrics dependent on organization size and needs. As DevSecOps continues to improve and more companies adopt the process, our web application will also need to improve and change to make sure we are providing the most up to date DevSecOps processes and security metrics.

Appendix A – References

1. "OWASP DevSecOps Maturity Model," 2019 [Online]. Available: https://www.owasp.org/index.php/OWASP_DevSecOps_Maturity_Model/.
2. "Top 31 Best DevSecOps Tools" 2019 [Online]. Available: https://blog.inedo.com/top-31-best-devsecops-tools/.
3. "SonarQube" [Online]. Available: https://www.sonarqube.org/
4. "DevSecOps Guide" [Online]. Available: https://tech.gsa.gov/guides/dev_sec_ops_guide

Appendix B– Additional Information

The team contacted the following professionals for additional insight and clarification for our DevSecOps inquiries.

1. Bo Vykhovanyuk, University of Cincinnati Information Security Assistant Vice President
2. Kurt Loock, FireEye, Partners-Regional
3. Ryan Hamrick, CBTS
4. Justin Hall, CBTS

Appendix C– Technologies Used

We used various technologies in the creation of AutoDSO. Below list the websites of all the technologies we used to create our web application.

- https://cloud.google.com/
- https://nodejs.org/
- https://angular.io/
- https://nestjs.com/
- https://www.mongodb.com/
- https://github.com/nccgroup/VCG

Appendix D– IT Tech Expo Poster

Figure 11: IT Tech Expo Poster displays the poster we created for the CECH IT Tech Expo 2020.

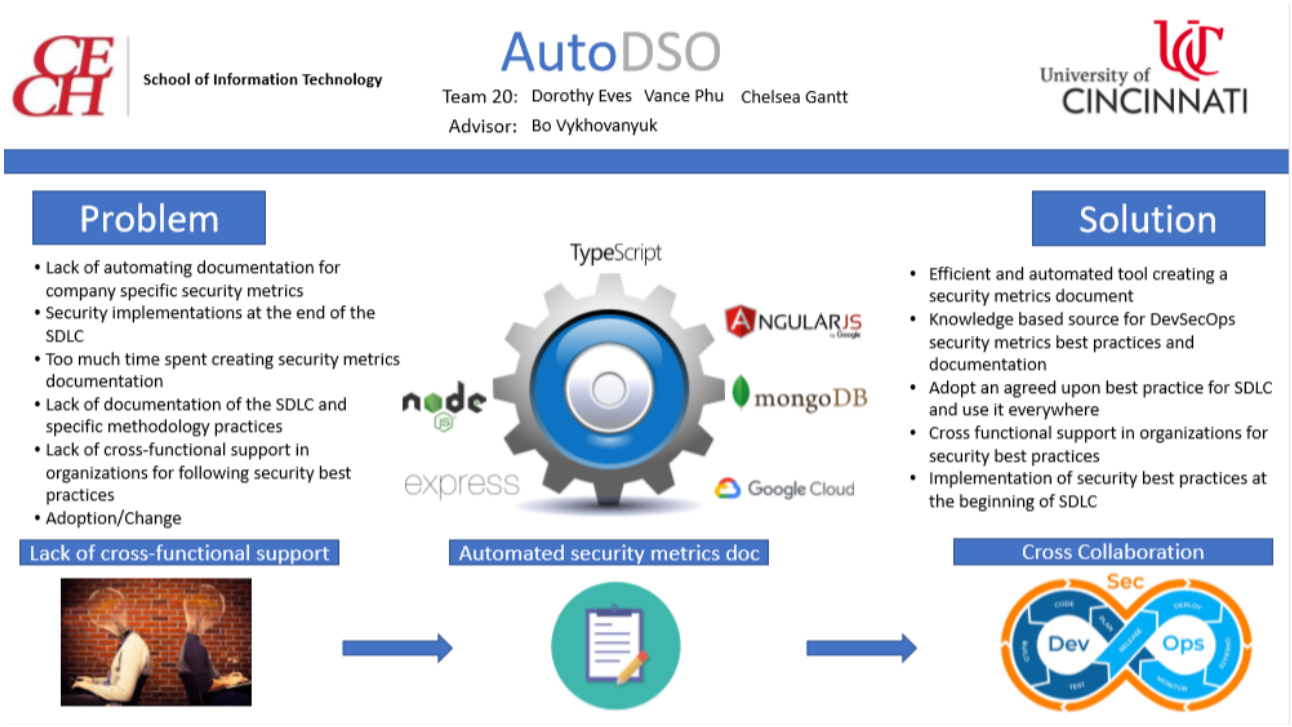


Figure 11: It Tech Expo Poster