

JONATHON BOURKE

C00242865

Functional Specification

CONTENTS

Abstract..... 3

Introduction 4

 Purpose..... 4

 Project Scope..... 4

 Background..... 5

 References..... 5

 Vulnerability Management Tool – Project Brief 5

 General Data Protection Regulation..... 6

 Privacy By Design Methodologies..... 7

 Asumptions and Constraints 7

 Document overview 8

Methodology..... 9

Context..... 10

Interface Requirements 10

 Penetration Testing Dashboard..... 11

 Compliance Dashboard 11

 Administrators Dashboard 12

 Vulnerability Management Queue..... 12

 Ticket Interface..... 13

 Dashboard Interface..... 13

 Widget Interface..... 14

User Requirements 14

 System actors 14

 Sample Use Cases 14

Data Flow DiagramS..... 31

 System Data Flow 32

 Ticket Data Flow 33

Logical Data Model 34

Functional Requirements..... 35

Hardware/Software Requirements..... 36

Operational Requirements 37

Error Handling..... 38

Validation Rules	38
Security and Privacy	38
Security required on the system	39
Reliability.....	39
Project plan	39
Core GUI – Sub Tasks.....	40
Ticket Operations – Sub Tasks.....	40
Dashboard Widget Creation.....	40
Vulnerability Scanner Integration	40
Optional GUI.....	40
Optional Operations.....	41
Bibliography	42

ABSTRACT

Vulnerability Management encompasses the acts that identify, evaluate, remediate, and document security weaknesses in software, hardware, and operating systems. Following good quality vulnerability management practices is vital in reducing organisations attack surface and prioritizing relevant security threats that are constantly discovered.

This document goes into low level detail on how the project brief will be satisfied and implemented. Further, a rough project plan is provided to give a future timeline to complete tasks.

INTRODUCTION

Vulnerability management is the constant cycle of finding, evaluating, and remediating cyber security issues in an environment. This helps an organisation identify and patch issues that arise from changing environments and new vulnerabilities. This can be a challenging prospect especially if acknowledging external factors like resources, sourcing skilled personnel and policy maturity.

PURPOSE

This document will outline how the requirements of the project are planned to be achieved and to give a high-level overview of the components and purposes of the application. The core proposes are included in the project brief.

The tool will capture the attributes of validated vulnerabilities by a penetration tester and manage the process and timeline of each vulnerability within an organisation.

A management interface will be provided to facilitate all aspects of vulnerability management for an organisation.

The tool will facilitate the capture and management of vulnerability data through its lifecycle

A GUI will allow users to store all attributes of vulnerabilities and provide a dashboard displaying the current security status of an organisation.

Additional features will be added if they will be seen as to supplement any of the discussed areas. These will be prioritized based on the potential value they can bring to the system.

PROJECT SCOPE

Implementing efficient policies and procedures for vulnerability management is a challenging prospect for any organisation. The only solution to guarantee refinement and improvement is through monitoring the lifecycle and uncovering weak areas in procedures. The root cause of these weaknesses can be determined and subsequently strengthened through policy changes, internal training or increasing manpower. As a result, SLA times will decrease, and remediation efficiency will improve for the next cycle.

We can look to many areas to supplement the vulnerability management lifecycle. Implementing a system that aids and tracks the process is a logical key step to improve tracking and efficiency. The following are areas of the cycle that a system can supplement:

Determining the scope: Asset tracking and prioritization

Roles and Responsibilities: Defined through users and their permissions

Vulnerability management tool: Track the activities of a dedicated vulnerability scanner

Policies and SLAs: Track remediation time and store policies. Match policies to assets

Find context sources: Collect data through internet parsing on scoped assets. Allow users to maintain a knowledge base within the application

Vulnerability reports: Generate reports through the application through both manual creation and scanning.

Remediation: Track the success of the remediation in progress. Confirm and validate the result. Return the outcome.

Improve Policies and SLAs: Use the metrics gathered by the system to identify weaknesses in the cycle. Update the policies with improvements and monitor performance.

BACKGROUND

This document is being produced to provide a high-level foundation for developing the vulnerability management tool as described in the brief. The project is a core requirement of the final year in Cybercrime and IT security. This functional specification is a prerequisite for beginning development of the product.

REFERENCES

Below is a list of controlling documents for the project to abide by or for applying guidelines to how development should be regulated:

VULNERABILITY MANAGEMENT TOOL – PROJECT BRIEF

Brief Description. A tool that captures the attributes of validated vulnerabilities by a penetration tester and manages the process and timeline of each vulnerability within an organisation.

Technologies

- (1) Windows or Linux
- (2) C#, C++, Python
- (3) Github

Full Description

Mandatory

Application Area Identified: Provide a management interface to facilitate all aspects of vulnerability management for an organisation.

Data Reading: The capture and management of vulnerability data through its lifecycle.

Display: A GUI that allows users to store all attributes of vulnerabilities and provide a dashboard displaying the current security status of an organisation.

Discretionary

Portable: App runs across multiple platforms (Android, PC, Mac);

Management: System contains management software to allow for multiple users.

Extraction: Relevant and useful information is extracted from the raw data and shown in a user-friendly manner.

Exceptional

Releasable: App is of releasable quality.

GENERAL DATA PROTECTION REGULATION

The GDPR is the current regulation on data privacy in Europe enforced as of 25th May 2018. This legislation aims to improve an individual's control over the rights and use of their personal data. The regulation also aims to improve harmonization across Europe around data protection.

Article 6 of the GDPR defines when it is lawful to process this data:

- Consent has been received from the individual
- To fulfil an agreement with the individual
- Compliance with legal requirements
- To protect the vital concerns of an individual
- If requested by an authority or in the interest of the public
- For the interests of the data controller unless contradicted by the data subject's interests.

The data subject is the individual who is can be identified by the data. Under the GDPR, controls are given to the data subject regarding their personal data in the following areas

- Access to their personal information and how it is processed
- Rights to be forgotten
- Objection in the event data is processed for marketing or reasons outside the scope of the agreement
- Rectification of any inaccurate data

The GDPR defines the roles of Controllers and Processors. The controller establishes the objectives for which and the methods by which personal data is processed. The processor is

defined as a “natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller”. (GDPR, 2021).

Some requirements defined in the GDPR for Controllers and Processors include:

Pseudonymisation: Store personal data in such a way in that it cannot be associated with the data subject without additional information.

Records of processing activities: Most be maintained by organisations who meet one of the following criteria:

- More than 250 employees
- The processed data presents risks to the rights and freedoms of data subjects
- Processing takes place regularly
- Involves special data like what is described in article 9

Controllers and processors are required to use appropriate technical and organisational methods for data protection. Under article 33, controllers are legally required to notify their supervisory authority of breaches of personal data within 72 hours unless it is extremely unlikely to result in damage to the rights or freedoms of its data subjects. The processor is required to make the controller aware if it is they who have become aware of the breach.

PRIVACY BY DESIGN METHODOLOGIES

Approaching development with a plan that factors Privacy by Design methodologies. Documentation like “Privacy by Design: The 7 Foundational Principles” (Cavoukian, 2011) will help identify key areas to ensure this purpose.

1. Proactive not Reactive: Proactive recognition of security issues in development and not uncovering vulnerabilities after release:
2. Privacy as the Default Setting: Techniques like collection limitation, data minimization and retention policies
3. Privacy Embedded into Design: Privacy and security functions should be embedded in the system
4. Full Functionality: All legitimate concerns are found before development.
5. End-to-End Security: Security integrated into the entire data life cycle
6. Visibility and Transparency: Openness to subjects, compliance with regulation, accountability for incidents
7. Respect for User Privacy: Consent should be received before using personal data, openness about activities concerning user data, accuracy of personal data retained.

(Cavoukian, 2011)

ASUMPTIONS AND CONSTRAINTS

Assumptions are conditions that are presumed to be true before development.

- Permission to manage and supervise specific components of organisational devices like open ports, services running and configurations.
- The controlling party must have authority to scan the environment.
- There needs to be a vulnerability scanning tool to integrate with
- A network scanning tool must exist that integrates with the application

Constraints are future challenges beyond the control of the project.

- Development of the application must be completed by: Friday, 29 April 2022
- Tools integrated will be open source and freely distributed
- Personal data must be stored in compliance with relevant regulations

DOCUMENT OVERVIEW

The remaining document is divided into the following main headings:

Methodology: Overall approach used in the determination of the Function Specification contents

Context: Provides a context diagram of the system with explanation.

Interface Requirements: High level overview of some critical interfaces needed in the system.

User Requirements: Provides requirements of the system and its users

Data Flow Diagrams: Provides Data Flow diagrams of the system and its components

Logical data model: Provides a high-level data model of the system

Functional Requirements: Lists the functional Requirements of the system

Hardware/Software Requirements: Determines hardware or software requirements

Operational Requirements: Essential capabilities needed by the system

Error Handling: Briefly describes how the system will manage unexpected events.

Validation Rules: Explains how the system validates data.

Security and Privacy: Consequences of a security/privacy compromise.

Reliability: How system outages will affect its end users.

METHODOLOGY

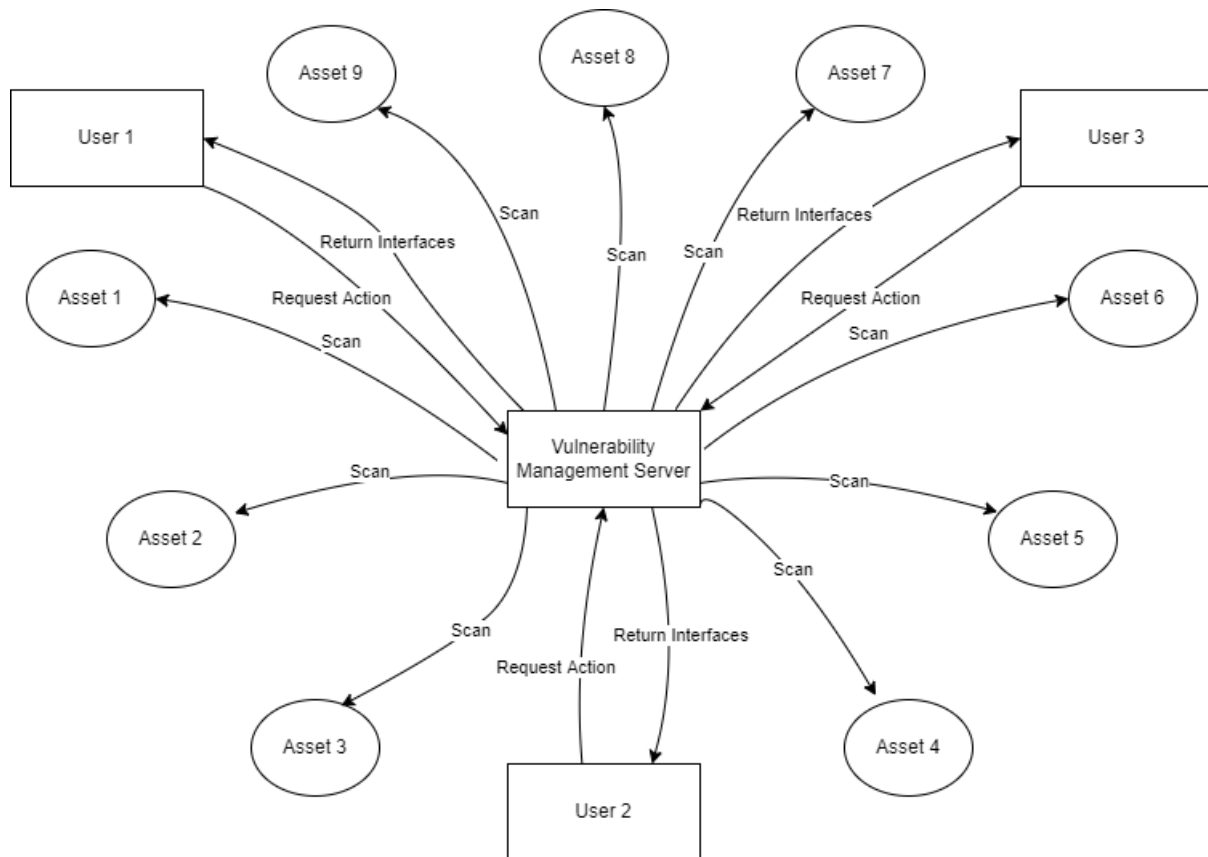
The functional specification was created to incorporate the requirements of the project brief and resolve them as a core prerequisite of the document subject areas. The sections go in depth on these requirements and explain them at a high level for readability across all levels of expertise.

The core requirements are all necessary components to achieve the project goal.

- Capture the attributes of validated vulnerabilities and manage the process and timeline of each vulnerability.
- Interfaces to facilitate all areas of vulnerability management.
- Capturing data from all aspects of the vulnerability management cycle.
- Dashboards that store attributes and visuals of the organisations posture on vulnerability management.
- Providing a system which allows multiple users and is cross platform.

Areas of vulnerability management that can be improved by the system have been defined under the Project Scope.

CONTEXT



The application will be hosted on a centralized server. Assets for the application to manage will be either discovered through probing the environment or manually through asset management. The assets will then be scanned by the application either manually or through scheduled scans. Users will be able to access the system to preform operations depending on their permissions.

INTERFACE REQUIREMENTS

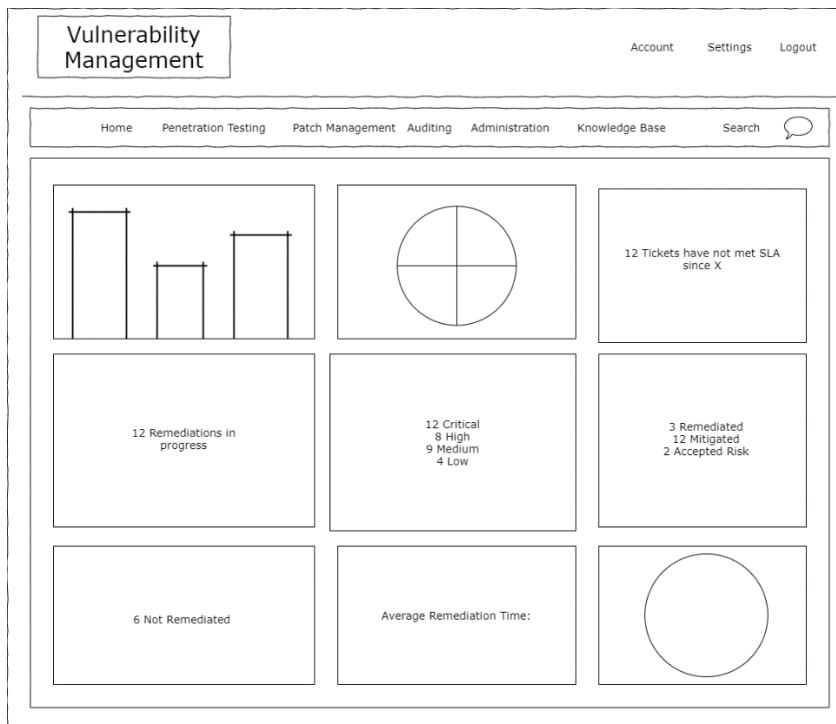
This section will display at a high level some of the critical interfaces that will be needed within the application

PENETRATION TESTING DASHBOARD



Main Penetration Testing Dashboard giving an overview of the relevant information regarding the organisation's vulnerability management stance.

COMPLIANCE DASHBOARD



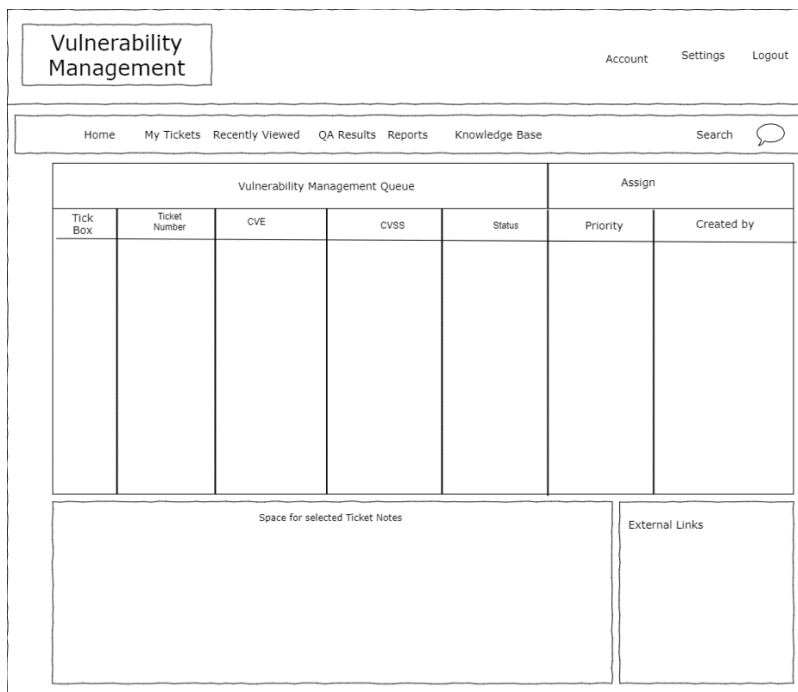
Home dashboard of compliance. This contains user created widgets that have been configured into a dashboard. Allows full control over the metrics displayed.

ADMINISTRATORS DASHBOARD



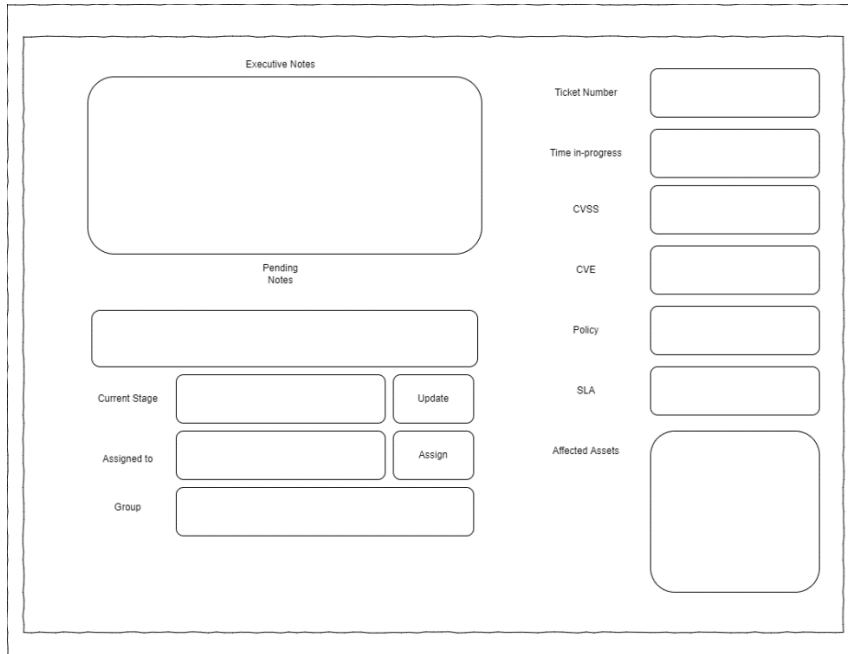
Administrators home dashboard which will give a high-level overview regarding activity within the application

VULNERABILITY MANAGEMENT QUEUE



The vulnerability management ticketing system. This interface will display any unassigned, in-progress tickets that need to be actioned. The user may highlight one to many tickets and assign them as necessary.

TICKET INTERFACE

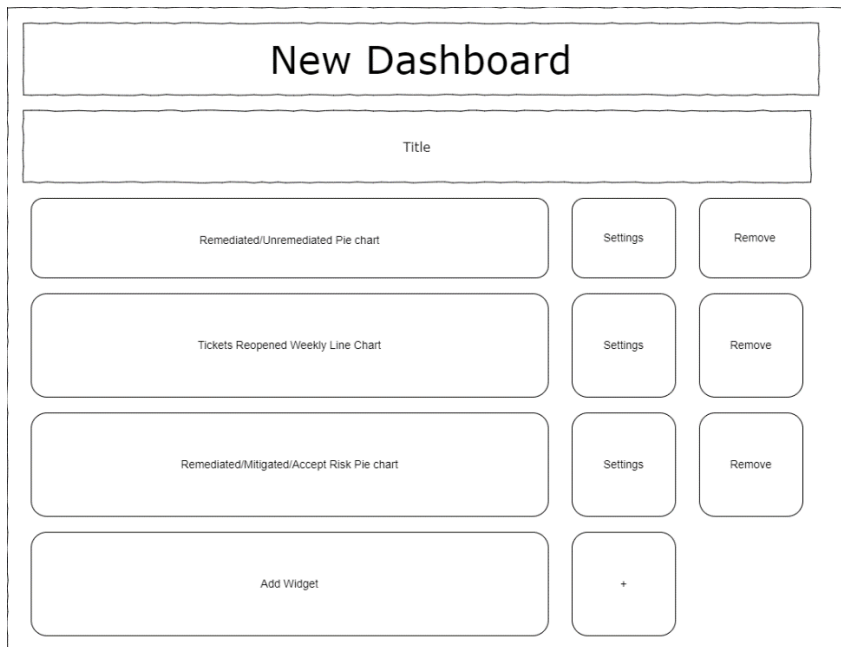


The Ticket Interface form is a structured layout for managing ticket details. It features several input fields and action buttons:

- Executive Notes:** A large rounded rectangular text area at the top left.
- Pending Notes:** A smaller rounded rectangular text area below the executive notes.
- Current Stage:** A text input field with an **Update** button to its right.
- Assigned to:** A text input field with an **Assign** button to its right.
- Group:** A text input field below the assigned to field.
- Metadata Fields:** A vertical column of text labels on the right side, each with a corresponding input field:
 - Ticket Number
 - Time in-progress
 - CVSS
 - CVE
 - Policy
 - SLA
 - Affected Assets (with a larger rounded rectangular input field)

The interface will be displayed to the user when navigating to a ticket. It will contain a mixture of editable and non-editable fields to aid in its completion. Some buttons will give special functionality like update and assign.

DASHBOARD INTERFACE

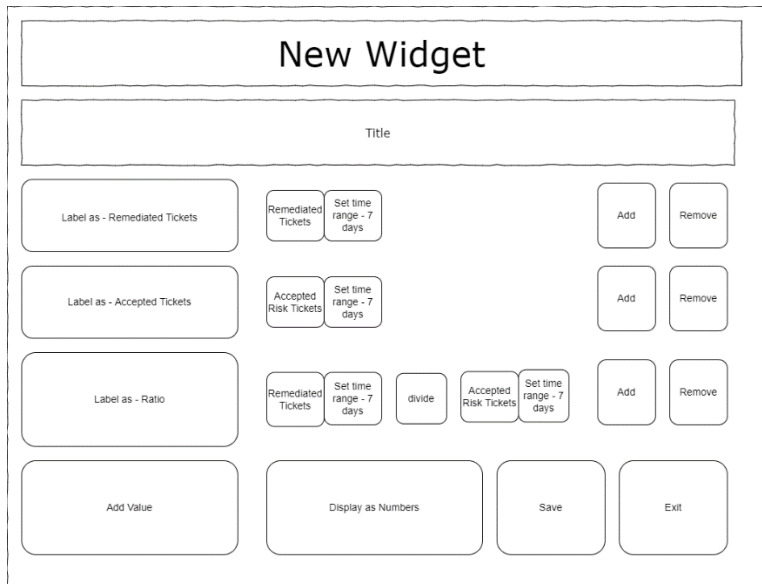


The New Dashboard interface is a grid-based layout for monitoring and managing tickets. It includes a title bar and several widget containers:

- Title:** A header bar at the top.
- Remediated/Unremediated Pie chart:** A widget with a **Settings** button and a **Remove** button.
- Tickets Reopened Weekly Line Chart:** A widget with a **Settings** button and a **Remove** button.
- Remediated/Mitigated/Accept Risk Pie chart:** A widget with a **Settings** button and a **Remove** button.
- Add Widget:** A widget with an **+** button to add new content.

This is the interface that will be seen when the user opts to create a new dashboard. The user may add or remove created widgets and adjust their settings regarding the dashboard creation. Saving the dashboard will add it to a repository for use.

WIDGET INTERFACE



The widget creation dashboard. It allows the user to add values that are calculated and displayed in the chosen format. Once finished, the user may save where the widget is added into a repository for later.

USER REQUIREMENTS

SYSTEM ACTORS

Administrators will have full access and control over the application. They will also be provided additional features that facilitate management over the application and its users. The application will begin with a default admin account which should be subsequently disabled upon configuring the first admin user

Users will have access to the application depending on assigned permissions which can be configured by an admin user. These accounts are intended for regular employees involved within the vulnerability management cycle to be able to perform their function. Least privilege methodologies are recommended.

SAMPLE USE CASES

All Use cases assume that the user is authenticated with the system.

All Use cases assume that the system is not suffering from an availability issue.

All Use cases assume that the user has correct permissions to perform the action

UC - 1	Ticket Creation
Primary Actor(s)	User
Trigger	The user has clicked new record on the vulnerability management ticketing system interface
Pre-conditions	The user is currently on the Vulnerability Management ticketing interface The user has clicked on "New ticket" on the Vulnerability Management.
Post-conditions	A new Ticket will be added to the queue
Main Success Scenario	The user selects new ticket All required fields are entered. Optional fields enrich the ticket The user clicks "Create New Ticket" once finished. The New ticket is assigned to the PVG queue is tagged with appropriate Policy and SLA. The System's stored metric are updated
Extensions	If the User clicks "Cancel Request" on the "New Ticket" interface. The User is asked for confirmation on cancelling the ticket. <i>If the User selects "Yes"</i> The ticket is cancelled The user is returned to the ticketing interface <i>If the User selects "No"</i> The confirmation screen is withdrawn, and no changes are made to the options entered on the ticket request
Priority	High

UC - 2	Initiate Scan
Primary Actor(s)	User

Trigger	The user has confirmed to initiate a scan after setting the scope
Pre-conditions	The user has navigated to the scan interface The user clicks the manual scan button The user has set the scope of the manual scan The user has confirmed their choice and clicked to initiate
Post-conditions	The system parses the selection and creates a custom scan based on the given parameters The scan is initiated, and any found vulnerabilities are returned to an interface and stored for later escalation/metrics.
Main Success Scenario	The user navigates to the scan interface The user selects manual scan The user inputs valid parameters into the manual scan interface The user confirms their choice
Extensions	Invalid parameters are given in the manual scan interface The user receives an error message with debug information surrounding the offending parameters.
Priority	Medium

UC – 3	Assign Ticket
Primary Actor(s)	User
Trigger	The user has confirmed to reassign a ticket from either the ticket interface or the queue interface
Pre-conditions	The user is on the Ticket interface or queue interface The user has selected a Ticket in the interface or is in ticket display view and is now in the ownership interface

Post-conditions	The ticket is assigned to the individual of the user's choosing or back to the queue
Main Success Scenario	<p>The user has selected a Ticket in the interface or is in ticket display view.</p> <p>The user clicks "change ownership"</p> <p>The user searches for a valid user with necessary permissions and group to assign the ticket to and selects it.</p> <p>The user confirms their choice, and the ticket is assigned to the other user</p>
Extensions	<p>The user searches for a user that does not exist</p> <p>An error popup is displayed relaying this information.</p> <p>The individual that the user is attempting to assign the ticket to is not part of the current group which is responsible for the remediation as dictated by the associated policy.</p> <p>An error message is displayed relaying this information to the user</p> <p>The system does not process any changes</p>
Priority	High

UC – 4	Search Term
Primary Actor(s)	User
Trigger	The user has clicked search with a valid query from either the dashboard
Pre-conditions	A search term is entered into the text field
Post-conditions	<p>The system checks the view permissions of the user.</p> <p>The system returns a list of resources that match the search terms based on the found permissions in the search interface</p>
Main Success Scenario	<p>The user enters a search string into the search bar.</p> <p>The system checks the users' permissions.</p>

	<p>The appropriate data bases are searched based on the permissions for the search terms</p> <p>The system returns the results in a new interface</p>
Extensions	<p>No records exist containing the search term.</p> <p>The user is directed to the search interface.</p> <p>A message is displayed stating no results were found where the results should be</p>
Priority	Low

UC – 5	View Assigned Tickets
Primary Actor(s)	User
Trigger	The user clicks the “My Tickets” button on the ticketing interface
Pre-conditions	The user has attempted to navigate to the assigned tickets interface
Post-conditions	The tickets that are assigned to the user are displayed in the assigned ticket interface.
Main Success Scenario	<p>The user has a valid session through authentication</p> <p>The user navigates to the ticketing interface</p> <p>The My Tickets button is clicked.</p> <p>The server receives the request and returns the assigned ticket interface</p>
Extensions	<p>The user has no assigned tickets</p> <p>An informational message is displayed notifying the user of this</p>
Priority	Medium

UC – 6	Close Ticket - Pending
Primary Actor(s)	User
Trigger	The user changes the status of the ticket to “Pending Validation”

Pre-conditions	The user is on the ticket interface
Post-conditions	The ticket progress field is updated to "Pending Validation"
Main Success Scenario	<p>The user finds a ticket through one of the systems features and opens it in a ticket interface.</p> <p>The ticket has passed the necessary checks dictated by the associated policy</p> <p>The tickets progress field is updated to "Pending Validation"</p> <p>The System's stored metric are updated</p>
Extensions	<p>The ticket fails to pass the system checks associated with the related policy.</p> <p>An error message is displayed with associated debug information</p>
Priority	Medium

UC – 7	Update Ticket Properties
Primary Actor(s)	User
Trigger	The user changes the fields of a ticket while in its interface and clicks the "Save button"
Pre-conditions	<p>The user is in possession of the ticket.</p> <p>The user is in the ticket's interface</p> <p>The user has the correct permissions</p> <p>The user has changed fields in the ticket and clicked save</p>
Post-conditions	<p>The system processes the request for validity</p> <p>The checks return no errors</p> <p>The System's stored metric are updated</p> <p>The tickets fields and properties are changed accordingly</p>
Main Success Scenario	<p>The user changes the fields of a ticket while in its interface</p> <p>The user clicks the "Save button"</p> <p>The system processes the request for validity</p>

	<p>The System’s stored metric are updated</p> <p>The tickets fields and properties are changed accordingly</p>
Extensions	<p>The user attempts to update the progress field of the ticket to the next stage.</p> <p>The user is asked for the group member to assign the ticket to who is responsible for that stage which is dictated by the associated policy</p>
Priority	High

UC – 8	Widget Creation
Primary Actor(s)	User
Trigger	The user preforms a query to create a widget.
Pre-conditions	<p>The user is on the widget creation dashboard</p> <p>A valid set of parameters and operations have been set in the search interface.</p> <p>The save button has been clicked</p>
Post-conditions	The system returns information depending on the query. This is displayed visually depending on the query
Main Success Scenario	<p>The user is on the widget creation dashboard and enters a query with valid operations and parameters.</p> <p>The create button is clicked</p> <p>The system processes the request</p> <p>The resulting query is displayed to the user in the formats requested.</p> <p>The widget is created for displaying in dashboards later.</p>
Extensions	<p>The entered parameters where invalid. They could either be out of range or the query syntax could be incorrect.</p> <p>The system returns the user to the previous interface.</p>

	<p>The system displays a debug message conveying this information</p> <p>The system does not process any changes</p>
Priority	Medium

UC – 9	Dashboard Creation
Primary Actor(s)	User
Trigger	The user preforms a query to create a Dashboard.
Pre-conditions	<p>The user is on the dashboard creation dashboard</p> <p>The widgets have been set with valid settings in the interface.</p> <p>The save button has been clicked</p>
Post-conditions	The Dashboard is saved and added to the repository. It may be used later for reports or displaying in the application
Main Success Scenario	<p>The user is on the dashboard creation dashboard</p> <p>The widgets have been set with valid settings in the interface. Size and grid displacement should be decided.</p> <p>The save button has been clicked.</p> <p>The system processes the request</p> <p>The dashboard is displayed to the user in the formats requested.</p> <p>The Dashboard is saved and added to the repository. It may be used later for reports or displaying in the application</p>
Extensions	<p>The widgets settings were invalid. This could be caused by configurations that would cause an overlap on the dashboard.</p> <p>The system returns the user to the previous interface.</p> <p>The system displays a debug message conveying this information</p> <p>The system does not process any changes</p>
Priority	Medium

UC – 10	Generate Report
Primary Actor(s)	User
Trigger	The user has clicked generate report on a metrics dashboard or on the metrics search interface
Pre-conditions	The user has a valid session The user has the correct permissions to perform the action Valid parameters are set in the interfaces
Post-conditions	The system processes the query and outputs the results to a report for the user
Main Success Scenario	The user has either preformed a successful query through metrics search or has click export dashboard on a metrics dashboard The system processes the request The system outputs the results to PDF and returns to the user.
Extensions	The system is unable to process the request due to invalid queries. The system returns a debug message and sends the user to the previous interface. The user clicks the generate report button with no query or parameters set. The system returns a debug message and sends the user to the previous interface.
Priority	Medium

UC – 11	View in-progress tickets
Primary Actor(s)	User
Trigger	The user requests the Tickets In-progress page.
Pre-conditions	None
Post-conditions	The user’s tickets are displayed in a table.

	Tickets in progress across users are displayed in a table
Main Success Scenario	<p>The user navigates to the Tickets in-progress interface.</p> <p>The system preforms its checks and confirms it a valid request</p> <p>The tickets in-progress interface is returned to the browser</p>
Extensions	<p>There are currently no tickets in progress</p> <p>The system displays a message conveying this information.</p> <p>The user currently has no tickets in their possession</p> <p>The system displays a message conveying this information.</p>
Priority	Medium

UC – 12	View tickets in progress – keywords
Primary Actor(s)	User
Trigger	The user enters search terms into the keyword filter and clicks confirm
Pre-conditions	<p>The user is on the closed tickets in progress page</p> <p>The user enters search terms into the keyword filter and clicks confirm</p>
Post-conditions	The system returns any tickets in progress that contain one or more of the keywords in its properties from most to least relevant
Main Success Scenario	<p>The user is on the tickets in progress interface.</p> <p>The user enters search terms into the keyword filter and clicks confirm</p> <p>The system preforms checks on the user input and determines it valid</p> <p>The system returns any tickets in progress that contain one or more of the keywords in its properties from most to least relevant</p>

Extensions	<p>There were no results containing any of the keywords</p> <p>The system displays a message conveying this information</p> <p>The user does not input any characters into the field and confirms a search.</p> <p>The system returns the tickets in-progress as its default.</p>
Priority	Medium

UC – 13	View closed tickets
Primary Actor(s)	User
Trigger	The user navigates to the closed tickets interface
Pre-conditions	None
Post-conditions	The system returns the closed ticketing interface
Main Success Scenario	<p>The user clicks on the closed ticket reference in the application.</p> <p>The system returns the closed ticket interface</p>
Extensions	<p>There are no closed tickets to display.</p> <p>The system displays a message conveying this information in their place</p>
Priority	Low

UC – 14	View QA Tickets
Primary Actor(s)	User
Trigger	<p>The user navigates to the closed tickets interface</p> <p>The user selects the display QA Tickets function</p>
Pre-conditions	The user is on the closed ticket interface
Post-conditions	The system returns the closed ticketing interface with results filtered on if they received QA.

Main Success Scenario	The system clicks on the display QA Tickets function from the closed ticket interface.
Extensions	There have been no tickets that received QA The system displays a message to the user conveying this information.
Priority	Low

UC – 15	Closed tickets – Modify time range
Primary Actor(s)	User
Trigger	The user enters a Date/time range on this closed ticket interface
Pre-conditions	The user is on the closed ticket interface
Post-conditions	The system returns closed ticket results from that time range
Main Success Scenario	The user is on the closed ticket interface. The clicks on the date/time range function A date/time selection is displayed to the user The user selects a valid date/time range and clicks confirm The system returns a new ticket table with only results from the selected range.
Extensions	The user attempts to enter a date range from the future The system returns a message stating that the date/time parameters are invalid.
Priority	Low

UC – 16	Closed tickets – keywords
Primary Actor(s)	User
Trigger	The user enters search terms into the keyword filter and clicks confirm
Pre-conditions	The user is on the closed tickets in progress page

	The user enters search terms into the keyword filter and clicks confirm
Post-conditions	The system returns any closed tickets that contain one or more of the keywords in its properties from most to least relevant
Main Success Scenario	<p>The user is on the tickets in progress interface.</p> <p>The user enters search terms into the keyword filter and clicks confirm</p> <p>The system performs checks on the user input and determines it valid</p> <p>The system returns any closed tickets that contain one or more of the keywords in its properties from most to least relevant</p>
Extensions	<p>There were no results containing any of the keywords</p> <p>The system displays a message conveying this information</p> <p>The user does not input any characters into the field and confirms a search.</p> <p>The system returns the tickets in-progress as its default.</p>
Priority	Low

UC – 17	Update Ticket properties - Reopen closed ticket
Primary Actor(s)	User
Trigger	The user changes the progress property of a closed ticket to in progress
Pre-conditions	The user is currently on a ticket’s interface that has the progress property “closed”
Post-conditions	The ticket’s progress property is reverted to “in progress” and the ticket is returned unassigned to the queue
Main Success Scenario	<p>The user is currently on a ticket’s interface that has the progress property “closed”</p> <p>The user changes the progress property of a closed ticket to in progress.</p>

	<p>The system reverts the field progress to closed and clears the remediated property.</p> <p>The system updates its metrics</p> <p>The ticket is returned unassigned to the queue</p>
Extensions	None
Priority	Medium

User/Role	Example	Frequency of Use	Security/Access, Features Used	Additional Notes
Administrator	Administrator	High	Extended Privileges, Create User, Update User, Lock User, Delete User, View Activity, View Session Log	Administrator users will have full control over the system

UC - 1	Create User
Primary Actor(s)	Administrator
Trigger	The Administrator has navigated to the "Create User" Interface. The required information for user creation has been filled. The new user has been confirmed
Pre-conditions	<p>The user is logged on to an Administrator account and has a valid session.</p> <p>The user creation form has its required fields filled.</p>
Post-conditions	A new user is successfully registered on the system for use.
Main Success Scenario	<p>The user logs into an Administrator account.</p> <p>The user navigates to the Create User page</p> <p>The user fills all required fields for user creation</p>

	<p>The clicks the create new user button and confirms their choice</p> <p>The system checks the validity of the entered information by performing a series of checks</p> <p>If all the checks are a success, a new user is created</p>
Extensions	<p>If the user clicks cancel on the create user page</p> <p>The entered information is cleared</p> <p>The entered information fails a system check like the attempt to create a duplicate user.</p> <p>An error message is displayed with debug information on what caused the failure.</p> <p>The user is returned to the User Creation page with the entered information intact</p> <p>If the user does not have a valid session</p> <p>The does not have a valid session.</p> <p>The system directs the user to the login screen</p>
Priority	High

UC - 2	Delete User
Primary Actor(s)	Administrator
Trigger	The Administrator clicks on the delete User button
Pre-conditions	<p>The user is logged on to an Administrator account and has a valid session.</p> <p>The Administrator is on the user properties interface</p> <p>The administrator confirms the dialog box that appears for the action</p>
Post-conditions	The user is moved to disabled accounts for a set period until the time passes and is permanently deleted

<p>Main Success Scenario</p>	<p>The user logs into an Administrator account.</p> <p>The user navigates to the User properties page</p> <p>The user clicks the delete user button</p> <p>The user confirms their choice</p> <p>The system a series of checks to ensure the authenticity of the action</p> <p>If all the checks are a success, the user account is disabled for a set period before being deleted</p>
<p>Extensions</p>	<p>If the user clicks cancel on the create user page</p> <p>The entered information is cleared.</p> <p>The system does not process any changes</p> <p>The entered information fails a system check like the attempt to create a duplicate user.</p> <p>An error message is displayed with debug information on what caused the failure.</p> <p>The user is returned to the User property page with the entered information intact.</p> <p>No changes are processed.</p>
<p>Priority</p>	<p>Low</p>

<p>UC - 3</p>	<p>Update User</p>
<p>Primary Actor(s)</p>	<p>Administrator</p>
<p>Trigger</p>	<p>The Administrator edits the user’s properties and clicks confirm</p>
<p>Pre-conditions</p>	<p>The user is logged on to an Administrator account.</p> <p>The Administrator is on the user properties interface</p> <p>The administrator edit user property fields and clicks confirm</p>
<p>Post-conditions</p>	<p>The user’s account is updated according to the saved changes</p>

<p>Main Success Scenario</p>	<p>The user logs into an Administrator account.</p> <p>The user navigates to the User properties page</p> <p>The administrator edit user property fields and clicks confirm.</p> <p>The system a series of checks to ensure the authenticity of the action</p> <p>The user’s account is updated according to the saved changes</p>
<p>Extensions</p>	<p>If the user clicks cancel while having changed properties.</p> <p>The entered information is cleared.</p> <p>The system does not process any changes</p> <p>The entered information fails a system check like the attempt to assign properties that are not valid.</p> <p>An error message is displayed with debug information on what caused the failure.</p> <p>The user is returned to the User property page with the entered information intact.</p> <p>No changes are processed.</p>
<p>Priority</p>	<p>Low</p>

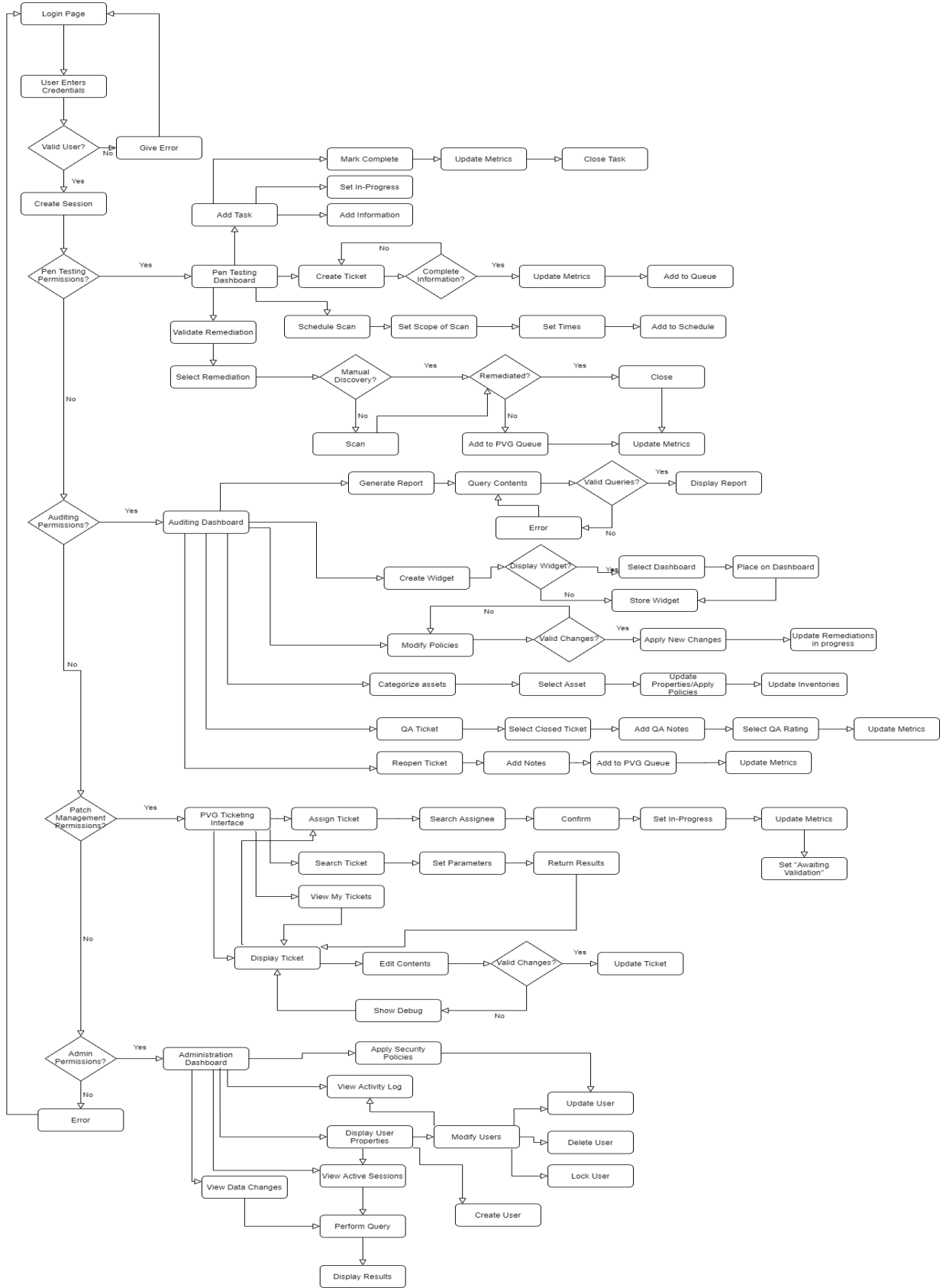
<p>UC - 4</p>	<p>View Activity</p>
<p>Primary Actor(s)</p>	<p>Administrator</p>
<p>Trigger</p>	<p>The Administrator enters the activity log interface</p>
<p>Pre-conditions</p>	<p>The user is logged on to an Administrator account.</p> <p>The user clicks the reference to the activity interface</p>
<p>Post-conditions</p>	<p>The user activity is displayed in a tabular form and can be further filtered</p>
<p>Main Success Scenario</p>	<p>The user logs into an Administrator account.</p> <p>The user clicks the reference to the activity interface</p>

	The user activity is displayed in a tabular form and can be further filtered
Extensions	<p>If the user clicks cancel while having changed properties.</p> <p>The entered information is cleared.</p> <p>The system does not process any changes</p> <p>The entered information fails a system check like the attempt to assign properties that are not valid.</p> <p>An error message is displayed with debug information on what caused the failure.</p> <p>The user is returned to the User property page with the entered information intact.</p> <p>No changes are processed.</p>
Priority	Low

DATA FLOW DIAGRAMS

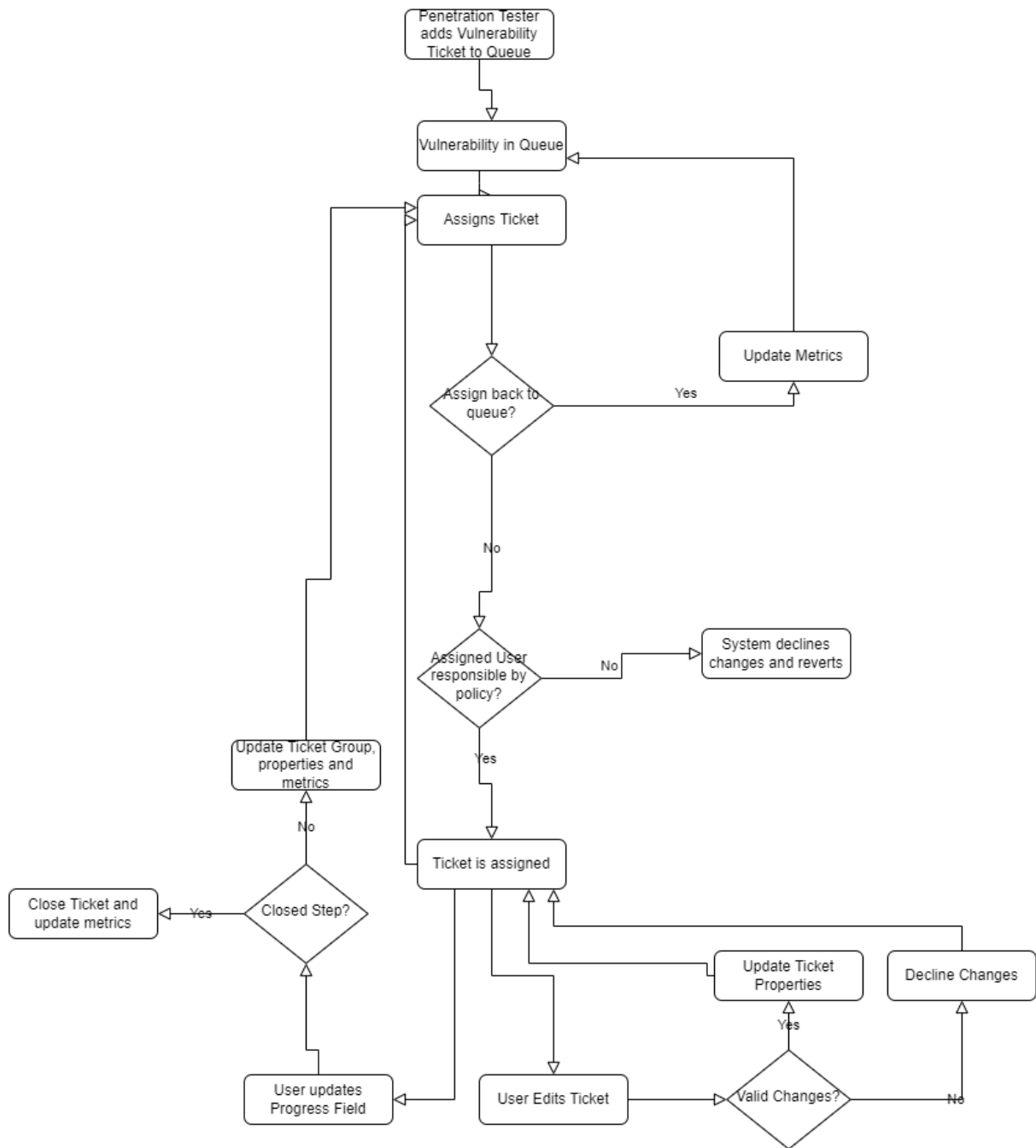
High level Data-flow diagrams of the system and its components.

SYSTEM DATA FLOW



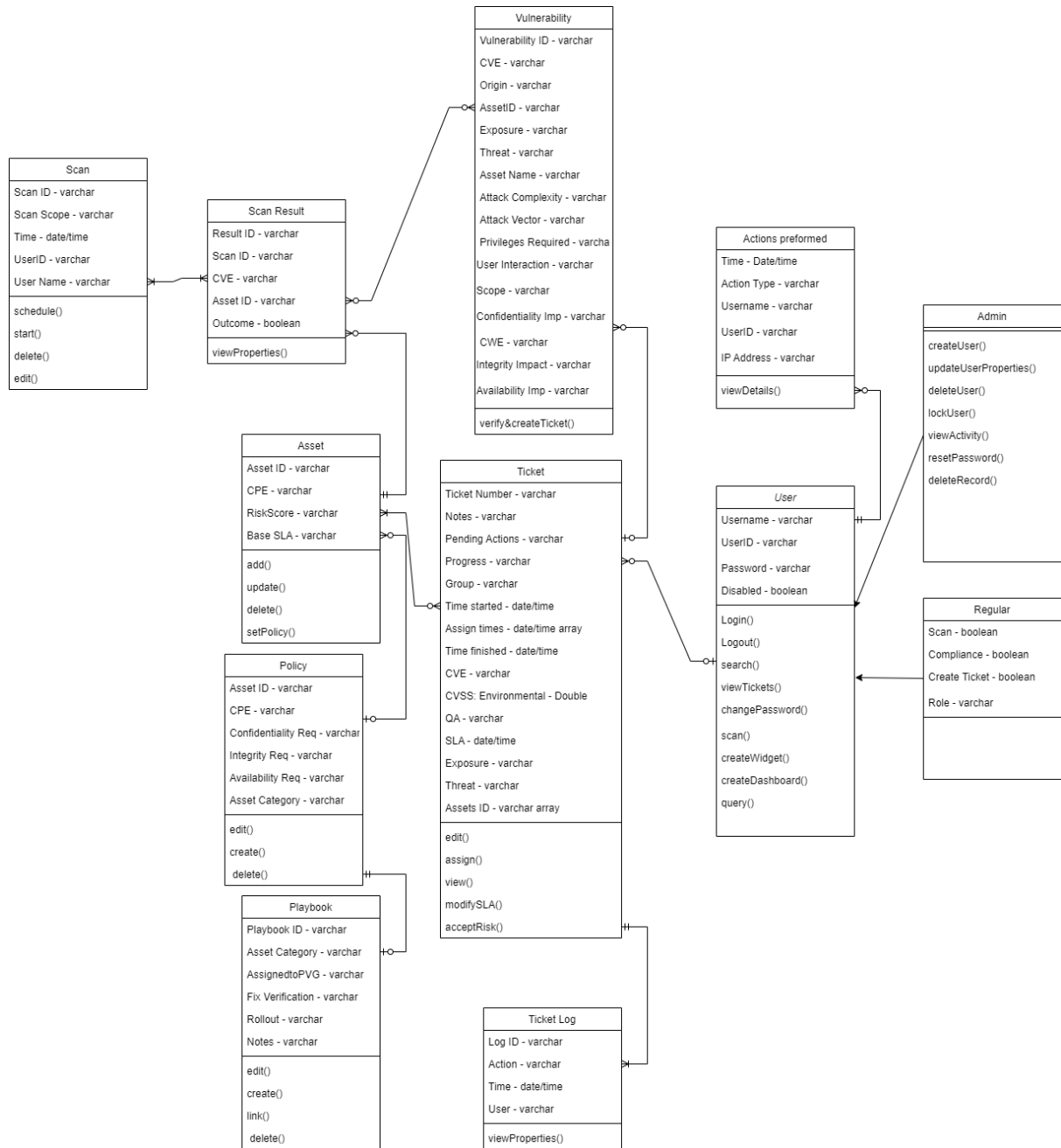
Data flow as it passes through the entire system.

TICKET DATA FLOW



The lifecycle of a ticket as it passes through the system

LOGICAL DATA MODEL



High level data model of the system.

FUNCTIONAL REQUIREMENTS

List of the main functional requirements of the system

Section/ Requirement ID	Requirement Definition
FR1.0	The system shall require authentication to access
FR1.1	The system shall determine the permissions of the user
FR1.1.1	The system shall provide access to resources based on the permissions

Section/ Requirement ID	Requirement Definition
FR2.0	The system shall allow a permitted user to create a vulnerability ticket.
FR2.0.1	The system shall add the ticket to the patch management queue
FR2.1	The system shall allow users to assign tickets
FR2.2	The system shall allow users to update tickets
FR2.3	The system shall allow users to close tickets
FR2.3.1	The system shall allow users to verify closed tickets
FR2.4	The system shall allow users to search tickets

Section/ Requirement ID	Requirement Definition
FR3.0	The system shall scan the environment for vulnerabilities based on a schedule.
FR3.0.1	The system shall allow users to add found vulnerabilities in scans to the queue.
FR3.1	The system shall allow users to modify the schedule
FR3.2	The system shall allow users to set the scope of scans
FR3.3	The system shall allow users to manually trigger a scan

Section/ Requirement ID	Requirement Definition
FR4.0	The system shall store the metrics involving the vulnerability management cycle
FR4.0.1	The system shall allow users to create dashboards and widgets with the data
FR4.0.2	The system shall allow the user to query metrics
FR4.0.3	The system shall return custom reports created by users
FR4.1	The system shall allow SLAs and policies to be modified

Section/ Requirement ID	Requirement Definition
FR5.0	The system shall allow the user to categorize assets.
FR5.1	The system shall allow the user to discover assets.

Section/ Requirement ID	Requirement Definition
FR6.0	The system shall allow an administrator to create users.
FR6.1	The system shall allow an administrator to modify user properties.
FR6.2	The system shall allow an administrator to delete users
FR6.3	The system shall allow an administrator to disable users
FR6.4	The system shall allow an administrator to view activity history
FR6.4.1	The system shall allow an administrator to search activity history

HARDWARE/SOFTWARE REQUIREMENTS

Recommended Technology Stack

Angular.js – Front-end (Excels at creating dynamic web pages which will be critical to the usability of the ticketing system. Angular can directly interact with database contents provided they are in json format. This means it is possible to have a system that updates in real time in the browser).

Python Django- Back-end (Powerful back-end framework that will allow access to the vast toolset of Python. Some of the planned functionality is certainly achieved with Python)

MySQL – Database (Excels with the relational database structure needed. Oracle is a decent choice too. I am choosing MySQL due to my familiarity)

OpenVas – Open-source vulnerability scanner that can detect over 50,000 CVEs through scanning assets. Support for integration through APIs.

The application will be cross-platform and will be possible to host on any noteworthy operating systems like Windows, Linux, and Macintosh. It will be recommended to be ran on a dedicated server device within the organisation’s environment. Using cloud-based solutions for hosting the application will require security configuration changes across the environment to achieve its goals but will be an option as it has many worthwhile advantages.

A database solution will have to be deployed to store application data. This can be either packaged into the application installation or allow the organisation to set the location of their own database.

User interfaces to access functionalities of application will be available as a hosted web service. This is the desired choice in this scenario as it will be incredibly easy to achieve compatibility with client devices. This adds a software requirement of a verified browser for users. Some challenges that will need to be overcome are cross-browser compatibility and mobile devices.

Both challenges can be overcome by continuously performing compatibility testing during development. This will mitigate incompatible features for some browsers and ensures the application performs optimally and is user friendly on all platforms.

OPERATIONAL REQUIREMENTS

The system will be responsible for tracking any noteworthy metrics in the vulnerability cycle

The system will be required to keep a comprehensive list of managed assets within the organisation

The system needs to keep records of authorized users and provide functionality to them based on permissions

The system will be required to track the vulnerability management lifecycle through its ticketing system.

The system will be responsible for ensuring only authorized users can access its web application.

The system will be able to use metrics to generate visuals and reports.

The system will be required to keep a scan schedule that is updated by users

The system will integrate with a suitable vulnerability and network scanning tool.

ERROR HANDLING

Attempting to process incomplete information will through an error message stating the missing fields. The user is sent back to the previous screen.

Executing an action that that the user does not have permission for will throw an error. The system will not preform the action.

The system will check any user input for correctness. If unexpected input is detected and sent to the server, an error message will be returned.

VALIDATION RULES

Time: Any action involving scheduling or automatic report generation will have check if the input is either in the present or future. Any attempt to put in a time from the past will display an error message.

Required Fields: The system will not process any requests from the user if it is missing data that has a required tag. The user will be returned to the previous selection where the request originated.

Out of scope input: The system will not allow the user to create or update tickets with incorrect information. An example would be setting a policy that does not match the vulnerability/asset.

Permissions: Any action preformed will prompt the system to check the authority of the user. If the permission is absent, they will receive an error message and the action will not be performed.

Encoding: Input inserted into a web page will be appropriately encoded depending on the location it is inserted in the source code before it is allowed to be displayed in the browser

Prepared Statements: User input that queries the database will use prepared SQL statements with variable binding.

SECURITY AND PRIVACY

The consequences of the system becoming compromised are:

- **Loss or corruption of data:** Stored vulnerability management metrics, tickets and user information could be lost
- **Disclosure of secrets or sensitive information:** Information on the organisations posture on vulnerabilities will be compromised. This can allow targeted attacks against vulnerable systems.

- **Disclosure of privileged/privacy information about individuals:** Any information that can identify a user will have to be treated according to appropriate regulation.
- **Corruption of software or introduction of malware, such as viruses:** Weaknesses in the application could allow attackers to compromise the server it is hosted on.

SECURITY REQUIRED ON THE SYSTEM

Physical: The system the application is hosted on should be contained within a locked room with only authorized access. A UPS power supply should be used to protect its availability.

Permissions: The system will require authorization on actions preformed. Least privilege methodologies should be used for users.

Access control requirements: Only administrators should have the authority to add, modify or restrict user accounts.

RELIABILITY

Failures in the system to preform its intended goals include

- Complete loss in being able to track the vulnerability management life cycle
- Partial loss in being able to co-ordinate the vulnerability management life cycle
- Loss of productivity within the vulnerability management team
- Loss in revenue due to an attack surface increase as remediations are impeded.

PROJECT PLAN

Action	Time
Learn Angular, Django	17 th December – 31 st December
Core GUI	1 st January – 22 nd January
Ticket Operations	23 rd January – 7 th February
Dashboard – Widget Creation	8 th February – 22 nd February
Vulnerability Scanner Integration	23 rd February – 9 th March
Optional GUI	10 th March – 17 th March
Optional Operations	18 th March – 25 th March
Reserved	26 th March – 5 th April
Final Report	6 th April – 17 th April

Scheduled may change depending on the success of the project. If an action exceeds its stated time, it will be postponed for the next stage unless critical.

CORE GUI – SUB TASKS

1. Application Web Template: Dependencies - None
2. Ticket – Queue Interface: Dependencies - 1
3. Ticket – View My Tickets: Dependencies - 1
4. Ticket - Search Ticket: Dependencies - 1
5. Ticket – Create Ticket: Dependencies - 1
6. Audit Dashboard – Main: Dependencies - 1
7. Audit Policy – Main: Dependencies - 1
8. Audit Policy – Playbook: Dependencies - 1
9. Admin Create User: Dependencies - 1
10. User Login/Logout: Dependencies – 9

TICKET OPERATIONS – SUB TASKS

11. Ticket Interface: Dependencies – 2
12. Assign Ticket: Dependencies – 2,3
13. Edit Ticket: Dependencies - 11
14. Ticket – Special Operations, Dormant, Reopen: 13

DASHBOARD WIDGET CREATION

15. Dashboard GUI Repository - 6
16. Widget GUI Repository - 6
17. Widget CRUD: Dependencies - 16
18. Dashboard CRUD: Dependencies - 15
19. Display Dashboard GUI: Dependencies - 18
20. Display Widget GUI: Dependencies - 17

VULNERABILITY SCANNER INTEGRATION

21. Scan GUI: Dependencies - 1
22. Scan operations: Dependencies - 21
23. Store Scan Results: Dependencies – 22
24. View History – Scan Results: Dependencies - 23
25. Penetration Tester Vulnerability GUI: Dependencies - 1
26. Add Vulnerability to Penetration Testers Vulnerability GUI: Dependencies - 24

OPTIONAL GUI

27. Knowledge Base GUI: Dependencies - 1
28. Administrator CRUD GUI: Dependencies - 1
29. Session Log GUI: Dependencies – 10,12,13,14,19,20,24,26,28,32,35,37
30. Asset Repository GUI: Dependencies - 1

- 31. Asset properties GUI: Dependencies - 30
- 32. Create Asset GUI: Dependencies - 31

OPTIONAL OPERATIONS

- 33. Web parse – CVE, CVSS, CWE: Dependencies – 11, 26
- 34. Ticket Searches: Dependencies - 11
- 35. Ticket GUI Results Filter: Dependencies - 34
- 36. Administrator CRUD: Dependencies - 1
- 37. Scan Asset Discovery: Dependencies - 22
- 38. Create Asset: Dependencies – 37

BIBLIOGRAPHY

almooc. (17, May 2017). *FRDTemplate*. Retrieved from almooc.com:

<https://almooc.com/downloads/>

Cavoukian, A. (2011, January 1). *Privacy by Design: The 7 Foundational Principles*. Retrieved from

<https://www.privacysecurityacademy.com/>: <https://www.privacysecurityacademy.com/wp-content/uploads/2020/08/PbD-Principles-and-Mapping.pdf>

GDPR. (2021, November 18). *Complete guide to GDPR compliance*. Retrieved from gdpr.eu:

<https://gdpr.eu/>