



**Quality.  
Timeliness.  
Customer Service.**

A Leidos Company

# **Secure Module/Key Management Application**

## **Technical Specification Document**

## Revision History

Rev #	Date	Author	Description of Change
1.0	10/25/2023	Design Team	Added Use Case 1 workflow diagram and SQL table.
2.0	11/1/2023	Design Team	Add Use Case 2&3 workflow diagram. Fixed SQL table.
3.0	11/8/2023	Design Team	Added descriptions to workflow diagrams. Highlight parts we don't understand in red.
4.0	11/24/2023	Design Team	Steps were added to the diagram to show how the application will operate in different use cases.
5.0	11/29/2023	Design Team	Updated steps and diagrams. Instructions on how to set up Azure Key Vault, its integration, and OpenSSL setup.
6.0	12/19/2023	Design Team	Added detailed steps for the creation and integration of Azure Key Vault and Certificates.
7.0	4/10/2024	Design Team	Changed diagrams and documentation to align with the new implementation.
8.0	4/12/2024	Documentation Team	SQL Database image has been updated. added controller documentation still needs information

## Table of Contents

1 General Information	1
1.1 Project Overview	1
1.2 References	1
2 Technical Design	2
2.1 Microservices	2
2.1.1 Microservice S1	2
2.2 Integrations	4
2.2.1 Azure Key Vault	4
2.2.2 Orchestration and Workflow	8
2.2.3 Extract Transform Load (ETL)	8
2.2.4 Robotic Process Automation (RPA)	8
2.3 User Interfaces (UI)	9
2.3.1 Generate Data Page	9
3 Use cases	11
3.1 Use Case 1	11
3.2 Use Cases 2 and 3	13
4 Appendix	15

## List of Figures

Figure 1: S1 Component Diagram	2
Figure 2: S1 Entity Relationship Diagram	4
Figure 3: P1 Process Model	6
Figure 4: Web 1 Component Diagram	
Figure 5: Generate Data Page	
Figure 6: View Data Page	
Figure 7: View Data Page Option	7
Figure 8: [Use Case 1] Sequence Diagram	9
Figure 9: [Use Case 2 & 3] Sequence Diagram	9

## List of Tables

Table 1: Stakeholders	1
Table 2: S1 Published API Discovery Links	2
Table 3: S1 Event Broker	2
Table 4: S1 Event Subscriptions	3
Table 5: S1 Event Broker	3
Table 6: S1 Produced Events	3
Table 7: Data Store Inventory	4
Table 8: S1 Logging and Monitoring	5
Table 9: Aggregated APIs Discovery Links	5
Table 10: Web 1 Links	8
Table 11: Web 1 Role-based Access Control (RBAC) Matrix	8
Table 12: Web 1 Logging and Monitoring	8
Table 13: S1 Controller Methods	8

# 1 GENERAL INFORMATION

## 1.1 Project Overview

Secure Module/Key Management Project is an application that does secure data transfer between two points in three use cases: passing data between servers in a trusted domain, passing data from a server in a trusted domain to another server outside the trusted domain, passing data from a server in a trusted domain to another server outside the trusted domain over an unknown number of intermediary systems. The application will use HTTPS to transfer data and log all HTTP activity from all three use cases. The logged data should be encrypted and be visible to allowed users. The application will use Microsoft Azure Vault, ASP.NET Core, Microsoft SQL Server.

Goals:

1. Passing data from one server to another inside a trusted domain across an HTTPS connection, but HTTP activity is logged to a central logging database that users are able to access. Sensitive data (PHI/PII) that is part of the data payload must be protected from being visible to users accessing the logs.
2. Passing data from one server in a trusted domain to another server outside of the trusted domain across an HTTPS connection. HTTP activity is logged as detailed above.
3. Passing data from one server in a trusted domain to another server outside of the trusted domain with an unknown number of intermediary systems handling the data between the two endpoints. Data is transmitted across HTTPS connections and HTTP activity should expect to be logged as detailed above.

Stakeholder	Name	Email Address
Owner	QTC	
Alternate Owner		
Technical Contact	Francisco Guzman	
Alternate Technical Contact	Richard Cullen	

Table 1: Stakeholders

## 1.2 References

- [Link to external document 1, etc. Include Architecture and Requirements artifacts]
- SDD (Dashboard Design Draft)
- SECURE MODULE / KEY MANAGEMENT PROJECT PDF from Richard Cullen

## 2 TECHNICAL DESIGN

### 2.1 Microservices

**Does not apply.**

#### 2.1.1 Microservice S1

*Figure 1: S1 Component Diagram*

##### 2.1.1.1 Inbound

Does not apply.

###### 2.1.1.1.1 S1 Event Subscriptions

Does not apply.

##### 2.1.1.2 Outbound

###### 2.1.1.2.1 S1 Aggregate APIs

Does not apply.

###### 2.1.1.2.2 S1 Produced Events

Does not apply.

##### 2.1.1.3 S1 Data Stores

Type	Name	Description
Databases Microsoft SQL Server	QTC SV Members.	Store PHI/PII data on SV Members.
File System For example: SMB/CIFS/Samba, NFS, Amazon EFS	N/A	N/A
Blob For example: Azure Blob Storage, Amazon S3, Amazon EBS	N/A	N/A
Message/Event Queue For example: RabbitMQ, Kafka/Confluent, Azure Event Grid/Hub/Azure Service Bus, Amazon SQS, Amazon	N/A	N/A

MQ/Apache MQ, Amazon SNS, Amazon Kinesis Streams, QTC MQ		
--	--	--

*Table 7: Data Store Inventory*

[In the following sections, list each owned data store.]

#### 2.1.1.3.1 S1 SQL Database

SV.Member	
PK	MemberID [Unique Identifier] NOT NULL  FirstName [nvarchar] NULL LastName [nvarchar] NULL SSN [nvarchar] NULL Encrypted DOB [Date] NULL MiddleInitial [nvarchar] NULL Gender[nvarchar] NULL Email [nvarchar] NULL CellPhone[nvarchar] NULL Session Key [varchar] NULL

*Figure 2: S1 Entity Relationship Diagram*

#### 2.1.1.4 S1 Logging and Monitoring

**Does not apply.**

Type	Name
Monitoring	
Data Store	[File; Database; Windows Event; Messaging]
Events Captured	[User Access Control; User Activity; Application Exception]
Audit Logging Review Process	

*Table 8: S1 Logging and Monitoring*

---

## 2.2 Integrations

---

### 2.2.1 Azure Key Vault

Azure Key Vault is a cloud service designed to safely store secrets, keys, and certificates.

#### 1. Set up an Azure subscription:

- Create an Azure account.

#### 2. Sign into the Azure Portal:

- Access Azure dashboard to manage resources.

#### 3. Creating a vault:

##### Step 1: Sign in to Azure Portal

- Go to the [Azure Portal](<https://portal.azure.com/>).
- Log in using your Azure account credentials.

##### Step 2: Open Azure Key Vault Service

- Once logged in, search for "Key Vault" in the search bar at the top of the portal.
- Select "Key Vault" from the search results.

##### Step 3: Create a New Key Vault

- Click on the "+ Create" button to start the process of creating a new vault.
- This opens the "Create Key Vault" wizard.

##### Step 4: Basics Tab

- Subscription: Select the Azure subscription you want to use.
- Resource Group: Choose an existing resource group or create a new one.  
Resource groups are containers that hold related resources for an Azure solution.
- Key Vault Name: Enter a unique name for your vault.
- Region: Select the geographical region where your vault will be located.

##### Step 5: Access Policy Configuration

- Go to the "Access policies" tab.

- Configure who has access to the Key Vault and what permissions they have. You can set permissions for keys, secrets, and certificates. Azure role-based access control or Vault access policy.
- You can add access policies later.

### **Step 6: Networking**

- Select the networking tab.
- Configure the networking settings as per your requirements. You can choose to allow access from all networks or set up specific network rules.
- Create a private endpoint to allow a private connection to this resource. Additional private endpoint connections can be created within the key vault or private link center. Fields for private endpoint: Name, Subscription, Resource group, Region, Subnet, Private DNS Zone.

### **Step 7: Tags (Optional)**

- Use the "Tags" tab to apply tags to your Key Vault. Tags are name/value pairs that enable you to categorize resources and view consolidated billing.

### **Step 8: Review and Create**

- Go to the "Review + create" tab.
- Azure will validate your configuration. Once validation passes, review your settings.
- Click "Create" to deploy the Key Vault.

### **Step 9: Manage the Vault**

- After the Key Vault is created, go to its overview page.
- Here, you can manage secrets, keys, and certificates.
- You can also modify access policies and networking settings as needed.

### **Step 10: Monitor and Maintain**

- You can also monitor the usage and access logs.
- Update policies and configurations as needed to ensure the security of your stored keys and secrets.

## **Integrate Azure Key Vault into your ASP.NET website:**

### **Prerequisites:**

- Visual Studio Code.
- Install the .NET SDK if it's not already installed.
- Install the C# extension for Visual Studio Code to work with .NET applications.

### **Refer to this link to install Azure Key Vault in our web application:**

<https://learn.microsoft.com/en-us/visualstudio/azure/vs-key-vault-add-connected-service?view=vs-2022>

### **1. Install Azure Key Vault NuGet packages:**

- Add the “Azure.Security.KeyVault.Secrets”, “Azure.Identity”, and “Microsoft.Extensions.Configuration.AzureKeyVault” packages and others as needed to our ASP.NET project.

### **2. Register your application with Microsoft Entra ID:**

#### **1. Register the Application:**

- Navigate to Microsoft Entra ID.
- Select “App registrations”.
- Click on “New registration”.
- Enter the name for your application.
- Specify the supported account types (choose according to your organization's needs).
  - Supported account types:

Who can use this application or access this API?

- Accounts in this organizational directory only (Cal State LA only (Organization) - Single tenant)
  - Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
  - Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
  - Personal Microsoft accounts only.
- Set the redirect URI (optional, mainly used for web applications). Return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

- Click “Register”.

## **2. Get Application (Client) ID and Directory (Tenant) ID:**

- After registering, you'll be taken to the application's overview page.
- Note down the Application (Client) ID and Directory (Tenant) ID, as these will be needed later.

## **3. Create a Client Secret:**

- Go to “Certificates & secrets” under your app registration.
- Click on “New client secret”.
- Provide a description and set an expiry period.
- Click “Add”.
- Note: the secret value won't be retrievable after you leave this page.

## **4. Assign Permissions to Your Application in Azure Key Vault**

### **1. Go to Your Key Vault:**

- In the Azure Portal, navigate to the Key Vault you want your application to access.

### **2. Add Access Policy:**

- Go to “Access policies” in your Key Vault settings.
- Click on “Create”.
- Select the secret permissions (like Get, List, Set) your application requires.
- Click on the “Principal” tab and search and choose your registered application.
- Click “Create”.

## **5. Update QTC and 3rd Application to Access Azure Key Vault**

### **1. Install Azure SDK Packages:**

- If not already done, install the necessary Azure SDK packages in your application, such as `Azure.Security.KeyVault.Secrets` and `Azure.Identity`.

## **2. Configure Your Application:**

- In your application, add the necessary code to authenticate using the Client ID, Tenant ID, and Client Secret.
- Use the Azure SDK to retrieve secrets from the Key Vault.

## **3. Test the Integration:**

- Run your application to ensure it can successfully retrieve secrets from Azure Key Vault.

## **3. Configure application:**

- In your “Startup.cs” file or through the app settings, use the “AzureServiceTokenProvider” class from the “Microsoft.Azure.Services.AppAuthentication” package to authenticate with Microsoft Entra ID and obtain tokens.

## **4. Access secrets in your code:**

- Utilize the “SecretClient” class from the “Azure.Security.KeyVault.Secrets” package to retrieve secrets from the Key Vault.

## **5. Implement proper access control:**

- Ensure that your application has the necessary permissions to access the Key Vault by setting access policies in the Key Vault settings.

## **6. Handle secrets::**

- Handle the rotation of secrets in the Key Vault.

## **Azure Key Vault Integration with 3rd Party Applications:**

### **1. SSL Certificate for Secure Communication:**

- Implement SSL certificates on our server and the 3rd party’s server. This ensures that any data transmitted between the two servers is encrypted and secure.

### **2. Encrypt Data Before Transmission:**

- Before sending data to the 3rd party, encrypt it with RSA.

### **3. 3rd Party Application Registered in Microsoft Entra:**

- The 3rd party application must be registered in Microsoft Entra. This registration allows the application to authenticate with Azure and request access tokens.

### **4. Configure Azure Key Vault:**

- Store the keys in Azure Key Vault. Configure access policies in Azure Key Vault to grant the 3rd party application permission to retrieve the decryption key.

### **5. 3rd Party Application Requests Access Token:**

- When the 3rd party application needs to decrypt the data, it first requests an access token from Microsoft Entra ID, authenticating itself using its credentials (like a client ID and secret, or a certificate).

### **6. 3rd Party Application Accesses Azure Key Vault:**

- With the access token, the 3rd party application makes a request to Azure Key Vault to retrieve the decryption key. The request must be authenticated and authorized by Azure Key Vault.

### **7. Decryption of the Data:**

- After successfully retrieving the decryption key from Azure Key Vault, the 3rd party application uses this key to decrypt the data.

### **8. Secure Storage of Keys and Credentials:**

- Ensure that the keys and credentials are securely stored and handle the decryption key securely.

### **9. Token and Key Management:**

- Handle token expiration and renewal when interacting with Azure AD. It should also follow best practices for key management, including key rotation. We will expire after use and create new keys for next use.

### **10. Audit and Monitoring:**

- Regularly monitor and audit the access logs of Azure Key Vault and Azure AD to ensure that the key access and authentication requests are legitimate.

## **Creating OpenSSL Certificate**

Refer to link for installation steps and creating Self Signing Certificate:

<https://knowledge.digicert.com/solution/generate-a-certificate-signing-request-using-openssl-on-microsoft-windows-system>

### **1. Install OpenSSL:**

- Ensure that OpenSSL is installed on your system. It's available for Windows, Linux, and macOS.

### **2. Generate a Private Key:**

- For each organization, generate a private key using OpenSSL.
- This generates a 2048-bit RSA private key and saves it to a .pem file.

### **3. Create a Certificate Signing Request:**

- Generate a CSR using the private key.
- The CSR includes: (referenced from link above.)
  - Country Name: Use the two-letter code without punctuation for country, for example: US or CA.
  - State or Province: Spell out the state completely; do not abbreviate the state or province name, for example: California.
  - Locality or City: The Locality field is the city or town name, for example: Berkeley. Do not abbreviate. For example: Saint Louis, not St. Louis.
  - Company: If the company or department has an &, @, or any other symbol using the shift key in its name, the symbol must be spelled out or omitted, in order to enroll.
  - Organizational Unit: The Organizational Unit (OU) field is the name of the department or organization unit making the request. To skip the OU field, press Enter on the keyboard.
  - Common Name: The Common Name is the Host + Domain Name. It looks like "www.digicert.com" or "digicert.com".

### **4. Sign and Self-Sign the Certificates:**

- For internal use, self-sign the certificate.

### **5. Repeat for the 3rd Party Organization:**

- Follow the same steps to generate a separate private key and certificate for the 3rd party organization.

## **Uploading Certificates to Azure Key Vault**

After creating the certificates, you can upload them to Azure Key Vault:

### **1. Access Your Key Vault:**

- Within the Key Vault, go to the Certificates section.

### **2. Import or Generate a Certificate:**

- You can either import an existing certificate or generate a new one directly in Key Vault.

- To import, click on “Generate/Import” and then choose Import to upload your certificate file (`.pem` or `.pfx` file).

### **3. Provide Certificate Details:**

- Fill in the necessary: (\* are required fields) \*name of the certificate, type of CA, \*Subject, DNS Names, Validity period (in months), file type, Lifetime Action Type, \*Percentage Lifetime, Advanced Policy Configuration and tags.
- Press “Create”

### **4. Azure Permissions and Renewal:**

- Make necessary permissions.
- Keep track of the 3rd party's certificate validity period.
- Renew certificates before they expire.

#### **2.2.2 Orchestration and Workflow**

Does not apply.

#### **2.2.3 Extract Transform Load (ETL)**

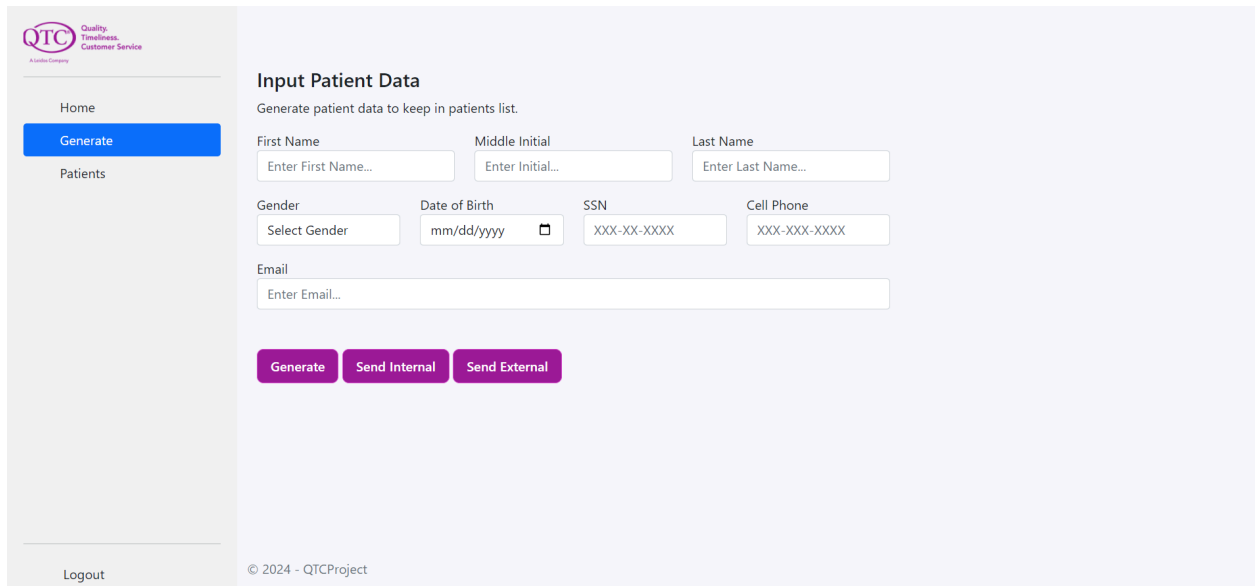
Does not apply.

#### **2.2.4 Robotic Process Automation (RPA)**

Does not apply.

## 2.3 User Interfaces (UI)

### 2.3.1 Pages



**QTC**<sup>®</sup> Quality. Timeliness. Customer Service.  
A Leidos Company

Home  
**Generate**  
Patients

**Input Patient Data**  
Generate patient data to keep in patients list.

First Name: Enter First Name... Middle Initial: Enter Initial... Last Name: Enter Last Name...

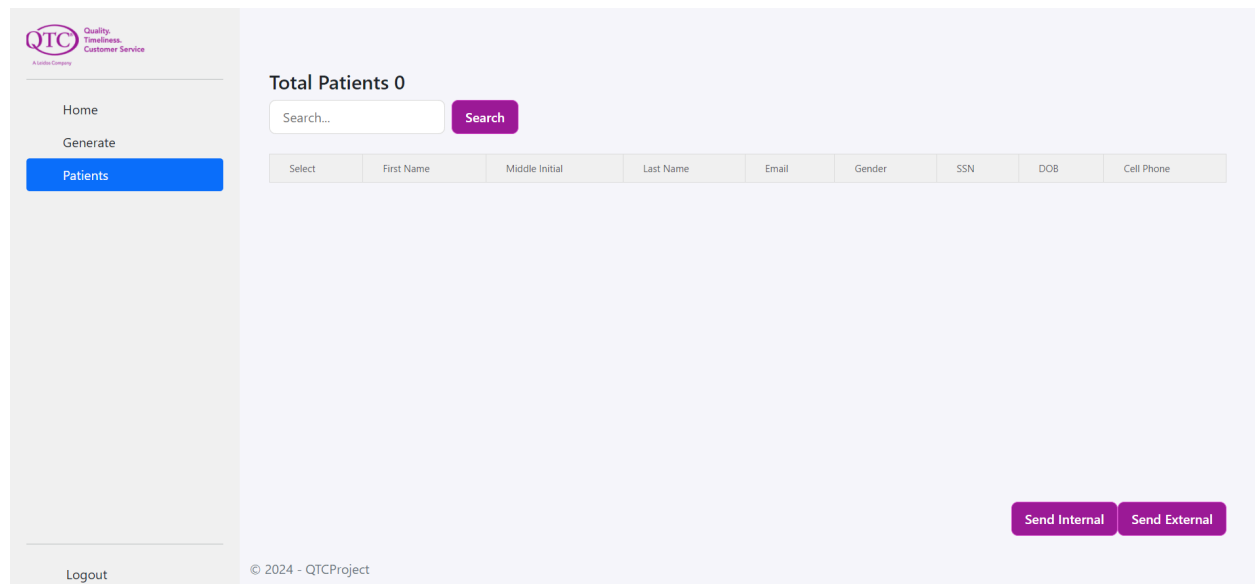
Gender: Select Gender Date of Birth: mm/dd/yyyy SSN: XXX-XX-XXXX Cell Phone: XXX-XXX-XXXX

Email: Enter Email...

**Generate** **Send Internal** **Send External**

Logout © 2024 - QTCProject

Figure 4: Generate Data Page



**QTC**<sup>®</sup> Quality. Timeliness. Customer Service.  
A Leidos Company

Home  
Generate  
**Patients**

**Total Patients 0**

Search... **Search**

Select	First Name	Middle Initial	Last Name	Email	Gender	SSN	DOB	Cell Phone
--------	------------	----------------	-----------	-------	--------	-----	-----	------------

**Send Internal** **Send External**

Logout © 2024 - QTCProject

Figure 5: View Data Page

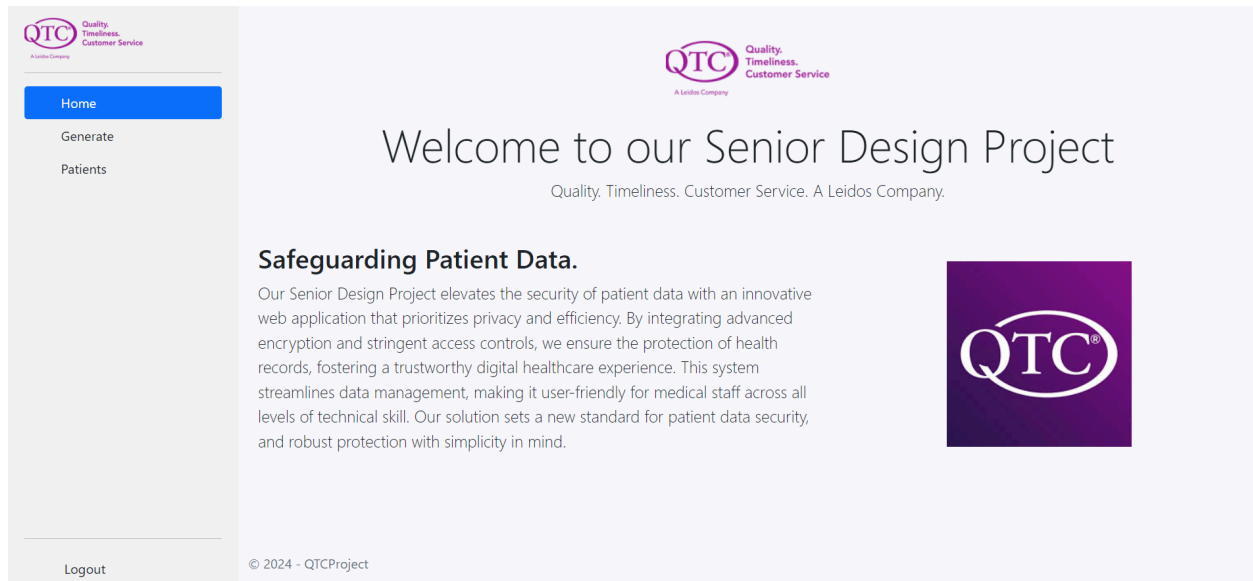


Figure 6: Home Page

Not applicable for the time being.

Environment	URL
DEV	
SQA	N/A
UAT	
PROD	

Table 10: Web 1 Links

Role	Permissions
RoleA	AView BView BEdit
RoleB	AView BView AEdit BEdit

Table 11: Web 1 Role-based Access Control (RBAC) Matrix

Logging/Tracing Data Store	[File; Database; Windows Event; Messaging]
Events Captured	[User Access Control; User Activity; Application Exception]
Audit Logging Review Process	[SOX Compliant; Manual Frequency; AIOPS]

Table 12: Web 1 Logging and Monitoring

The following sections describe the web controllers.

### 2.3.1.1 Account Controller

Account Controller - This controller handles the login page.

Action	Request Parameters	Request Body	Role	Permission	Model	View	Service
Login	Username Password	username admin input data patients login view				LoginPage	

Table 13: Account Controller Methods

### 2.3.1.2 Home Controller

Home Controller - This controller handles the landing page.

Action	Request Parameters	Request Body	Role	Permission	Model	View	Service
Index		View				Index	
Privacy		View				Privacy	
Error		Error View				Error View Model	

Table 13: Home Controller Methods

### 2.3.1.3 API Controller

API Controller - This is receiving and confirms that the data was received.

Action	Request Parameters	Request Body	Role	Permission	Model	View	Service
ReceiveData	Data	receive data data.ToString					

Table 13: API Controller Methods

### 2.3.1.4 Patient Controller

Patient Controller - This controller handles the generating of the patient data and all methods for the patient.

Action	Request Parameters	Request Body	Role	Permission	Model	View	Service
InputData		View			Patient	InputData	
GenerateData	Patient	addPatient Writeline InputData			Patient		
SearchPatient	Patient	View			Patient	SearchData	
SendDataAsync	patient action				Patient		

Table 13: Patient Controller Methods

#### 2.3.1.4.1 Aggregate APIs

Does not apply.

## 3 USE CASES

The following are the primary use cases represented as sequence diagrams with narrative and the accompanying user interface.

### 3.1 Use Case 1

The process involves the secure transmission of data containing sensitive information (such as Personal Health Information [PHI] and Personally Identifiable Information [PII]) between servers within a trusted domain. The communication channel is safeguarded using HTTPS to ensure encryption in transit. Concurrently, HTTP requests and responses are logged for auditing or analytical purposes. However, to maintain the confidentiality of sensitive data, measures are implemented to encrypt PHI/PII before the logs are stored. This ensures that while users can access these logs for legitimate purposes, they are unable to view or retrieve any sensitive personal information contained within the data payload.

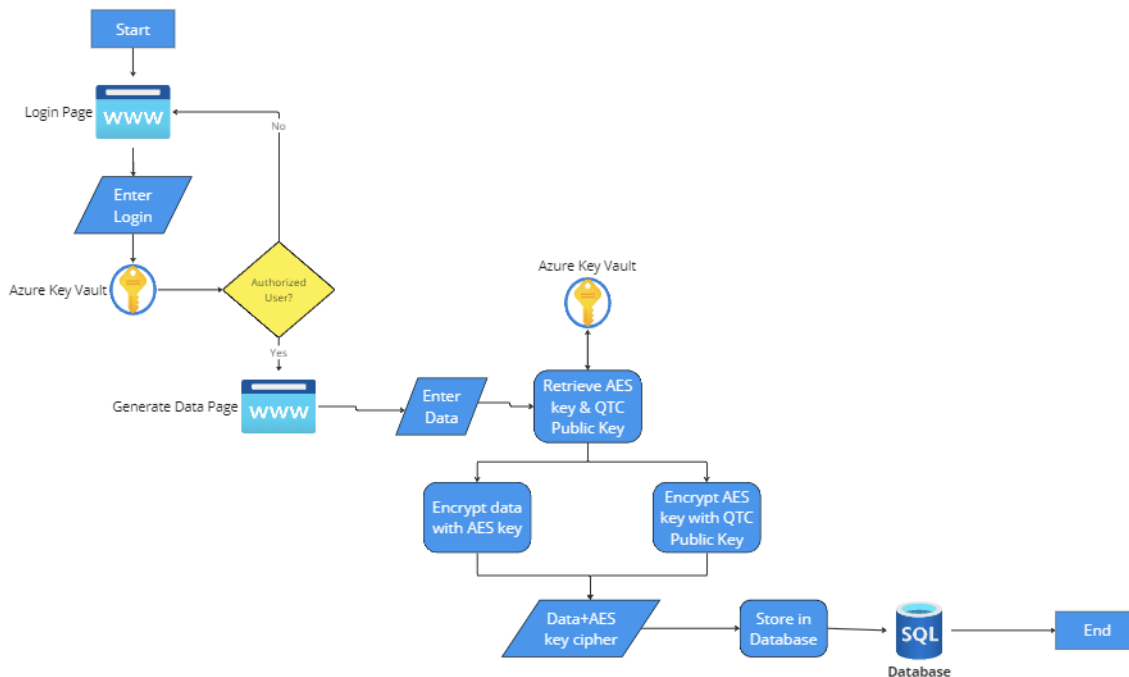


Figure 8: [Use Case 1] Sequence Diagram

Steps:

1. User inputs login credentials.

2. Authenticate credentials.
3. Display Home Page(Generate Data Page).
4. User enters data in the form.
5. Retrieve an available AES key and QTC's public RSA key from Azure Key Vault.
6. Encrypt the data using AES key.
7. Encrypt the AES key using the public key.
8. Store encrypted data and encrypted AES key in database.

---

## 3.2 Use Cases 2 and 3

---

**Use Case 2:** The procedure entails transferring data from a server within a trusted domain to an external server outside of this domain, using an HTTPS connection to ensure secure data transit. Similar to the internal process, all HTTP activity is meticulously logged, capturing the details of the data exchange without compromising security. Although the destination server lies outside the trusted domain, the integrity of the data in transit is maintained through HTTPS encryption. As with internal logging, any sensitive information within the payload, such as PHI/PII, is encrypted before logging to ensure it remains invisible and inaccessible to users who have permission to view the HTTP logs. This practice upholds privacy and security standards even when interacting with servers beyond the trusted domain.

**Use Case 3:** In this scenario, data is being transmitted from a server within a trusted domain to another server situated outside the trusted domain. The data transfer may pass through an unspecified number of intermediary systems. Throughout this transmission chain, each segment of the data's journey is secured using HTTPS connections to ensure encryption and integrity of the data in transit. As is standard practice, HTTP activities—including those across the intermediary systems—are comprehensively logged. These logs detail the data exchange process without compromising the encrypted state of sensitive content. Precautions are taken to obscure or strip any Personal Health Information (PHI) or Personally Identifiable Information (PII) from these logs, safeguarding the sensitive data from exposure to users who have access to these logs, thus maintaining privacy and compliance with data protection regulations.

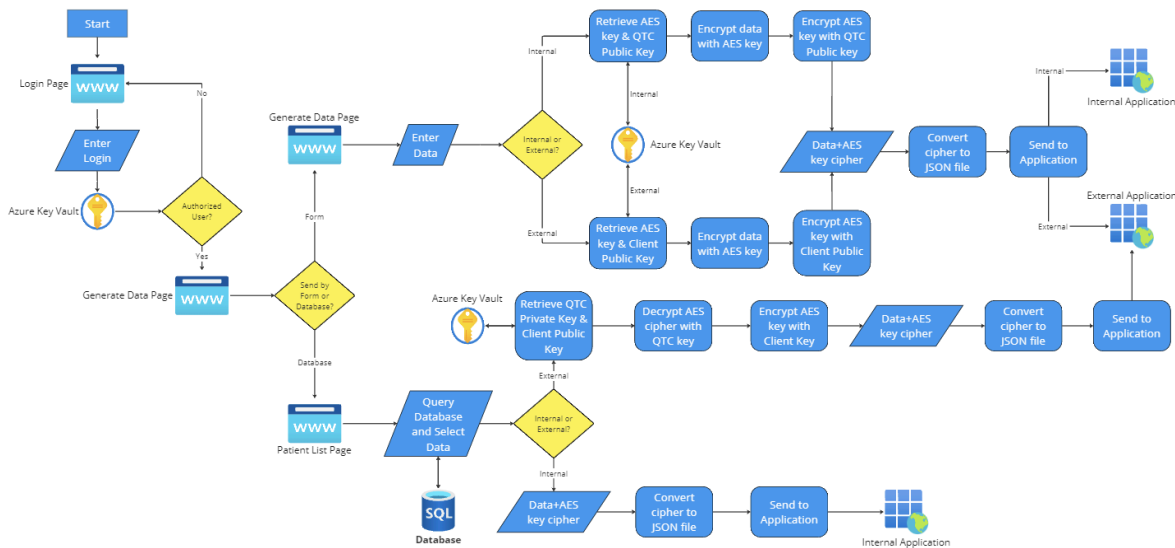


Figure 9: [Use Case 2 & 3] Sequence Diagram

#### Steps (On the Fly):

1. Input Login.
2. Authenticate credentials.
3. Display Home Page (Generate Data Page).
4. Enter data in the form.
5. Retrieve an available AES key from Azure Key Vault.
6. If sending to an internal application (Use Case 2):
  - Retrieve company's public RSA key from Azure Key Vault.
7. If sending to an external application (Use Case 3):
  - Retrieve the third party's public RSA key from Azure Key Vault.
8. Encrypt the data using the AES key.
9. Encrypt the AES key with the public RSA key.
10. Convert encrypted data and encrypted AES key into JSON file.
11. Send to internal application.

#### Steps (Sending from Database):

12. Input Login.
13. Authenticate credentials.
14. Display Home Page (Generate Data Page).
15. Click on View Patients Page.
16. Enter a search query to find data.
17. Select data.
18. If sending to internal application:
  - Data is unmodified.
19. If sending to an external application:

- Retrieve company's private key from Azure Key Vault.
  - Decrypt AES key with private key.
  - Retrieve the third party's public key.
  - Encrypt AES key with public key.
20. Convert encrypted data and encrypted AES key into JSON file.
21. Exchange certificates with Azure Key Vault.
22. Authenticate the 3rd party:
- If authenticated, the encrypted data is exchanged and authenticated.
  - If not authorized, the encrypted data is not exchanged.

## 4 APPENDIX

**PHI/PII** - Personally Identifiable Information.

**RSA** - Rivest–Shamir–Adleman.

**AES** - Advanced Encryption Standard.

**HTTPS** - Hypertext Transfer Protocol Secure

**MS SQL-** Microsoft SQL Server

**HIPAA** - Health Insurance Portability and Accountability Act

**Key Management** - Azure Key Vault allows users to generate, import, and manage cryptographic keys used for encryption and decryption in applications and services.

**Secrets Management** - Enables the secure storage and management of application secrets, such as connection strings, API keys, and passwords.

**Certificate Management:** Azure Key Vault allows the storage and management of digital certificates used for secure communication and authentication in applications.

**Access Control** - Role-based access control (RBAC) and permissions can be configured to control who can access and manage the stored keys, secrets, and certificates.