Hamdard University
Department of Computing
Final Year Project

CryptoCamGuard – Elevating Image Security

(FYP-008/FL24)

Software Requirements Specifications

Submitted by
Sardar Nazeer (2630-2021)
Ali Sher Siyal (2201-2021)

Supervisor
Mr. Saifullah Adnan

Fall 2021

Document Sign off Sheet

Document Information

Project Title	CryptoCamGuard Elevating Image Security
Project Code	FYP-008/FL24
Document Name	Software Requirements Specifications
Document Version	<1.0>
Document Identifier	FYP-008/FL24-SRS
Document Status	Draft
Author(s)	Sardar Nazeer, Ali Sher Siyal
Approver(s)	Saifullah Adnan
Issue Date	<Date of issuance of this document>

Name	Role	Signature	Date
Sardar Nazeer	Team Lead		
Ali Sher Siyal	Team Member 2		
Saifullah Adnan	Supervisor		

Revision History

Date	Version	Description	Author
<dd/mmm/yyyy>	1.0	First Draft	<name>

Definition of Terms, Acronyms, and Abbreviations

Term	Description
AES	Advanced Encryption Standard
CryptoCamGuard	The image security application under development.
QR Code	Quick Response System

GB	Gigabyte
SSD	Solid-State Drives
HTTPS	Hypertext Transfer Protocol Secure
API	Application Programming Interface
PDF	Portable Document Format
DOC	Document
SHA-256	Secure Hash Algorithm (256-bit)

Table of Contents

1. Introduction 6
1.1 Purpose of Document 6
1.2 Intended Audience 6
1.3 Abbreviations **Error! Bookmark not defined.**
2. Overall System Description 7
2.1 Project Background 7
2.2 Problem Statement 7
2.3 Project Scope 7
2.4 Not In Scope 7
2.5 Project Objectives 8
2.6 Stakeholders & Affected Groups 8
2.7 Operating Environment 8
2.8 System Constraints 8
2.9 Assumptions & Dependencies 9
3. External Interface Requirements 10
3.1 Hardware Interfaces 10
3.2 Software Interfaces 10
3.3 Communications Interfaces 10
4. System Functions / Functional Requirements 11
4.1 System Functions 11
4.2 Use Cases 11
4.2.1 List of Actors **Error! Bookmark not defined.**
4.2.2 List of Use Cases **Error! Bookmark not defined.**
4.2.3 Use Case Diagram 12
4.2.4 Description of Use Cases 13
5. Non - Functional Requirements 16
5.1 Performance Requirements 16
5.2 Safety Requirements 16
5.3 Security Requirements 16
5.4 Reliability Requirements 16
5.5 Usability Requirements 16
5.6 Supportability Requirements 16
5.7 User Documentation 16
6. References 17

1. Introduction

CryptoCamGuard is a image security application designed to protect sensitive digital images from unauthorized access. The app leverages advanced encryption technologies to ensure secure storage, sharing, and management of visual content. With rising concerns over data breaches and image misuse, CryptoCamGuard addresses the critical need for enhanced privacy, making it an ideal solution for photographers, content creators, businesses, and privacy-conscious individuals. Its user-friendly interface and robust security features offer a reliable safeguard against digital threats while ensuring ease of use for non-technical users.

.1 Purpose of Document

This document specifies the functional and non-functional requirements for CryptoCamGuard, a secure image encryption and sharing application.

.2 Intended Audience

The purpose of this project is to develop CryptoCamGuard, an image security app designed for photographers, content creators, and privacy-conscious users. It ensures secure image storage and transmission through advanced encryption, preventing unauthorized access and safeguarding digital assets.

- Developers
- Testers
- Supervisors
- End Users

- Overall System Description

.1 Project Background

In today's digital era, image security has become a growing concern due to the rising incidents of data breaches, unauthorized access, and misuse of personal media. Content creators, photographers, and privacy-conscious users often face challenges in safeguarding their digital assets, especially when sharing images online or storing them on cloud platforms.

The **CryptoCamGuard (CCG)** project aims to address these vulnerabilities by providing a secure image management solution with advanced encryption techniques. It ensures that only authorized users can access, share, or store images, offering a robust defense against data leaks and unauthorized sharing.

CCG is designed to be user-friendly while integrating cutting-edge security measures, making it ideal for both professional and personal use cases. This project emphasizes privacy, control, and ease of access, ensuring users can protect their visual content with confidence.

.2 Problem Statement

With the increasing use of digital images across various platforms, users face significant risks of unauthorized access, data theft, and image misuse. Existing image security solutions often lack robust encryption and easy-to-use features, leaving sensitive content vulnerable. There is a need for a secure, reliable, and user-friendly application that ensures the protection of images throughout their lifecycle, from storage to sharing, especially for photographers, content creators, and privacy-conscious individuals. CryptoCamGuard aims to fill this gap by providing a comprehensive solution to safeguard digital images against security threats.

.3 Project Scope

CryptoCamGuard will focus on:

- Real-time image encryption.
- Integration with existing surveillance systems.
- User-friendly interface for encryption settings.
- Performance optimization to ensure minimal latency.

.4 Not In Scope

The project will not cover:

- Audio data encryption.
- Cloud storage solutions.
- Integration with non-digital (analog) surveillance systems.

.5 Project Objectives

The primary goal of CryptoCamGuard is to empower users to take control of their digital privacy by offering a secure solution for photo management.

.6 Stakeholders & Affected Groups

Primary Stakeholders:

1. **End Users**
2. **App Developers**
3. **Product Managers**

Secondary Stakeholders:

1. **Marketing and Sales Teams**
2. **Legal and Compliance Teams**
3. **Security Experts**

.7 Operating Environment

The system will operate in a browser-based environment and should support the following:

- **Browsers:** Chrome, Firefox, Safari.
- **Devices:** Desktop, laptop, tablet, and smartphone.

.8 System Constraints

1. **Performance**:
   - Encryption may impact performance on lower-end devices.
   - High storage requirements for encrypted images.
2. **Compatibility**:
   - Need for cross-platform compatibility (Android, iOS, Web).
   - Variability in device performance (e.g., RAM, processor).
3. **Network**:
   - Slow data transfer speeds on low-bandwidth connections.
4. **Security**:
   - Encryption can introduce delays in processing.
   - Compliance with data protection laws (e.g., GDPR, CCPA).
5. **Scalability**:
   - Handling large volumes of image data and user growth.
6. **Usability**:

- Need for a user-friendly interface despite security features.
- Consistent experience across platforms.

7. **Budget/Resources**:
   - High development and maintenance costs.
   - Limited time for development and updates.
8. **Legal/Compliance**:
   - Adherence to privacy laws and regulations, affecting features and data management.

.9 Assumptions & Dependencies

Assumptions:

1. Users are aware of the need for image security.
2. The app will be compatible with recent Android and iOS devices.
3. Cloud services for storage will be reliable.
4. Users will have internet access for syncing and sharing images.
5. User-uploaded images will be valid and not corrupted.
6. The app will comply with privacy regulations (e.g., GDPR, CCPA).
7. Stable encryption libraries will be available and updated.

Dependencies:

1. Cloud services (e.g., AWS, Azure) for image storage.
2. Updates to mobile OS (Android, iOS).
3. Encryption libraries for data security.
4. Third-party authentication services (e.g., OAuth, biometrics).
5. Legal compliance with data protection laws.
6. Stable internet access for syncing and sharing.

- External Interface Requirements

.1 Hardware Interfaces

- **Server Requirement**
- Processor: 2 GHz or higher.
- RAM: 8 GB minimum.
- Storage: 100 GB SSD.
- **Client Requirement**
- Any modern device that supports web browser.

.2 Software Interfaces

- **Server Requirement**
- **Backend**: Node.js, Express.js
- **Database**: MongoDB (Mongoose for connection)
- **Frontend**: React.js

.3 Communications Interfaces

- HTTPS for secure data transmission.
- REST APIs for interaction between components.

- System Functions / Functional Requirements

.1 System Functions

Ref #	Functions	Category	Attribute	Details & Boundary Constraints
R1.1	User registration and login	Evident	Response time	Registration should complete within 5 seconds.
R1.2	Image encryption using AES-256	Evident	Security	Ensures secure image storage and transfer.
R1.3	Secure sharing with access controls	Evident	Functionality	Users can specify access permissions.
R1.4	Access monitoring and logging	Hidden	Auditability	Logs access attempts and usage statistics.
R1.5	Cloud-based storage with encryption	Evident	Scalability	Supports dynamic storage allocation.

.2 Use Cases

List of Actors:

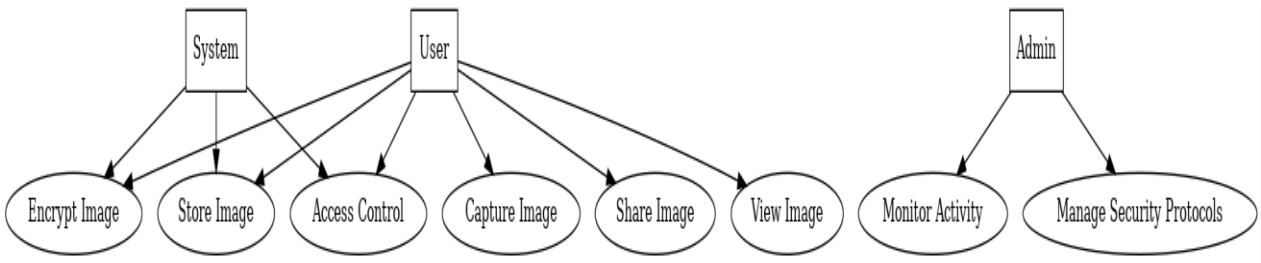
- User
- System Administrator

List of Use Cases:

- Upload and encrypt an image.
- Detect unauthorized access attempts.
- Retrieve and decrypt images.

Use Case #	Name	Brief Description
UC1	Account Registration / Login	Allows users to register securely and create unique profiles.
UC2	Upload Image	Users can upload images for encryption and secure storage.
UC3	Retrieve Image	Users can decrypt and retrieve their encrypted images.
UC4	Detect Intrusions	The system detects and alerts users of unauthorized access.
UC5	Manage Accounts	Administrators can manage user accounts and system settings.

.2.1 Use Case Diagram



.2.2 Description of Use Cases

Section: Main	
Name:	**Account Registration / Login**
Actors:	Users
Purpose:	To allow users with a secure account to access the system.
Description:	Users can create an account or log in using valid credentials.
Cross References:	Functions: R1.1
Pre-Conditions	User must have valid email and password for login or registration information.
Successful Post-Conditions	Upon a successful login, the user is taken to their dashboard.
Failure Post-Conditions	Login fails and error message is displayed.

Typical Course of Events			
Actor Action			**System Response**
1	User enters registration details or login credentials.		System verifies the credentials and redirects to the appropriate dashboard.
2	Incorrect credentials are provided.		System displays an error message and asks the user to retry or reset password.

Section: Main	
Name:	**Upload and Encrypt Image**
Actors:	User
Purpose:	To allow users to upload images and secure them with encryption.
Description:	Users upload images, which are encrypted and stored in the system.
Cross References:	Functions: R1.2

Pre-Conditions	User must be logged into their account.
Successful Post-Conditions	Image is encrypted and stored successfully.
Failure Post-Conditions	Image upload fails due to unsupported format or network error.

Typical Course of Events

	Actor Action		System Response
1	User selects a image file and clicks upload.		System validates the image format and stores the image in the database.
2	Invalid file format is selected.		System displays an error message and asks the candidate to re-upload.

Section: Main	
Name:	**Share Image Securely**
Actors:	Users, Third-party Viewers
Purpose:	To allow users to share images with specified permissions.
Description:	Users select images to share and set access permissions for specific viewers.
Cross References:	Functions: R1.3
Pre-Conditions	User must have encrypted images stored in the system.
Successful Post-Conditions	Image is accessible only to specified viewers.
Failure Post-Conditions	Sharing fails if permissions are invalid or there is a system error.

Typical Course of Events

	Actor Action		System Response
1	User select image and clicks “Share”		System validates the information and grant access for sharing.
2	If image is not stored in the system.		System displays an error message and asks the user to complete missing fields.

Section: Main	
Name:	**Monitor Access**

Actors:	User
Purpose:	To allow users to track access to their shared images.
Description:	Users can view access logs, including time and viewer details.
Cross References:	Functions: R1.4
Pre-Conditions	Images must be shared.
Successful Post-Conditions	Access logs are displayed.
Failure Post-Conditions	Logs fail to display due to a system error.

Typical Course of Events

	Actor Action		System Response
1	To allow users to track access to their shared images.		System will verify and displays the images.

Section: Main	
Name:	**Manage Permissions**
Actors:	User
Purpose:	To allow users to update or revoke access permissions for shared images.
Description:	Users modify access settings or revoke permissions as needed.
Cross References:	Functions: R1.5
Pre-Conditions	Images must be shared with permissions.
Successful Post-Conditions	Updated permissions are enforced.
Failure Post-Conditions	Update fails due to a system error.

Typical Course of Events

	Actor Action		System Response
1	User selects a shared image and modifies permissions.		System validates and updates permissions.
2	If the update fails, the system displays an error message.		System records the user performance and saves the results.

- Non - Functional Requirements

.1 Performance Requirements

- The system should handle up to 500 concurrent users without significant performance degradation.
- Each operation, including encryption and sharing, should complete within 2 seconds.

.2 Safety Requirements

- The application must prevent unauthorized access by using industry-standard encryption methods.
- All image data transfers must occur over HTTPS.

.3 Security Requirements

- Passwords must be hashed and stored using SHA-256.
- Two-factor authentication should be available for all user accounts.
- Encrypted images must not be accessible without valid decryption keys.

.4 Reliability Requirements

- The system must have an uptime of 99.5% or higher.
- Backup services should run daily to ensure data recovery in case of a failure.

.5 Usability Requirements

- The user interface should be intuitive and accessible, with clear guidance for all primary functions.
- The application should be responsive across various devices, including desktops, tablets, and smartphones.

.6 Supportability Requirements

- The application should support future enhancements without major overhauls.
- Cloud-based architecture must allow scalability to accommodate increased user loads.

.7 User Documentation

- Comprehensive user guides must be provided for all user roles (e.g., Users, Administrators).
- FAQs and troubleshooting steps should be easily accessible within the app.

- References

List References

1. Stallings, W. (2017). Cryptography and Network Security Principles and Practice. Pearson.
2. Schneier, B. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley.
3. IEEE Standards Association. (2020). IEEE 802.11 Wireless LAN Standards.
4. OWASP Foundation. (2023). OWASP Application Security Verification Standard (ASVS).
5. Amazon Web Services (AWS) Documentation. Retrieved from: https://docs.aws.amazon.com/