CryptoCamGuard-Elevating Image Security

Final Year Project Report

Submitted by

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

Supervisor

Mr. Saifullah Adnan

In partial fulfilment of the requirements for the degree of
Bachelor of Science in Software Engineering
2025

Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology

Hamdard University, Main Campus, Karachi, Pakistan

Certificate of Approval

Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology
Hamdard University, Karachi, Pakistan

This project "**CryptoCamGuard – Elevating Image Security App**" is presented by Sardar Nazeer, Ali Sher Siyal under the supervision of their project advisor and approved by the project examination committee, and acknowledged by the Hamdard Institute of Engineering and Technology, in the fulfillment of the requirements for the Bachelor degree of Software Engineering.

_____
Mr. Saifullah Adnan
(Project Supervisor)

_____
In-charge FYP-Committee

_____
(Project Co-Supervisor)

_____
Chairman
(Department of Computing)

_____
(Dean, FEST)

Authors' Declaration

We declare that this project report was carried out in accordance with the rules and regulations of Hamdard University. The work is original except where indicated by special references in the text and no part of the report has been submitted for any other degree. The report has not been presented to any other University for examination.

Dated:

Authors Signatures:

_____

Sardar Nazeer

_____

Ali Sher Siyal

Plagiarism Undertaking

We, Sardar Nazeer, Ali Sher Siyal solemnly declare that the work presented in the Final Year Project Report titled **CryptoCamGuard – Elevating Image Security** has been carried out solely by ourselves with no significant help from any other person except few of those which are duly acknowledged. We confirm that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced.

Dated:

Authors Signatures:

_____

Sardar Nazeer

_____

Ali Sher Siyal

Acknowledgments

We are deeply grateful to our supervisor, **Mr. Saifullah Adnan**, for his invaluable guidance, support, and encouragement throughout this project. We also extend our gratitude to the faculty members of **Hamdard Institute of Engineering and Technology** for their continuous support. Finally, we would like to thank our families and friends for their patience and encouragement during this journey.

Document Information

Table 1: Document Information

Customer	
Project Title	CryptoCamGuard – Elevating Image Security
Document	Final Year Project Report
Document Version	1.0
Identifier	FYP-008/FL24 Final Report
Status	Final
Author(s)	Sardar Nazeer, Ali Sher Siyal
Approver(s)	Saifullah Adnan
Issue Date	

Definition of Terms, Acronyms, and Abbreviations

This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.

Table 2: Definition of Terms, Acronyms, and Abbreviations

Abstract

The widespread use of digital images in modern communication and sharing platforms often exposes users to privacy and security risks. Unauthorized access, data breaches, and misuse of sensitive images are becoming increasingly prevalent. To address these challenges, **CryptoCamGuard** provides a robust mobile application that ensures the security of images through **AES-256 encryption**, secure sharing with customizable access controls, and real-time monitoring of image usage. This app prioritizes user-friendliness, scalability, and high-level security to redefine how images are shared and protected in a digital ecosystem.

Keywords: Image Security, AES-256 Encryption, Secure Sharing, Digital Privacy

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Motivation

The CryptoCamGuard project is an innovative mobile application designed to secure personal photos through advanced encryption techniques. This application empowers users to take control of their digital privacy by providing a safe and private platform to capture, encrypt, and store images. By prioritizing security, user-friendliness, and performance, the project addresses growing concerns about unauthorized access, privacy breaches, and image theft in the digital age.

1.2 Problem Statement

In the digital era, mobile devices have become an integral part of everyday life, serving as repositories for personal memories and sensitive information. Among the most frequently stored data are images, which often include private moments, family photos, and confidential documents. Unfortunately, the security measures in place for managing these images on mobile devices are inadequate.

1.3 Goals and Objectives

The primary objective of CryptoCamGuard is to provide users with a secure platform for capturing, encrypting, and storing personal images, thereby ensuring their privacy and protection from unauthorized access. The application aims to streamline the photo management process while empowering users to maintain full control over their digital memories.

1.4 Project Scope

The following significant elements will be a part of the project.

- **In Scope**:

    1. **Advanced Encryption:** Secures images upon capture with AES encryption algorithms.
    2. **User-Friendly Interface:** Intuitive design for easy navigation by all users.
    3. **Secure Image Storage:** Safely stores images within the app, away from traditional galleries.
    4. **Authentication Mechanisms:** Ensures only authorized users can access their images.
    5. **Optimized Performance:** Fast and reliable handling of photos for efficient uploads and downloads.

CHAPTER 2

RELEVANT BACKGROUND & DEFINITIONS

Traditional photo management solutions, such as mobile galleries or cloud services, do not offer sufficient protection against these risks. Many of these platforms store images in unencrypted formats, making them easily accessible to unauthorized users if a device is lost or hacked. Even when encryption is applied, it is often implemented only during data transmission, while the images themselves remain exposed during storage. This lack of comprehensive security measures has led to a growing demand for more secure solutions that can

guarantee the privacy and protection of personal images. In response to these challenges, CryptoCamGuard was conceived as a mobile application that offers a secure platform for capturing and storing images. By employing advanced encryption techniques, the app ensures that personal photos are not only protected during transmission but also stored in a fully encrypted format. Unlike traditional solutions, CryptoCamGuard enables users to take photos directly through the app and encrypt them immediately, ensuring that these images remain secure throughout their lifecycle.

MANUAL PROCUREMENT: Manual procurement refers to the traditional process of sourcing goods and services through manual paperwork, phone calls, or in-person meetings. This method is often inefficient, time consuming, and prone to errors due to a lack of digital automation.

LIMITED TRANSPARENCY: In conventional image storage solutions, users often face limited transparency regarding who can access their photos and how their data is managed. This lack of visibility increases the risk of unauthorized access and data misuse, as users are unaware of potential privacy breaches.

INEFFICIENCY: Traditional image management methods often involve manual processes that are time consuming and prone to errors, leading to delays in accessing or organizing photos. This inefficiency can frustrate users, making it difficult to manage and retrieve their personal images quickly and securely.

Chapter 3: Literature Review and Related Work

Introduction

The digital age has brought forth a surge in image sharing and online data storage. While this has improved convenience, it has also exposed users to significant risks regarding data privacy and security. The need for secure image management systems has never been more urgent, especially with the rise of hacking incidents, data leaks, and unauthorized access to private information.

This chapter explores existing research, technologies, and applications relevant to CryptoCamGuard. It also identifies gaps in current systems and justifies the need for a secure, performance-optimized, and user-friendly solution.

Related Technologies and Security Protocols

AES (Advanced Encryption Standard)

AES is a widely adopted symmetric encryption algorithm recognized for its speed and security. AES-256, used in CryptoCamGuard, provides robust protection for digital data and is recommended by NIST for encrypting sensitive information. It offers a strong security foundation against brute-force and dictionary attacks.

JWT (JSON Web Tokens)

JWT is an open standard used for securely transmitting information between parties. In CryptoCamGuard, JWT is used for user session management, ensuring secure and stateless authentication.

HTTPS & TLS

To ensure secure communication between the client and server, CryptoCamGuard uses HTTPS protocol layered with TLS (Transport Layer Security). This prevents data tampering and eavesdropping during image metadata transmission.

Review of Existing Applications

SecureCam (2020)

A mobile app focused on live camera feeds with basic encryption features. However, it lacked strong local storage encryption and did not support offline functionality.

CrypPic (2022)

An image encryption app that used AES but lacked user experience enhancements. It had performance issues on older devices and didn't offer responsive design.

Tresorit

A cloud-based file encryption platform that provides end-to-end security. While powerful, it is enterprise-focused and not tailored to image-specific mobile experiences.

PixelKnot

An Android app that uses steganography to hide messages within images. While creative, it is intended for message secrecy, not full image encryption or secure storage.

Gap Analysis

After reviewing several existing solutions, we observed key shortcomings. Apps like SecureCam and CrypPic offer encryption features but either lack a proper user experience or fail to store images securely. Tresorit is focused on enterprise-level cloud encryption and doesn't cater specifically to mobile image protection. PixelKnot offers a different concept (steganography) but lacks full encryption and storage functionality.

In contrast, CryptoCamGuard stands out by combining multiple essential features — including AES-256 encryption, offline functionality, secure local storage, optimized performance, and a clean Figma-based UI — in one mobile application. It also introduces an admin dashboard for monitoring, which is missing in all compared solutions.

These comparisons clearly show the gap in current offerings, highlighting the need for an all-in-one secure image application like CryptoCamGuard. either focus on one aspect (e.g., encryption or cloud sync) but fail to offer a holistic, user-centered, and secure environment like CryptoCamGuard does.

Research Insights

Recent academic research has stressed the importance of local image encryption to prevent data interception. Studies also highlight the role of usability in encouraging users to adopt secure apps. CryptoCamGuard bridges this divide by integrating strong encryption with a seamless UI.

A review of NIST guidelines for mobile data protection further validated our choice of AES-256 and secure local storage over reliance on cloud-based options.

Summary

The literature and tools reviewed demonstrate that while partial solutions exist, a gap remains in providing an all-in-one secure image management tool for everyday users. CryptoCamGuard emerges as a unique solution that combines military-grade encryption, intuitive design, and robust backend architecture.

Chapter 4: Project Discussion

Methodology

PROJECT DESIGN AND ARCHITECTURE

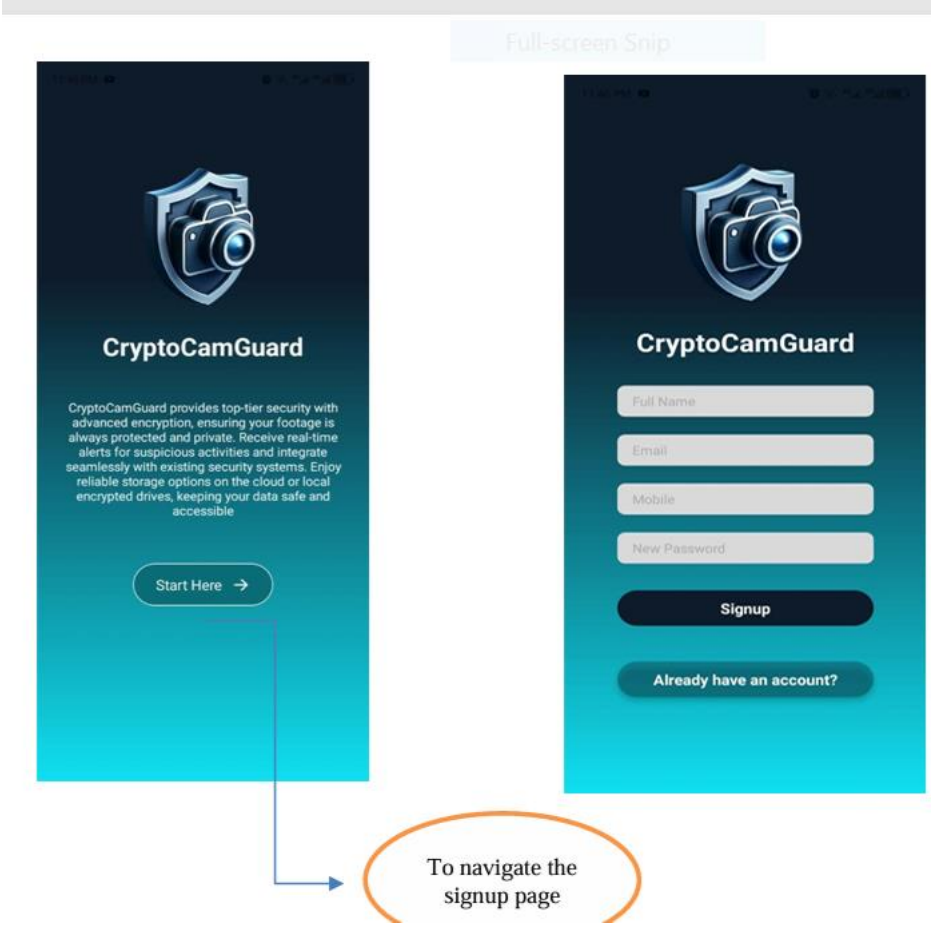
The design and architecture of CryptoCamGuard are centered around creating a secure, efficient, and user-friendly platform for capturing, encrypting, and storing personal images. The architecture leverages a client-server model, where the mobile application interacts with the backend server to ensure secure communication, data storage, and user authentication. Below is a breakdown of the key components of the system's design and architecture:

Client-Side (Mobile Application)

The Client-Side of CryptoCamGuard is the user-facing component of the system, built using React Native to ensure compatibility with both iOS and Android platforms. The mobile application is designed to handle all interactions with the user, from account management (signup and login) to the core functionalities of image capture and encryption. Below is a breakdown of the key processes and features handled on the client-side.User Interface (UI): The UI is designed with an intuitive, user-friendly approach, allowing users to easily navigate between functionalities such as capturing images, viewing encrypted images, and managing their accounts.

Signup and Login Functionality

The first interaction users have with CryptoCamGuard is through the signup and login process. This ensures that only authenticated users can access the application's features, especially the secure image storage functionality.'



Signup Process:

- Users create an account by providing basic details such as their email address and password.
- The system applies password strength rules to ensure security, requiring users to create strong passwords.
- Once the information is submitted, a secure API call is made to the backend, where the user's data is stored securely after hashing the password.
- Upon successful signup, the user is prompted to log in.

Login Process:

- For returning users, the login screen allows them to authenticate using their email and password.
- Once the user enters valid credentials, the app sends an authentication request to the backend, where **JWT (JSON Web Tokens)** are used for secure authentication.
- If the login is successful, the backend issues a JWT, which is stored locally on the user's device for the session duration. This token is required for accessing protected routes in the

app, such as image capture and viewing encrypted photos.

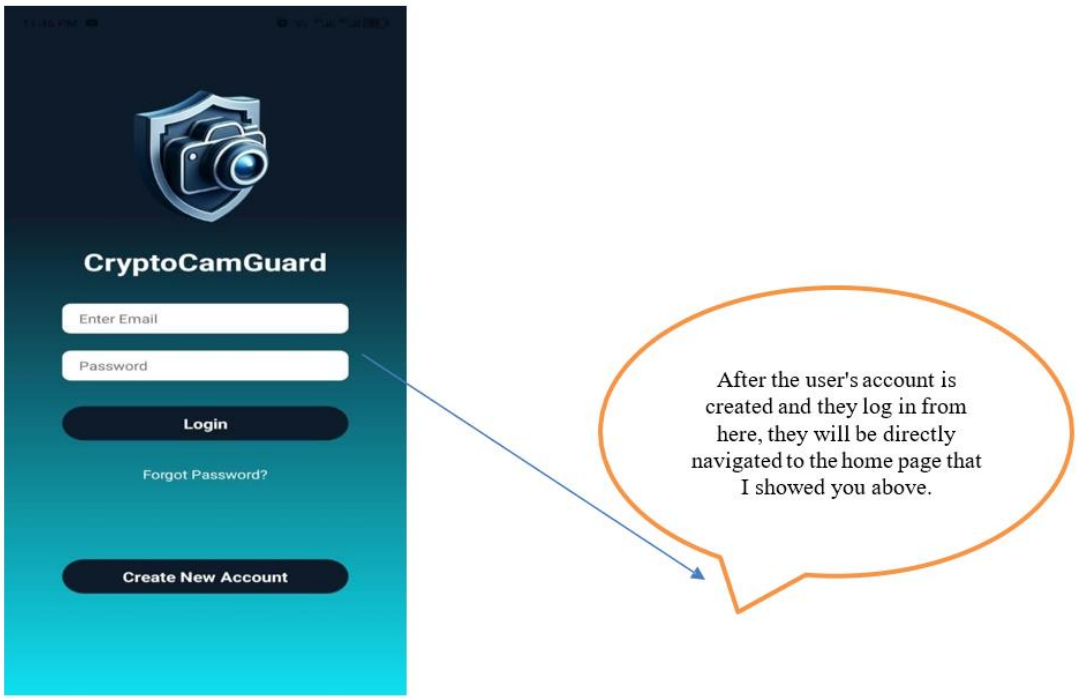
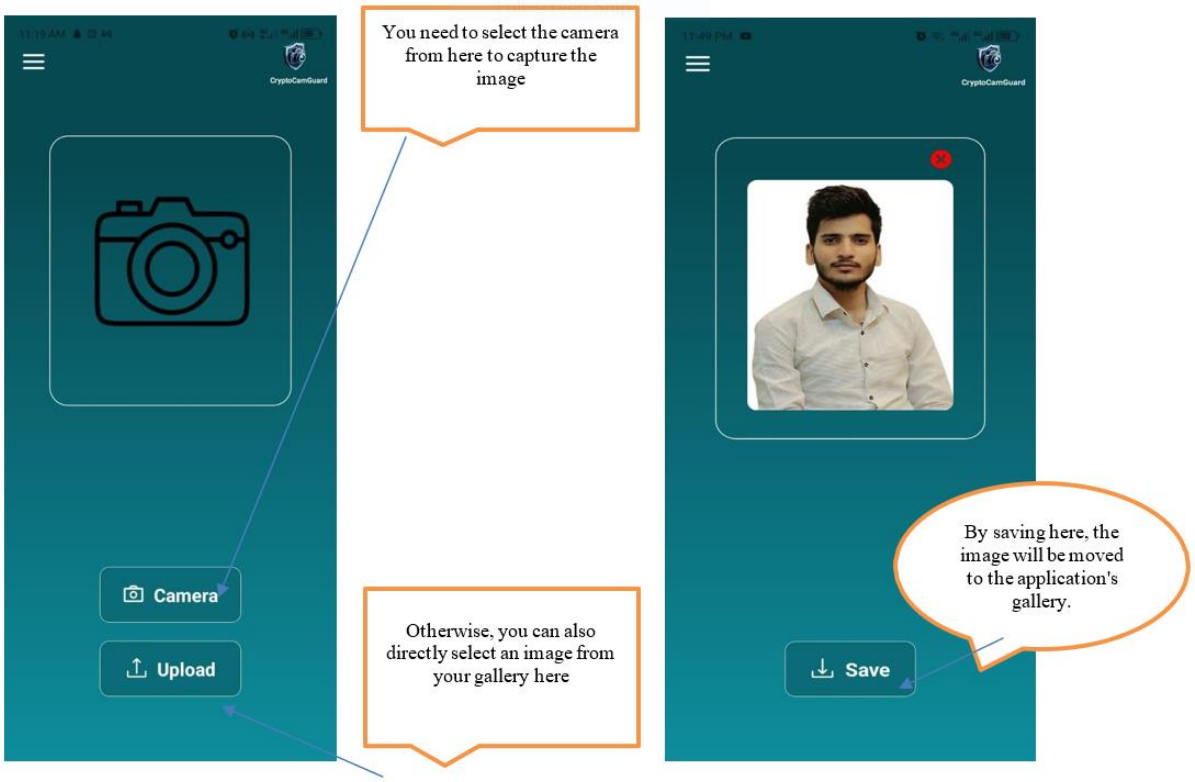


Image Capture Functionality

Once authenticated, users can access the core functionality of **CryptoCamGuard**: capturing and encrypting images.

- ***Accessing the Camera****:*
    - Users are presented with a camera interface within the app, allowing them to take photos without leaving the application.
    - The app leverages the native camera capabilities of the mobile device, ensuring high-quality image capture while maintaining a smooth user experience.
    - The camera module is integrated using **React Native's Camera APIs**, which provide control over resolution, focus, and other camera settings.
- ***Taking a Picture****:*
    - Once the user takes a photo, it is not stored in the device's gallery. Instead, the image is processed within the app and is prepared for encryption.
    - The user can choose to capture additional photos or proceed to encryption immediately.

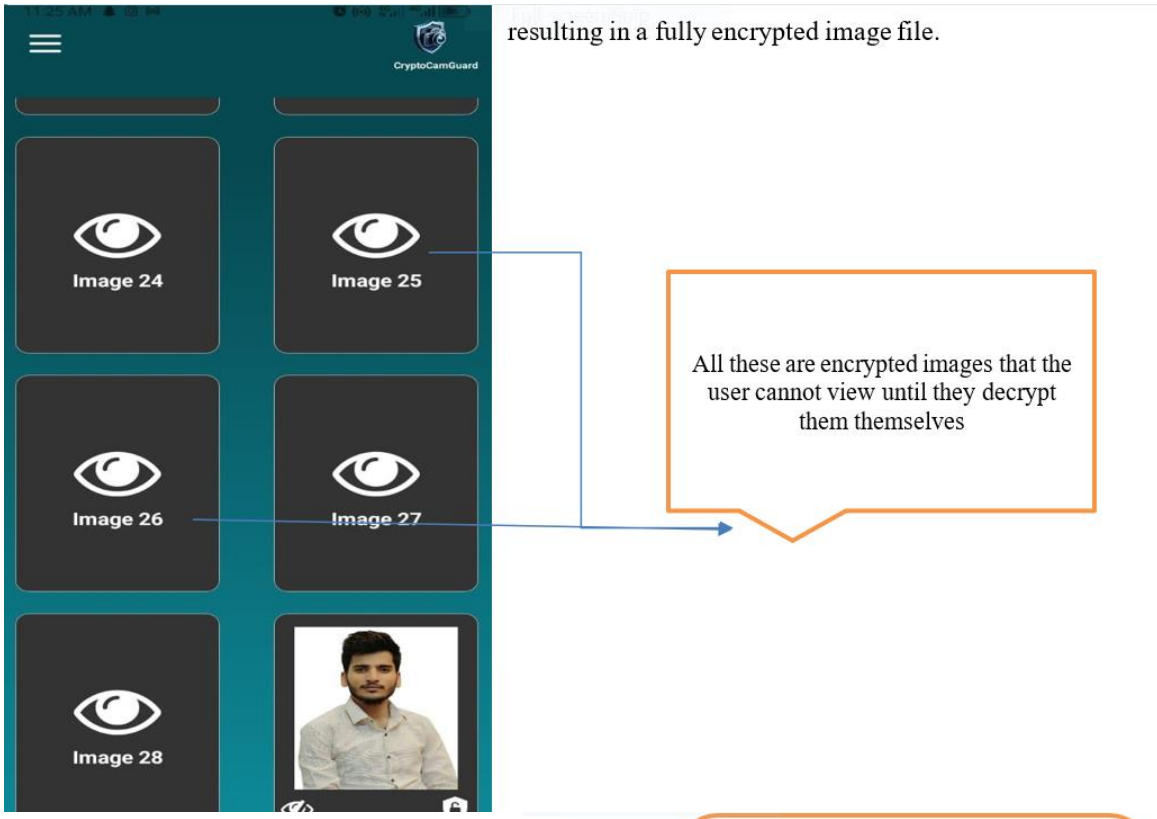


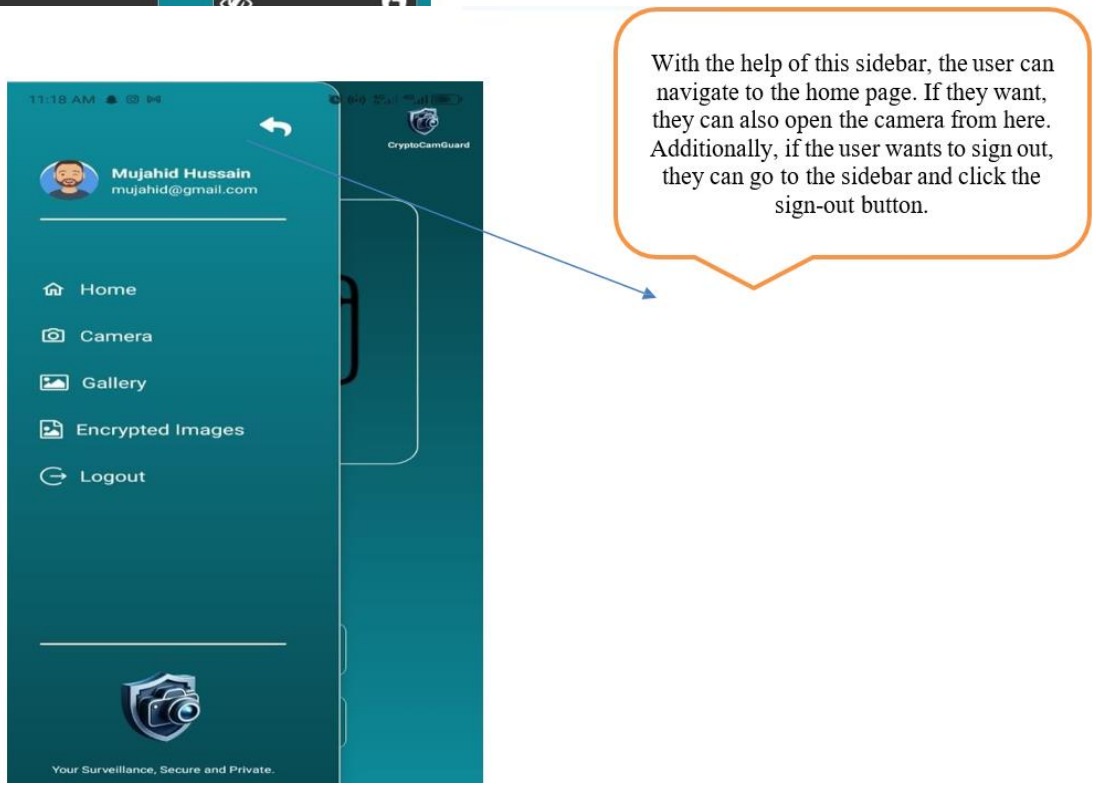
Encryption Process

After the image is captured, **CryptoCamGuard** immediately encrypts the photo to ensure that it is securely stored and inaccessible to unauthorized users.

- ***Encryption:***
    - The app utilizes **(Advanced Encryption Standard)**, which is one of the most secure encryption algorithms available. This ensures that even if the photo is intercepted or accessed on the device, it remains unreadable without the correct decryption key.
    - The encryption occurs locally on the device, meaning that the image is encrypted before it is stored or transmitted, ensuring end-to-end security.
- ***Encryption Flow:***
    - The app generates a unique encryption key for each image, ensuring that each photo is individually encrypted.
    - The encryption key and the image binary data are processed through the algorithm,

resulting in a fully encrypted image file.





- ***Storing the Encrypted Image**:*

    - Once the image is encrypted, it is stored locally in the app's secure storage area. This prevents the image from appearing in the device's default gallery or cloud-based services, keeping it isolated and protected.

    - The encrypted image file is also associated with metadata, such as a timestamp and its encryption key (which is also securely stored).

1.1 CLIENT-SERVER ARCHITECTURE

The **Client-Server Architecture** of **CryptoCamGuard** is designed to ensure secure, efficient, and reliable communication between the mobile application (client) and the backend server. This architecture follows a clear separation of concerns, with the mobile app handling user interactions, image encryption, and storage, while the server manages authentication, data requests, and additional secure data processing. Below is an in-depth breakdown of the architecture:

1.1.1 *Server-Side (Backend)*

The **server-side**, built using **.NET Core**, acts as the intermediary between the mobile application and the database. It handles crucial processes like user authentication, managing encrypted images, and processing API requests. The server ensures that communication with the client is secure and validated.

Key Server-Side Responsibilities:

1. **Authentication and Authorization**:
    - **JWT (JSON Web Tokens)**: The server generates and validates JWT tokens during user authentication, ensuring that only authorized users can access sensitive features.
    - **Login & Signup**: During the signup process, user credentials (passwords) are securely hashed using algorithms such as **bcrypt** before storage in the database.

- **Session Management**: The server validates each user's JWT token on every request to ensure session integrity and secure access.

2. **API Endpoints**: The server provides several **RESTful API endpoints** for the client to interact with:
    - **User Authentication APIs**: Endpoints for login and signup.
    - **Image Management APIs**: Endpoints to upload and retrieve metadata for encrypted images, handle image-related metadata, and securely transmit necessary information.
3. ***Secure Data Transfer**:*
    - The server uses **HTTPS (Hypertext Transfer Protocol Secure)** to encrypt data during transmission, ensuring that no sensitive data (such as login credentials or image metadata) can be intercepted by malicious actors.
4. ***Database Communication**:*
    - **User Data**: The server interacts with the database to securely store user credentials (hashed) and JWT tokens.
    - **Image Metadata**: Instead of storing actual images on the server, only metadata related to the images (such as timestamps, filenames, and encryption keys) is stored securely in the database.
5. ***Error Handling and Logging**:*
    - The server manages errors through centralized logging, providing insights into failed requests, authentication failures, or server-side errors. It can also return meaningful HTTP status codes to the client to inform users of potential issues.

1.1.2 *Database Layer*

The **database layer** stores essential user data and metadata related to encrypted images. It does not store the actual encrypted images (which remain on the client) but manages necessary information to ensure that image retrieval and security measures are properly enforced.

Key Database Responsibilities:

1. **User Data**:
    - Securely stores user information such as usernames, email addresses, and hashed passwords.
    - Authentication-related information like JWT tokens is also maintained for session validation.
2. ***Image Metadata****:*
    - Stores metadata about the images (e.g., file name, encryption details, timestamps) but not the images themselves.
    - The database ensures that metadata is associated with the correct user and is securely stored to prevent unauthorized access.

1.1.3 *Communication Flow Between Client and Server*

The communication between the client and server follows a secure and structured process to ensure efficient data handling and protection:

1. ***User Signup/Login****:*
    - **Client**: User enters their credentials, which are sent to the server via an HTTPS request.
    - **Server**: Receives the request, validates the credentials, hashes the password, and stores it in the database. A JWT token is generated for successful authentication.
    - **Client**: The JWT token is received and stored locally to validate future requests.
2. ***Capturing and Encrypting an Image****:*
    - **Client**: The user captures an image, which is encrypted.
    - **Client**: Sends a request to the server to store metadata about the encrypted image without sending the actual image.

- **Server**: Receives and stores the metadata in the database, ensuring it is linked to the

correct user.

3. ***Viewing Encrypted Images**:*

- **Client**: The user requests to view stored images. The app retrieves encrypted image metadata from the local storage and presents it to the user for decryption and viewing.
- **Server**: Verifies the JWT token to ensure that the request is authorized and returns the relevant metadata (if required).

4. ***Secure Communication**:*

- All requests and responses between the client and server are encrypted using **HTTPS**, ensuring that sensitive data (such as credentials and tokens) cannot be intercepted or tampered with.

5. Security Mechanisms in Client-Server Architecture

Security is a critical aspect of the **CryptoCamGuard** architecture, and it is addressed at multiple levels:

1. ***Encryption**:*

- **Client-Side Encryption**: All images are encrypted on the client-side using **AES-256** before any data is sent to the server, ensuring that sensitive data never leaves the client unprotected.
- **Transport Layer Security (TLS)**: Communication between the client and server is encrypted using HTTPS to prevent eavesdropping or man-in-the-middle attacks.

2. ***Authentication and Authorization**:*

- **JWT Tokens**: The server uses JWT tokens for session validation. Tokens are issued at login and are required for all future API requests to ensure that only authorized users can access sensitive data.
- **Session Expiry**: JWT tokens have a set expiry time, reducing the risk of unauthorized access in case tokens are compromised.

3. ***Data Integrity**:*

- All data transferred between the client and server is checked for integrity to ensure that it has not been altered during transmission.

4. ***Access Control**:*

    - The server enforces strict access control by validating JWT tokens on each request, ensuring that unauthorized users cannot access or modify other users' data.