Hamdard University
Department of Computing
Final Year Project

CryptoCamGuard Elevating Image Security

Software Design 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-SDS
Document Status	< Draft/ Final / etc..>
Author(s)	Sardar Nazeer, Ali Sher Siyal
Approver(s)	Saifullah Adnan
Issue Date	<Date of issuance of this document>

Name	Role	Signature	Date
Mohib Khan	Team Lead		
Ali Sher Siyal	Team Member 2		
Saifullah Adnan	Supervisor		
	Co-Supervisor		
	Project Coordinator		

Revision History

Date	Version	Description	Author
<dd/mmm/yyyy>	<x.x>	<details of the changes made>	<name>

Definition of Terms, Acronyms, and Abbreviations

Term	Description
SDS	Software Design Specifications
UI	User Interface
DB	Database
SRS	Software Requirement Specifications
CCG	CryptoCamGuard – Elevating Image Security App
API	Application Programming Interface
SHA (256)	Secure Hash Algorithm (256)
DOC	Document
HTTP, HTTPS	Hypertext Transfer Protocol, Hypertext Transfer Protocol Secure

1 Table of Contents

1 Introduction

1.1 Purpose of Document

The purpose of this document is to provide a detailed design specification for CryptoCamGuard, a security-focused application designed to enhance image security through advanced encryption and controlled sharing mechanisms.

1.2 Intended Audience

This document is intended for:

- Development team members
- Project managers
- System architects
- QA testers
- Stakeholders and supervisors

1.3 Project Overview

CryptoCamGuard is an innovative application focused on securing digital images using cryptographic techniques. Users can encrypt, store, and share their images securely, ensuring privacy and protection against unauthorized access.

1.4 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.

Not In Scope

The project will not cover:

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

2 Design Considerations

The design of the CryptoCamGuard application takes into account several critical factors to ensure the system's effectiveness, scalability, and security:

2.1 Assumptions and Dependencies

- The application will be accessible via web and mobile platforms.
- Users will have access to a stable internet connection.
- The backend will utilize a secure database like PostgreSQL.
- Hosting will be done on a cloud platform like AWS.

2.2 Risks and Volatile Areas

- Evolving security threats necessitating regular updates.
- Scalability issues with a growing user base.
- User adoption challenges for non-tech-savvy individuals.

2.3 Scalability and Performance

- The application should handle a high volume of image uploads and downloads.
- Load balancing will ensure performance consistency during peak usage.
- Caching mechanisms will reduce latency.

3 System Architecture

The CryptoCamGuard application is designed using a three-tier architecture to ensure scalability, robustness, and security. This architecture decomposes the system into subsystems or components, each with specific responsibilities that work together to provide the desired functionality.

Overview of Architecture

CryptoCamGuard follows a three-tier architecture:

1. **Presentation Layer:** Web and mobile interface.
2. **Application Layer:** Business logic and processing.
3. **Data Layer:** Secure storage and encryption.

3.1 System Level Architecture

The system architecture of CryptoCamGuard is designed to ensure a secure, scalable, and robust application. This section elaborates on the decomposition of the system into elements, their relationships, and integration with external systems.

3.1.1 System Decomposition into Elements

CryptoCamGuard is divided into the following primary components:

- **Presentation Layer:** This includes the user-facing components such as web and mobile applications built with React Native. It provides a user-friendly interface for uploading, encrypting, and sharing images.
- **Application Layer:** This layer is responsible for executing the business logic of the system. It consists of REST APIs developed using Django REST Framework, which handle encryption, user authentication, and image management.
- **Data Layer:** This layer stores all the data, including encrypted images, user information, and logs. PostgreSQL serves as the primary database for structured data, while AWS S3 stores encrypted image files.

3.1.2 The Relationship between the Elements

The elements of CryptoCamGuard interact through clearly defined interfaces to ensure smooth communication:

1. **User Interaction:** Users interact with the system through the web or mobile application (presentation layer), which forwards requests to the backend (application layer).
2. **API Communication:** The application layer processes these requests and communicates with the data layer using secure APIs to retrieve or update data.

3. **Data Storage:** Encrypted images and metadata are securely stored in the data layer, ensuring data integrity and availability.

3.1.2.1 Interfaces to External Systems

CryptoCamGuard integrates with external systems to enhance functionality and security:

- **Email Service:** Used for account verification, notifications, and alerts.
- **Authentication Provider:** OAuth 2.0 is used to authenticate users securely and manage tokens.
- **Cloud Storage Service:** AWS S3 is used to store encrypted images, ensuring scalability and durability.

3.1.3 Major Physical Design Issues

Several key considerations are addressed to ensure the system's physical design aligns with its requirements:

1. **Server Load Balancing:** Multiple application servers are deployed with load balancers to handle high user traffic efficiently.
2. **Data Redundancy:** Database and storage redundancy are implemented to prevent data loss and ensure high availability.
3. **Secure Communication:** All data exchanges between the layers and external systems are encrypted using HTTPS and TLS protocols.

3.2 Software Architecture

The software architecture defines the internal design of CryptoCamGuard, emphasizing modularity, scalability, and security.

3.2.1 User Interface Layer

The user interface layer is responsible for presenting data to the user and capturing input:

- **Technologies:** Built using React Native to ensure a seamless experience across web and mobile platforms.
- **Responsibilities:**
    - Rendering user interfaces.
    - Handling user interactions like uploading images and setting permissions.
    - Sending API requests to the backend.

3.2.2 Middle Tier

The middle tier handles the application's core functionalities:

- **Components:** Django REST Framework APIs and business logic modules.
- **Responsibilities:**
  - Validating user inputs and ensuring they meet security standards.
  - Implementing encryption algorithms (e.g., AES-256) for image security.
  - Managing session states and user authentication.

3.2.3 Data Access Layer

The data access layer is responsible for securely storing and retrieving data:

- **Technologies:** PostgreSQL for structured data and AWS S3 for encrypted image files.
- **Responsibilities:**
  - Performing CRUD operations for user data, permissions, and audit logs.
  - Ensuring data consistency and integrity.
  - Enforcing access controls to prevent unauthorized data access.

4 Design Strategy

The design strategy focuses on creating a robust, scalable, secure, and user-friendly platform through the following key principles:

4.1 Modular Architecture

Each module (encryption, authentication, sharing) is independently designed for scalability and maintainability.

4.2 Scalability and Performance

- Horizontal scaling with containerized services.
- Use of CDN for faster image delivery.

4.3 Security

- End-to-end encryption for all data.
- Regular penetration testing.

4.4 User Experience (UX)

- Intuitive design with accessible features for all user levels.
- Real-time feedback for user actions.

4.5 Maintainability

- Modular code with comprehensive documentation.
- Automated CI/CD pipelines for seamless updates.

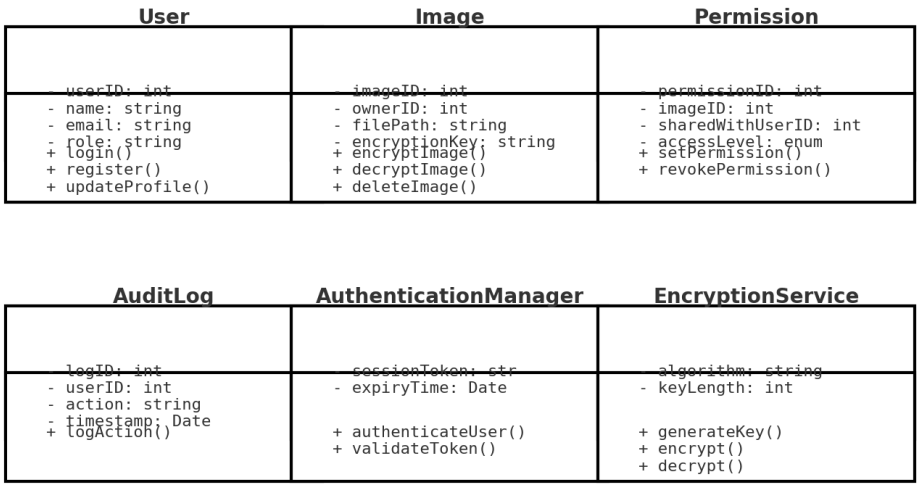
4.6 Reliability and Availability

- **Redundancy:** Prevent single points of failure with redundant servers and data replication.
- **Monitoring and Alerts:** Track performance and health, with alerts for issues.

5 Detailed System Design

[A detailed design should include the following:

5.1 Design Class Diagram

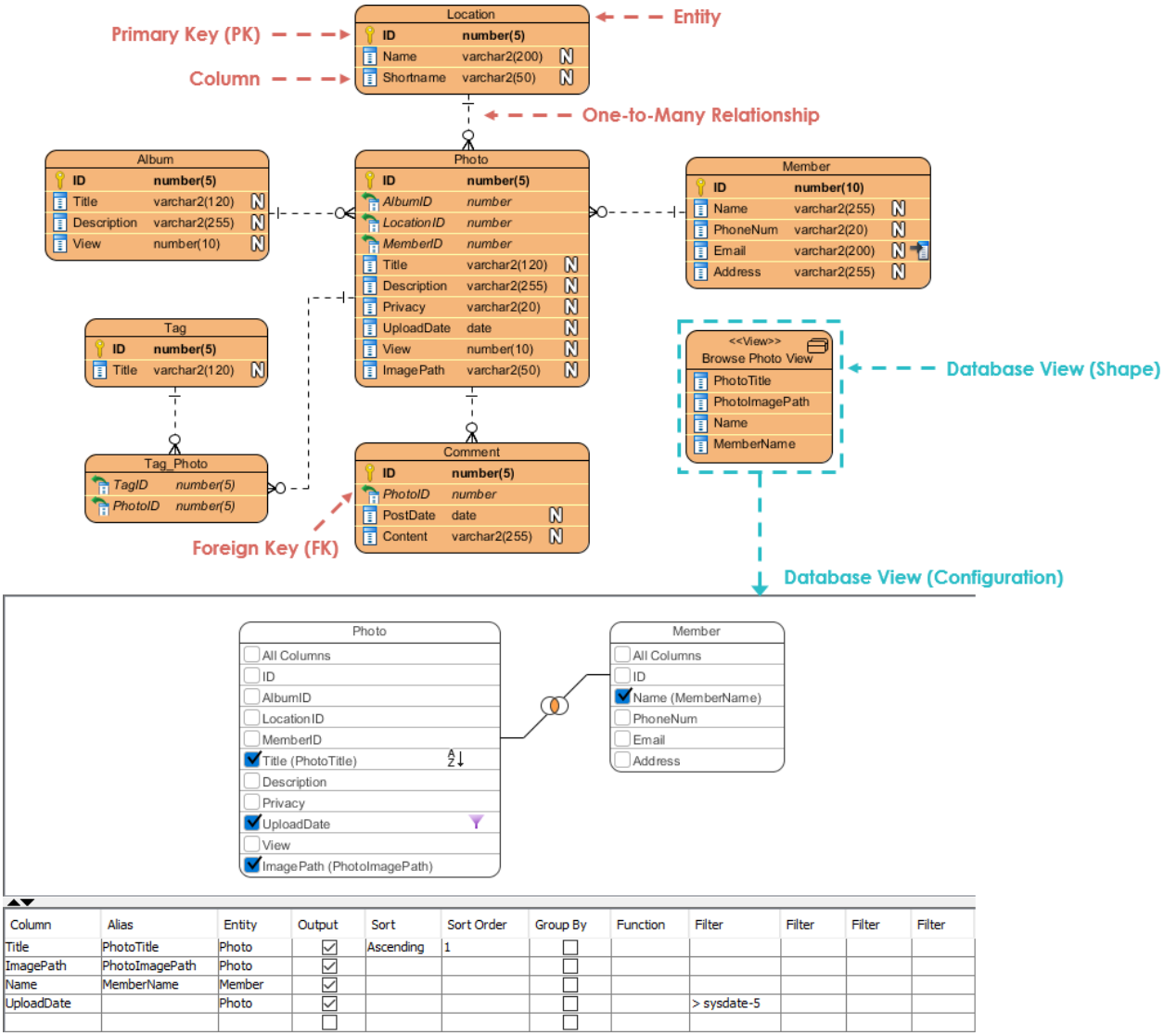


5.2 Database Design

[A detailed Database design should include the following:

- *Logical data model (E/R model)*
- *Data dictionary]*

5.3 ER Diagram

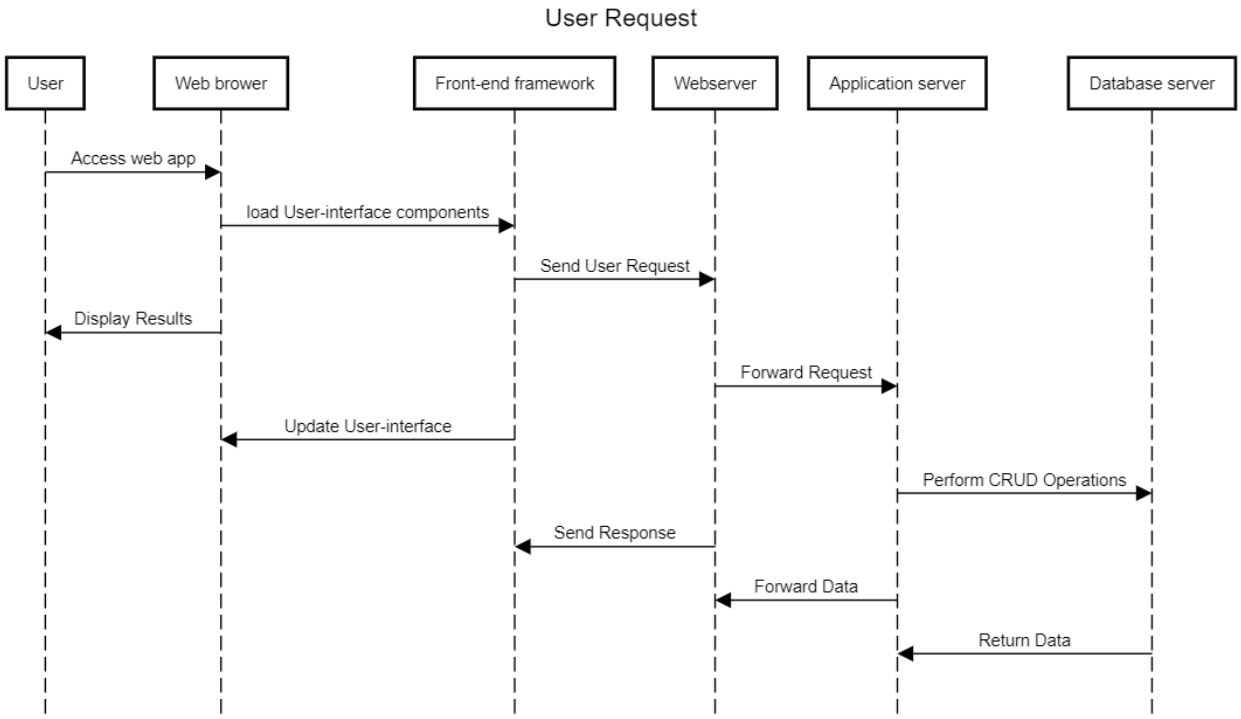


Column	Alias	Entity	Output	Sort	Sort Order	Group By	Function	Filter	Filter	Filter	Filter
Title	PhotoTitle	Photo	☑	Ascending	1	☐					
ImagePath	PhotoImagePath	Photo	☑			☐					
Name	MemberName	Member	☑			☐					
UploadDate		Photo	☑			☐		> sysdate-5			
			☐			☐					

5.4 Application Design

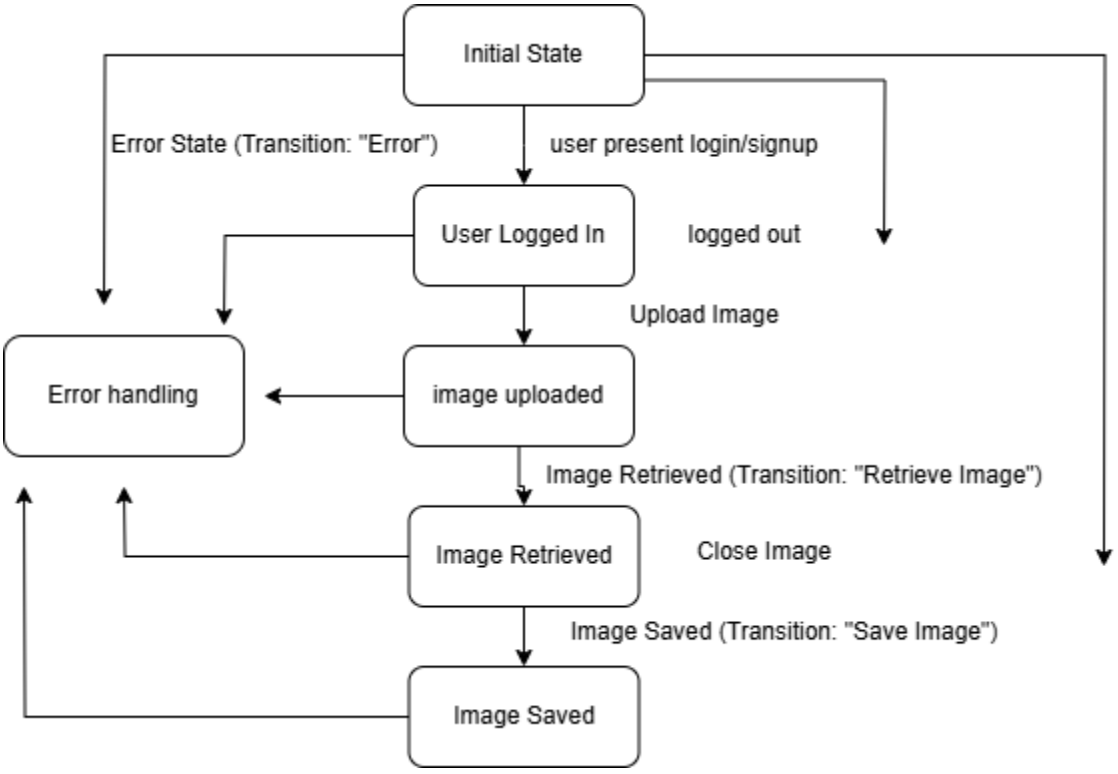
5.4.1 Sequence Diagram

5.4.1.1 <Sequence Diagram 1>

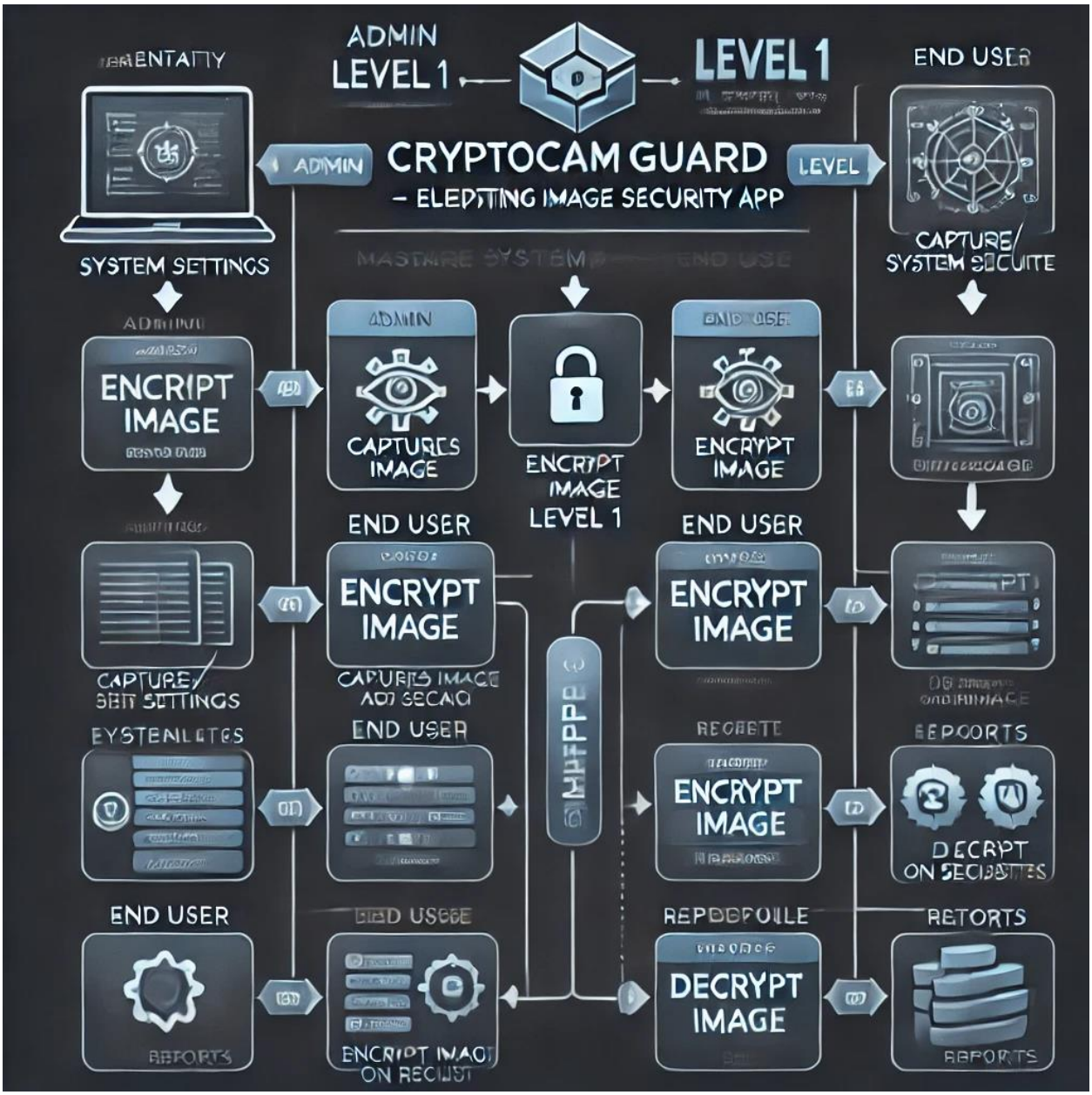


5.4.2 State Diagram

5.4.2.1 <State Diagram 1>

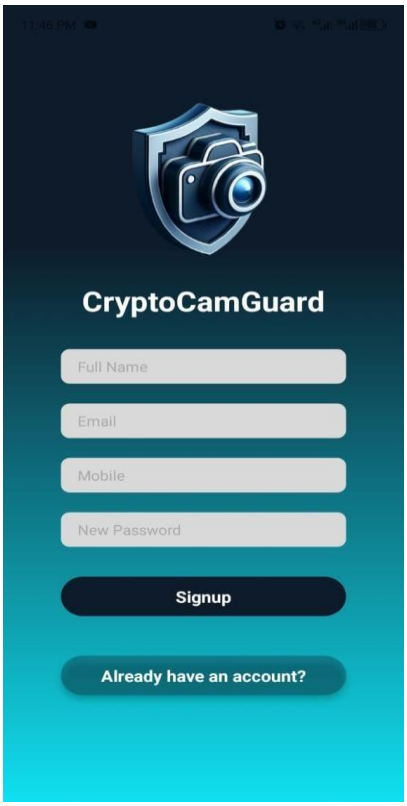


5.4.3 DFD level 1



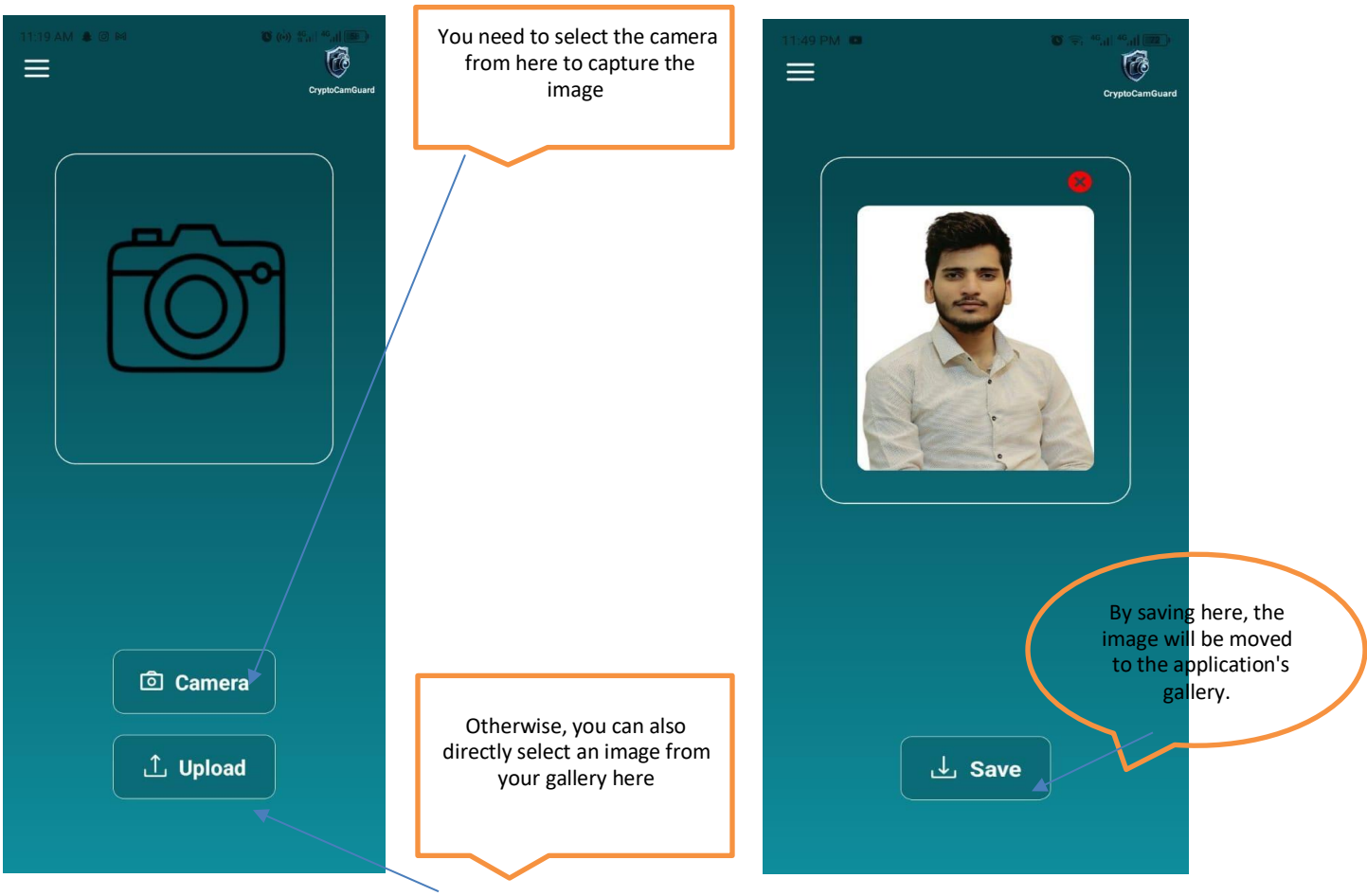
5.5 GUI Design

5.5.1 App Landing Page



To navigate the signup page

5.5.2 Project Core Features



- 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.

5.5.3 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.

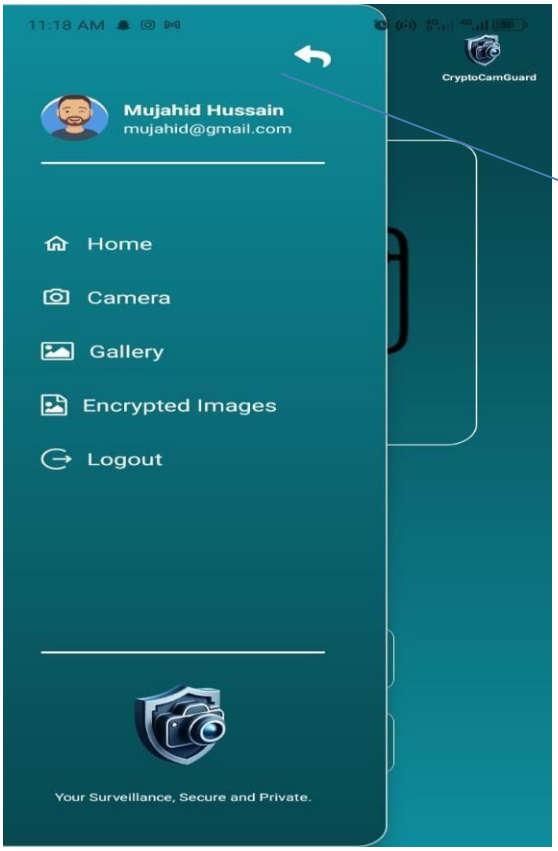
- **Encryption Flow**:

- 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.

5.5.4 Sidebar



With the help of this sidebar, the user can navigate to the hompage. If they want, they can also open the camera from here. Additionally, if the user wants to sign out, they can go to the sidebar and click the sign-out button.

- **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).

6 References

Tresorit

uses encryption technology to protect your files end-to-end, meaning no third-party or service provider can access them from upload to download.Overall, Tresorit is a safe and secure way to store and share your sensitive data. For more information, visit their website directly https://tresorit.com/

PixelKnot

is an Android app that allows you to hide messages within images and share them securely through any messaging platform. Its main purpose is to keep your images secure and aid in sharing them confidentially. PixelKnot utilizes steganography, meaning it hides your messages within the pixels of images, making them appear normal but containing hidden information. PixelKnot: Hidden Messages - Apps on Google Play

7 Appendices

[Include supporting detail that would be too distracting to include in the main body of the document.]