UNIVERSITY OF WOLLONGONG AUSTRALIA

SIM Global Education

School of Computer Science & Software Engineering

Bachelor of Computer Science (Digital Systems Security)

CSCI321 - Project
Technical Document

Group: FYP-25-S2-14P

Zikry Bin Affendi	8380338	zba505@uowmail.edu.au
Tan Zhi Qin	8259240	zqt072@uowmail.edu.au
Muhammad Hannan Azman	8220852	mha986@uowmail.edu.au
Wong Xin Yang	8497035	xyw523@uowmail.edu.au
Ernest Yeo Jun Long	8535139	ejly205@uowmail.edu.au

Supervisor: Tian Sion Hui

Assessor: Japit Sionggo

1. Introduction

1.1 Overview

The Safe QR Application allows its user to be in a safe and secured position when scanning and interacting with QR codes. It protects users from harmful threats, gives warnings of different categories and allows users to access links in a sandbox environment. This project aims to enhance public safety by developing a QR code scanner application that detects malware embedded within QR codes.

1.2 Scope

This document outlines the technical aspects of the SafeQR application, including system architecture, features, technologies used, and user interface design. It serves as a comprehensive guide for developers, testers, and stakeholders to understand the inner workings of the application and its components.

1.3 User Documentation

When users open the app, they will first see a screen asking them to either sign up or log in. Once logged in, they can navigate different features using tabs or buttons, including scanning QR codes, checking their scan history, managing user profile and adjusting settings. The app will then assess the QR code and deliver instant safety feedback and suggestions. After scanning, users can review the results and access their history from their profile. They can also personalize settings for QR scanning and safety alerts.

2. Functional Requirements

2.1 Core Feature

S/N	Feature	Description
01	Allow/Disallow Camera Access	A safety feature that asks user permission for access to the camera. User can select “While using the app”, “Only this time”, “Don’t Allow”
02	Scan QR Code via Device’s In-built Camera	Allow user to scan QR codes directly using their device’s camera
03	Turn On Flashlight when scanning QR Code	Allow user to turn on the flashlight to scan QR code when in a low lighting environment
04	Receive results on QR Code Scan results	Allow users to receive corresponding alerts after scanning the QR Code. The application categorizes the QR code into one of four outcomes—Safe, Unsafe. A short and brief description will also be included to educate users behind every result.
05	Importing QR Code from gallery	Allow users to take a photo of QR Code and save it in the gallery first. Then, import it to the application later to scan.
06	View History	Allows users to review their past QR code scan results, providing a record of previous interactions for reference or security purposes
07	Toggle App Theme	Users can adjust the app theme to their preference such as switching between dark and light modes for optimal viewing comfort.
08	Sandbox for Safe Browsing	Run potentially unsafe URLs in a secured, isolated environment to protect the user’s device from malicious content
09	Front/Back Camera Scanning	Allowing users to use both front and back cameras to scan the QR code. Providing a backup option if one camera malfunctions. This dual-camera functionality ensures uninterrupted scanning and enhances user convenience.

10	Clearing Scanned QR Codes History	Allowing users to clear the scanned QR codes history. This feature provides an additional layer of privacy and helps users manage their stored data more effectively.
11	Report Malicious Link	Allowing users to report malicious links embedded in QR code. This feature helps protect other users from potential threats and contributes to a safer scanning environment via the relevant channel.
12	Share QR Results	Allowing users to share the results with their friends via messenger application or email. So that friends and family can be cautious of these malicious links.

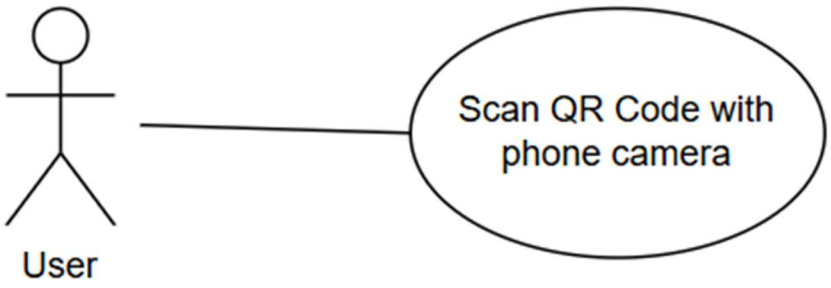
2.2 User Stories

.

- As a user, I want to scan the QR code with my phone camera to quickly access the encoded information.
- As a user, I want to upload QR code from my phone's photo library. So I can upload in the application and check on the link when I am free.
- As a user, I want to be able to view the past histories of QR codes scanned, so I would have a list of records for my future reference
- As a user, I want the application to provide me with precautions if the scanned QR code is flagged as having malicious content.
- As a user, I want to be able to open the link in a sandbox environment within the application, so I can safely view the content without being exposed to suspicious content.
- As a user, I want to be able to choose between light or dark mode within the application so that I can adjust the display for optimal viewing comfort.
- As a user, I want to be able to report malicious links to the relevant authorities so that appropriate action can be taken to block or remove the link, ensuring a safer scanning experience for everyone.
- As a user, I want to share the QR code results with my family and friends so that they can access the same information quickly and conveniently.
- As a user, I want to be able to turn on a flashlight when scanning QR code. So that even in a bad lighting environment, I will still be able to scan QR code and determine if the QR is safe or malicious
- As a user, I want to clear the history result of the QR code I have scanned.
- As a user, I want to be able to use the front camera to scan the QR code. In the event where my rear camera is not working, I still have an option to use the front camera.

2.3 Use Case Diagram and Description

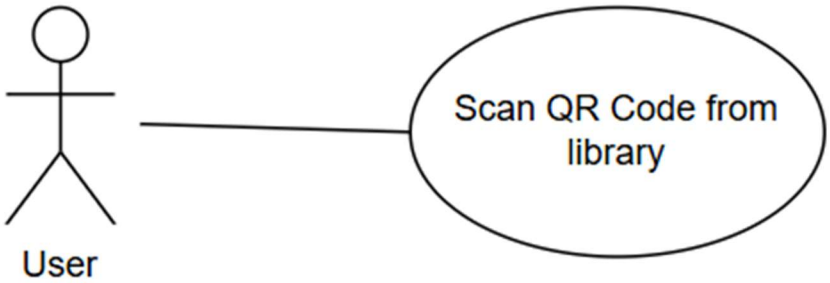
2.3.1 Use Case Diagram



Use Case Description

Use Case Name:	Scan QR code with phone camera	**Use Case ID:** #TC-004
Pre-Condition(s):	User must install and open the application	
Actor:	User	
Description:	As a user, I want to scan the QR code with my phone camera to quickly access the encoded information	
Normal Flow:	1) User aligns phone camera to the QR code	
Alternate Flow:	-	

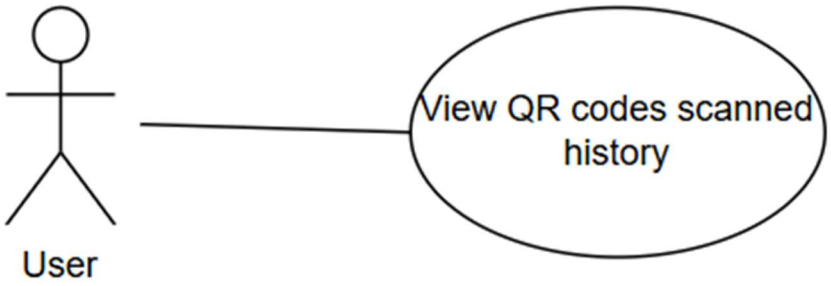
2.3.2 Use Case Diagram



Use Case Description

Use Case Name:	Upload QR code from photo library	**Use Case ID:** #TC-005
Pre-Condition(s):	User must have a valid QR code in library	
Actor:	User	
Description:	As a user, I want to upload QR code from my phone's photo library.	
Normal Flow:	1) On the main page after successful login, user clicks on "Upload QR" 2) User selects the QR code from photo library 3) System will scan QR code that the user chose	
Alternate Flow:	If the user chose an invalid image - for example a blurry QR code image or an image which is not a QR code	

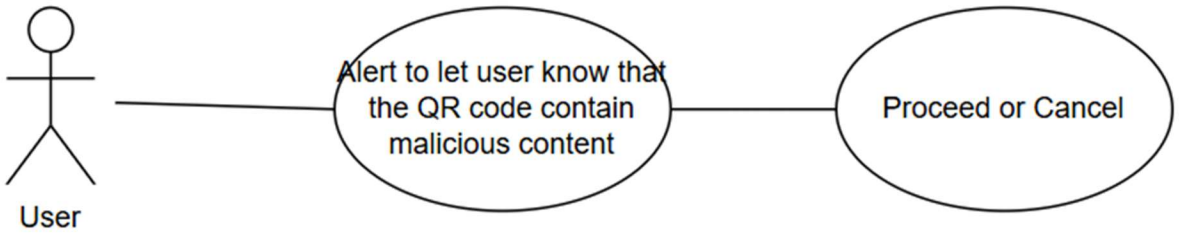
2.3.3 Use Case Diagram



Use Case Description

Use Case Name:	View QR codes scanned history	**Use Case ID:** #TC-006
Pre-Condition(s):	Have already scanned QR codes before	
Actor:	User	
Description:	As a user, I want to be able to view the past histories of QR codes scanned, so I would have a list of records for my future reference	
Normal Flow:	1) On the main page, user clicks "QR History" button 2) System will retrieve a list of QR codes that user have scanned in the past	
Alternate Flow:	-	

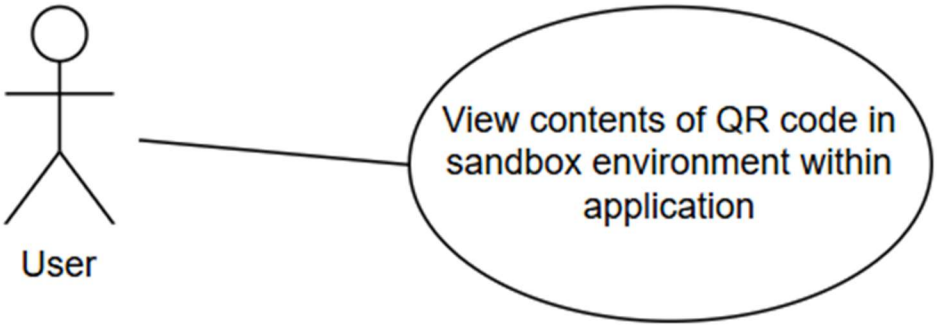
2.3.4 Use Case Diagram



Use Case Description

Use Case Name:	Actions to take if QR codes contains malicious contents	**Use Case ID:** #TC-007
Pre-Condition(s):	The contents of the QR code scanned has to be malicious	
Actor:	User	
Description:	As a user, I want the application to provide me with precautions if the scanned QR code is flagged as having malicious content.	
Normal Flow:	1) User scans QR code from phone camera or photo library 2) System will flag the content of the QR codes to be malicious	
Alternate Flow:	Invalid QR code	

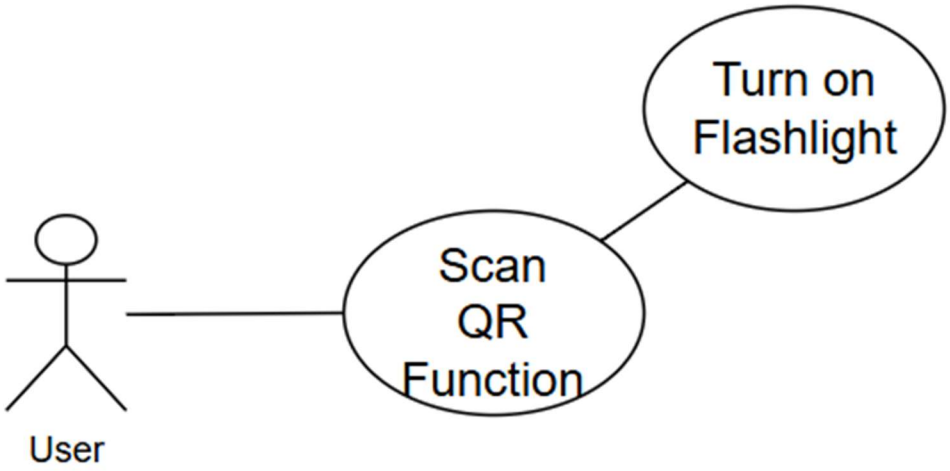
2.3.5 Use Case Diagram



Use Case Description

Use Case Name:	View contents of QR code in sandbox environment	**Use Case ID:**#TC-008
Pre-Condition(s):	Have already scanned a valid QR code	
Actor:	User	
Description:	As a user, I want to be able to open the link in a sandbox environment within the application, so I can safely view the content without being exposed to suspicious content.	
Normal Flow:	1) Application will open the link in a sandbox environment within application 2) Contents of the link will be displayed	
Alternate Flow:	Application will show an error message if the contents of the link cannot be loaded.	

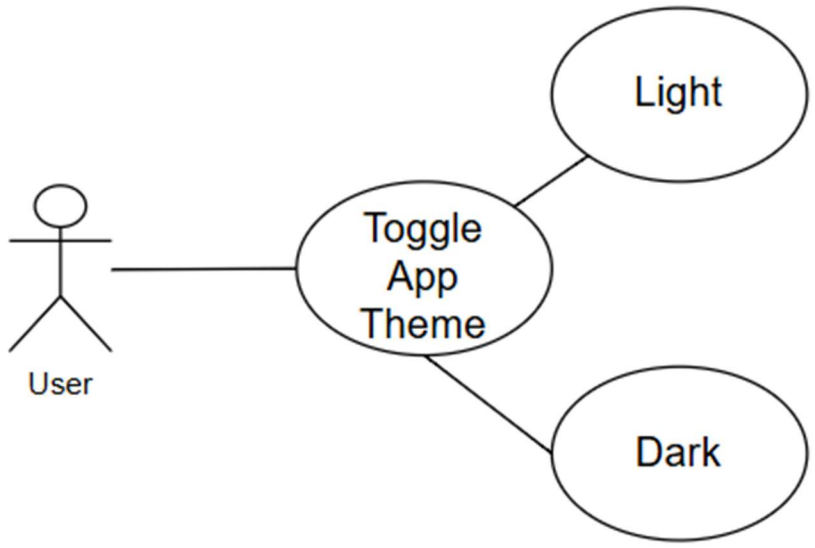
2.3.6 Use Case Diagram



Use Case Description

Use Case Name:	Turn on Flashlight when scanning QR Code	**Use Case ID:** #TC-009
Pre-Condition(s):	In a bad lighting environment	
Actor:	User	
Description:	As a user, I want to be able to scan QR code in a bad lighting environment	
Normal Flow:	1) Go to Scan QR function 2) Press the flashlight button to turn on flashlight	
Alternate Flow:	-	

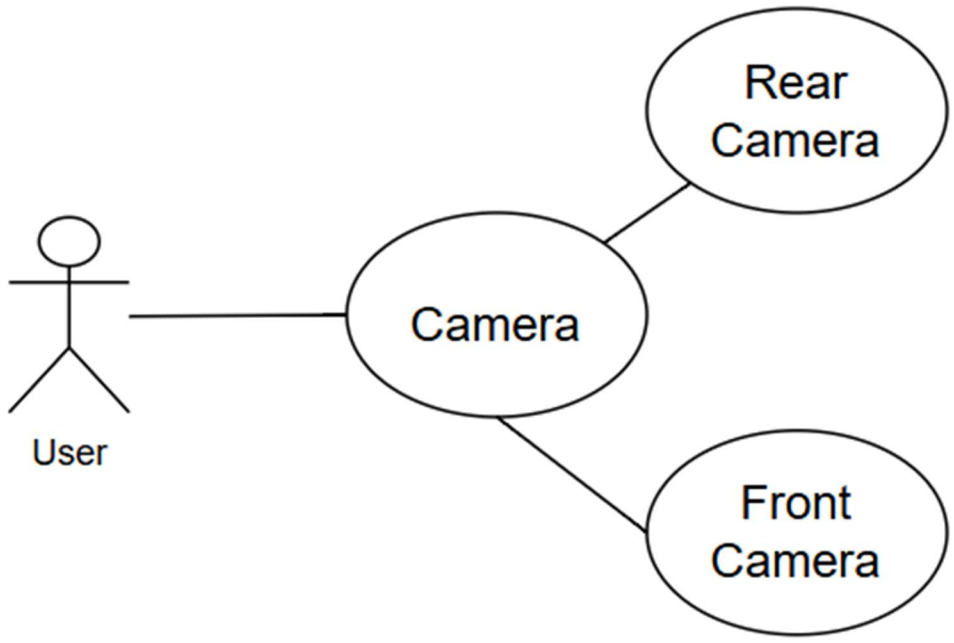
2.3.7 Use Case Diagram



Use Case Description

Use Case Name:	Toggle App Theme	**Use Case ID:** #TC-010
Pre-Condition(s):	-	
Actor:	User	
Description:	As a user, I want to be able to choose between light or dark mode within the application so that I can adjust the display for optimal viewing comfort.	
Normal Flow:	1) Go to Settings 2) User can make the option to toggle “Dark mode” on app	
Alternate Flow:	-	

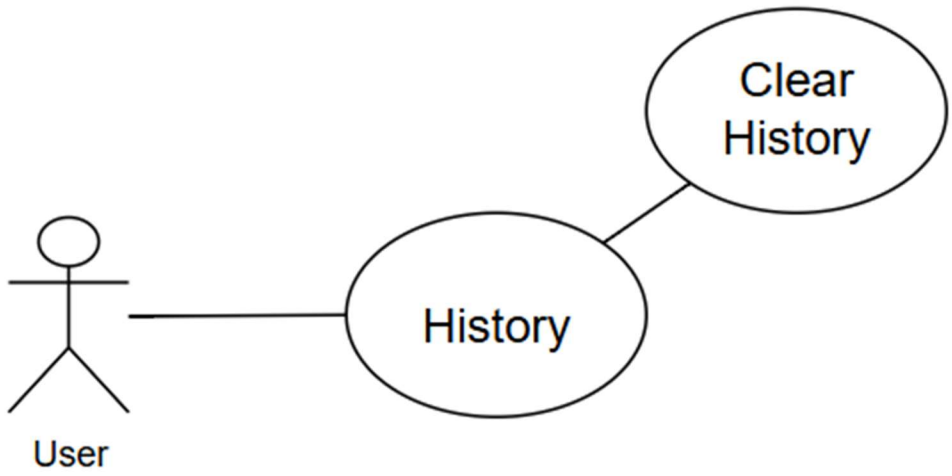
2.3.8 Use Case Diagram



Use Case Description

Use Case Name:	Flip to front camera	**Use Case ID:** #TC-011
Pre-Condition(s):	User must be at the scan QR scanner page	
Actor:	User	
Description:	As a user, I want to have the option to use either a front or rear camera to scan QR code. In the event that one of the cameras breaks down, I still can use the other one.	
Normal Flow:	1) Go to Scan QR function 2) Tap the on camera icon to switch camera	
Alternate Flow:	-	

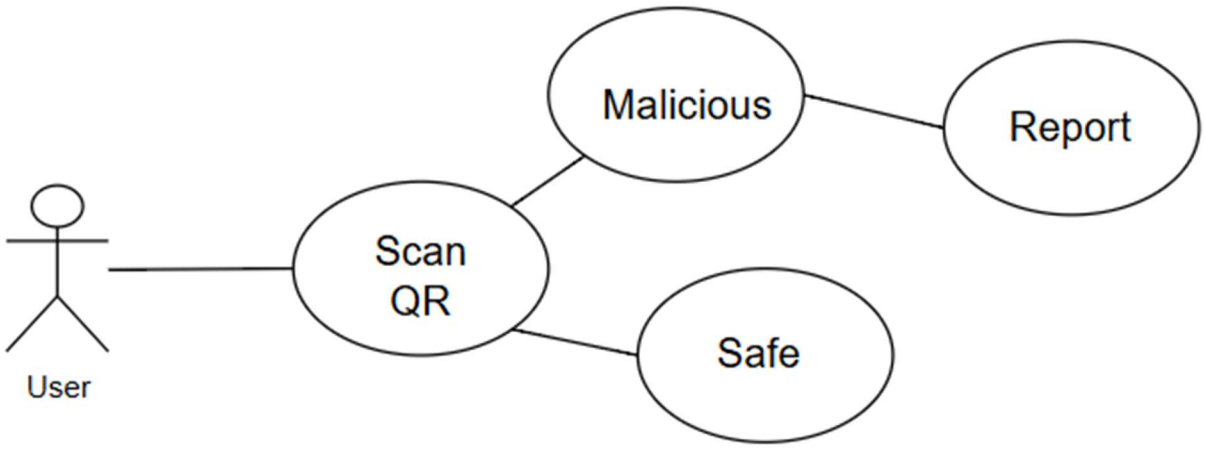
2.3.9 Use Case Diagram



Use Case Description

Use Case Name:	Clear History	**Use Case ID:** #TC-012
Pre-Condition(s):	User must be at the History page	
Actor:	User	
Description:	As a user, I want to be able to clear the search history of QR codes that were scanned before.	
Normal Flow:	1) Go to “History” page 2) Tap on the “Click history” icon	
Alternate Flow:	-	

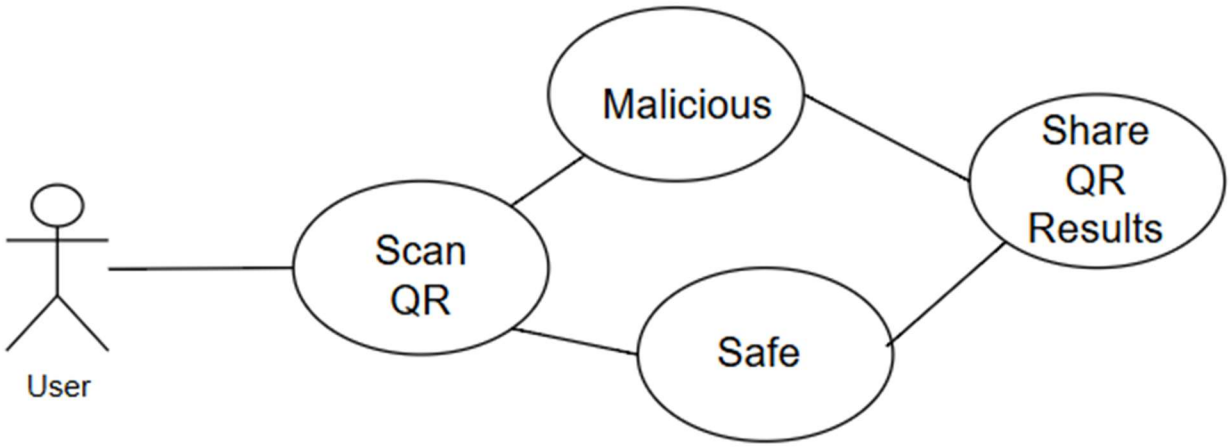
2.3.10 Use Case Diagram



Use Case Description

Use Case Name:	Report Malicious Link	**Use Case ID:** #TC-013
Pre-Condition(s):	Reporting button is only available when the scanned result is marked as malicious	
Actor:	User	
Description:	As a user, after knowing that the QR code I scanned is malicious, I want to report it to the relevant authorities so that further actions can be taken.	
Normal Flow:	1) Scan QR code 2) Result came out and it is malicious 3) Tap on the "Report This Scam"	
Alternate Flow:	-	

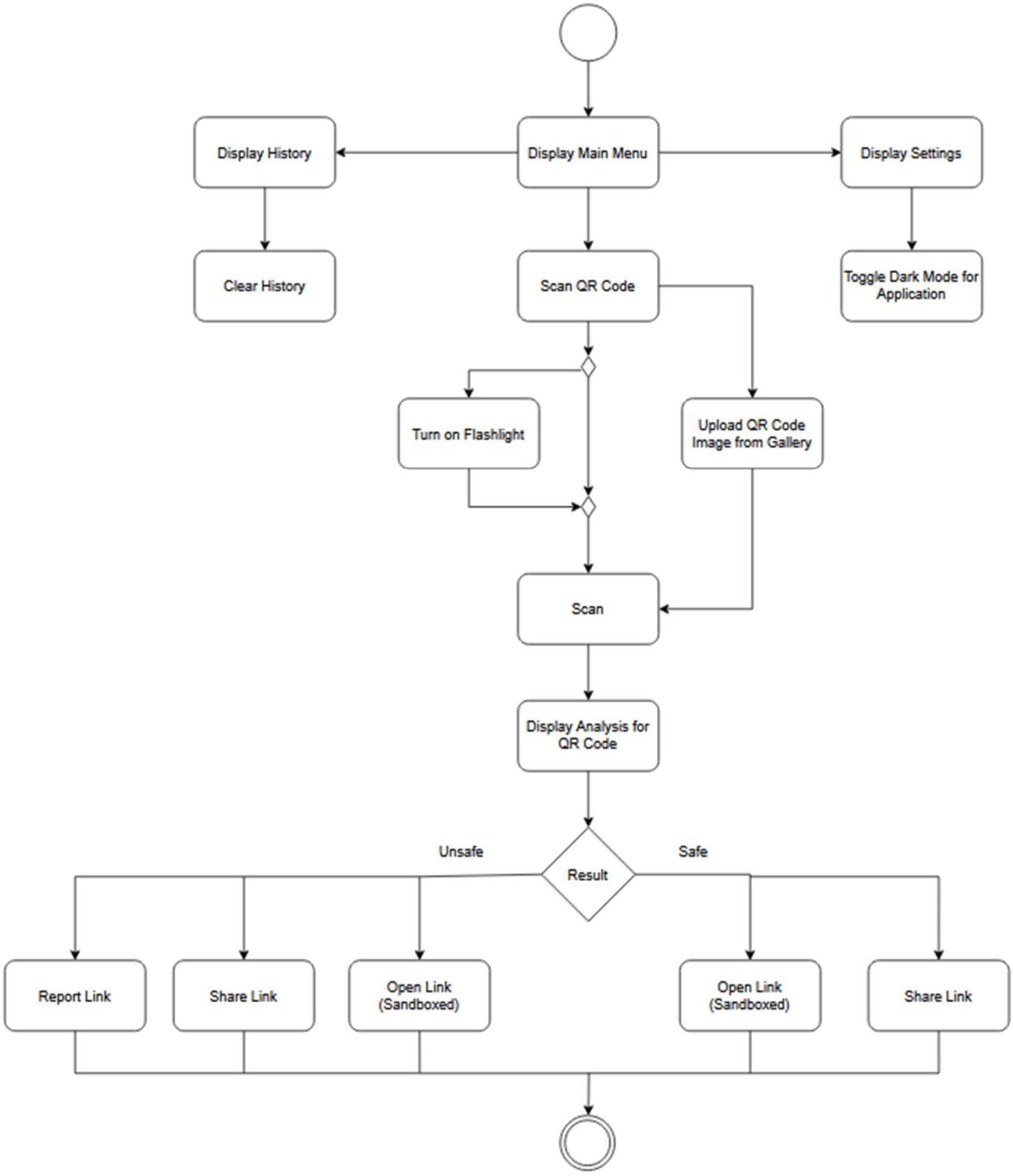
2.3.11 Use Case Diagram



Use Case Description

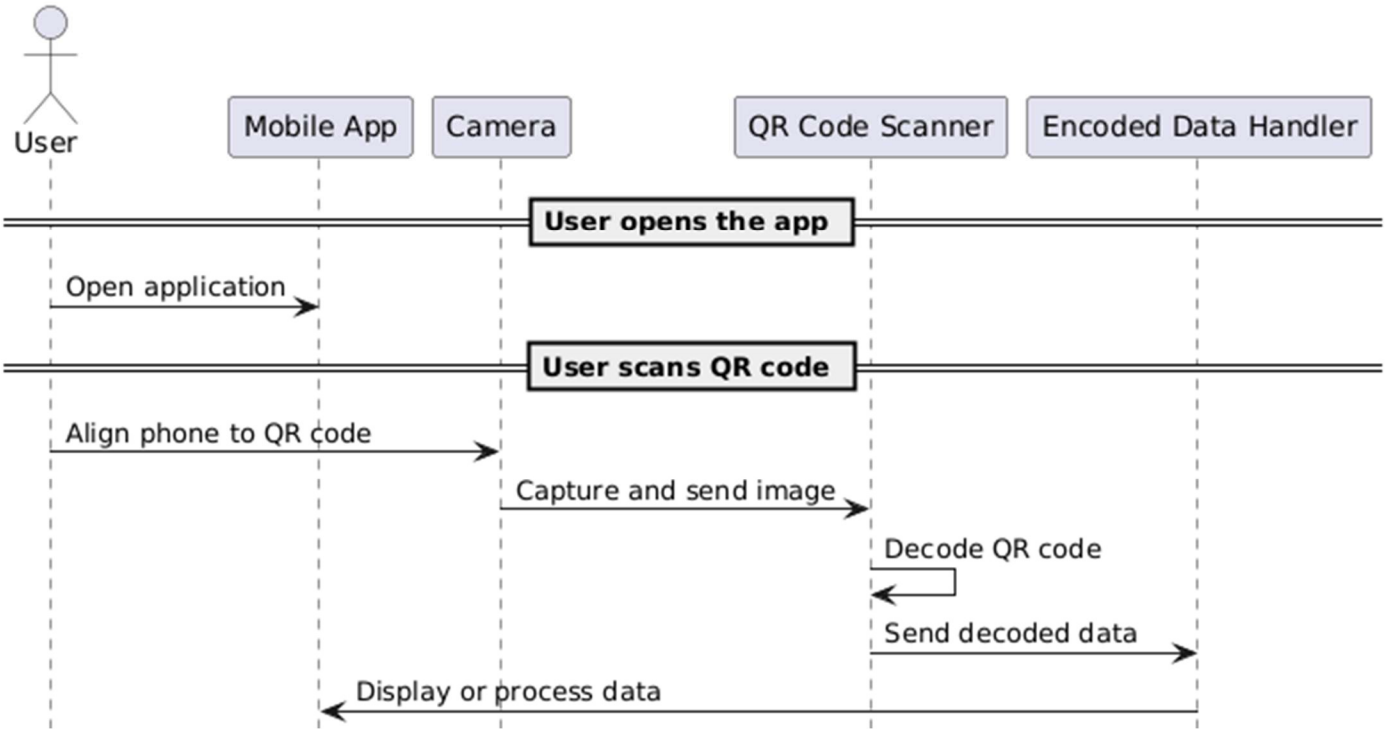
Use Case Name:	Share QR Results	**Use Case ID:** #11
Pre-Condition(s):	Have a scanned result.	
Actor:	User	
Description:	As a user, I want to share the scanned QR results to my friends and family. So that they are able to know what is safe and what to be cautious.	
Normal Flow:	1) Scan QR 2) Receive Result 3) Click on “Share results” to share the scanned QR results via messenger/email	
Alternate Flow:	-	

2.5 User Activity Diagram

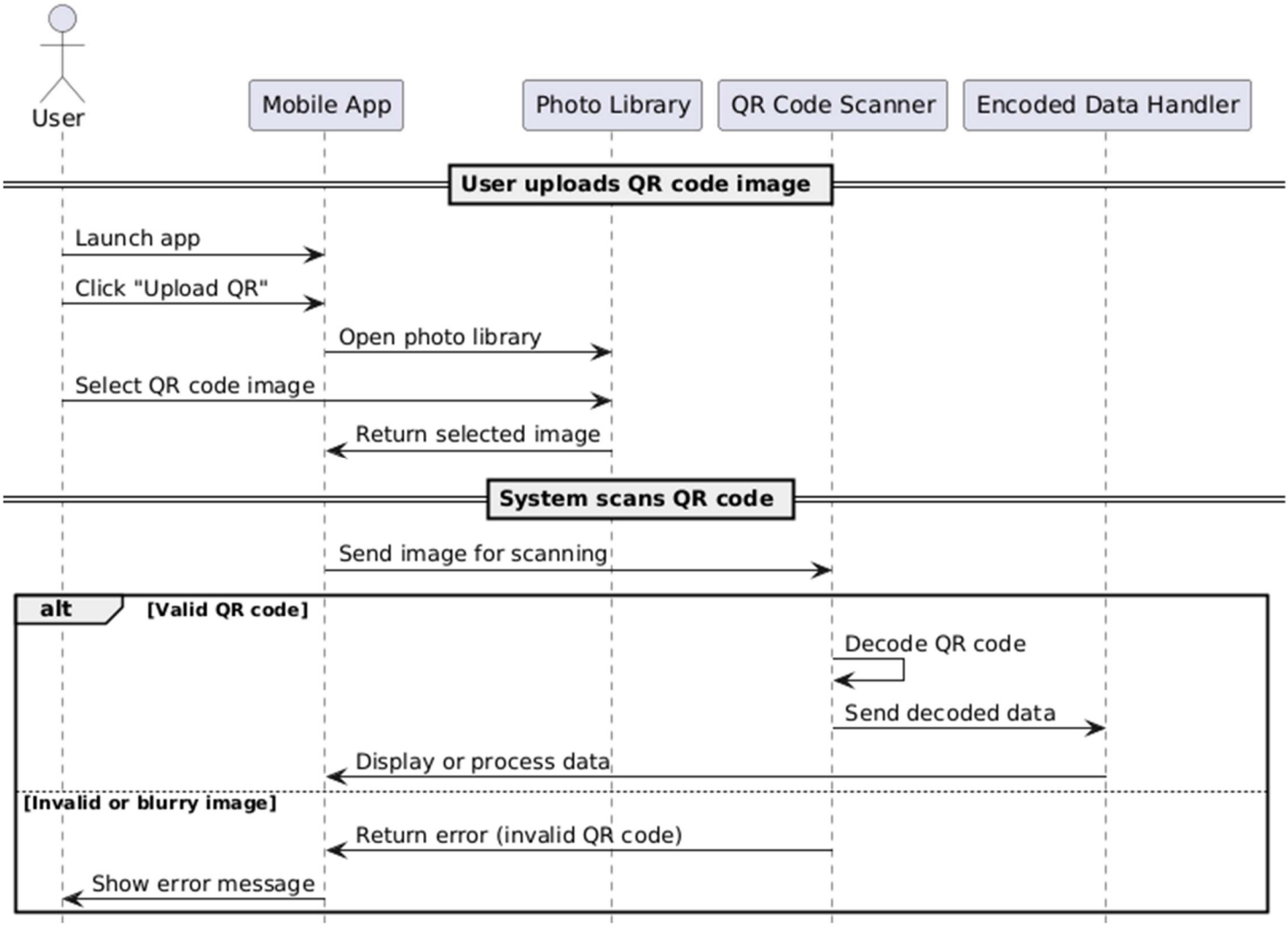


2.6 Sequence Diagram

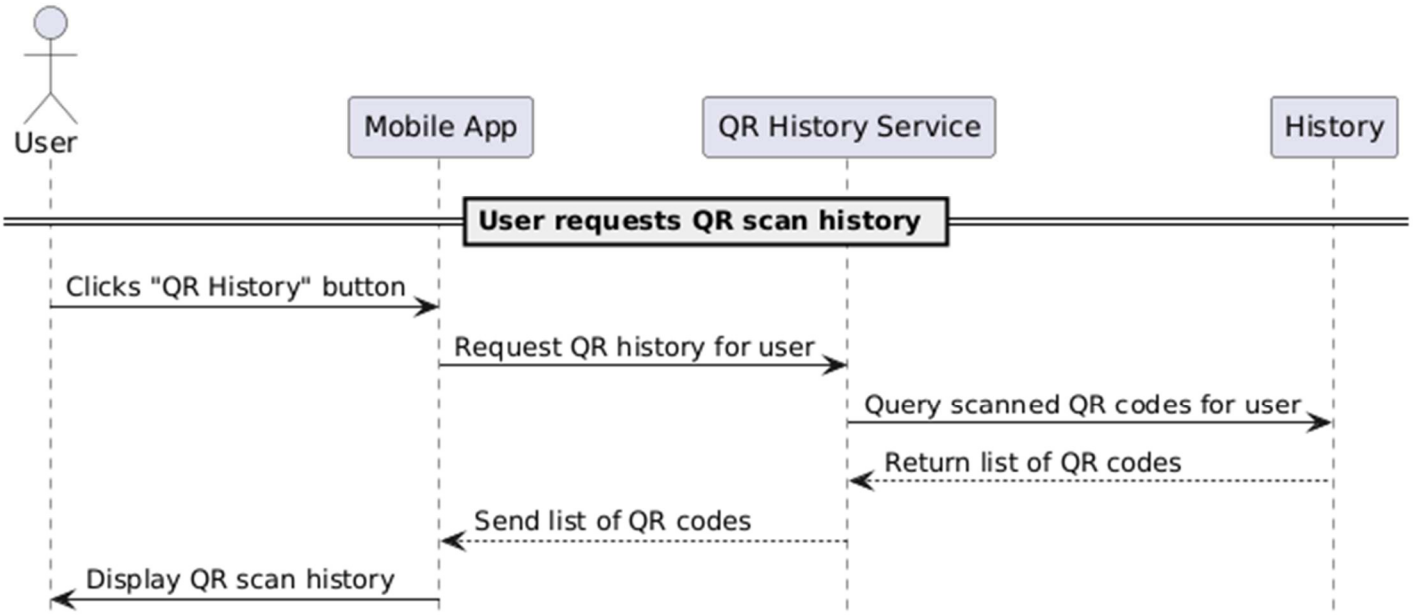
Scan QR Code



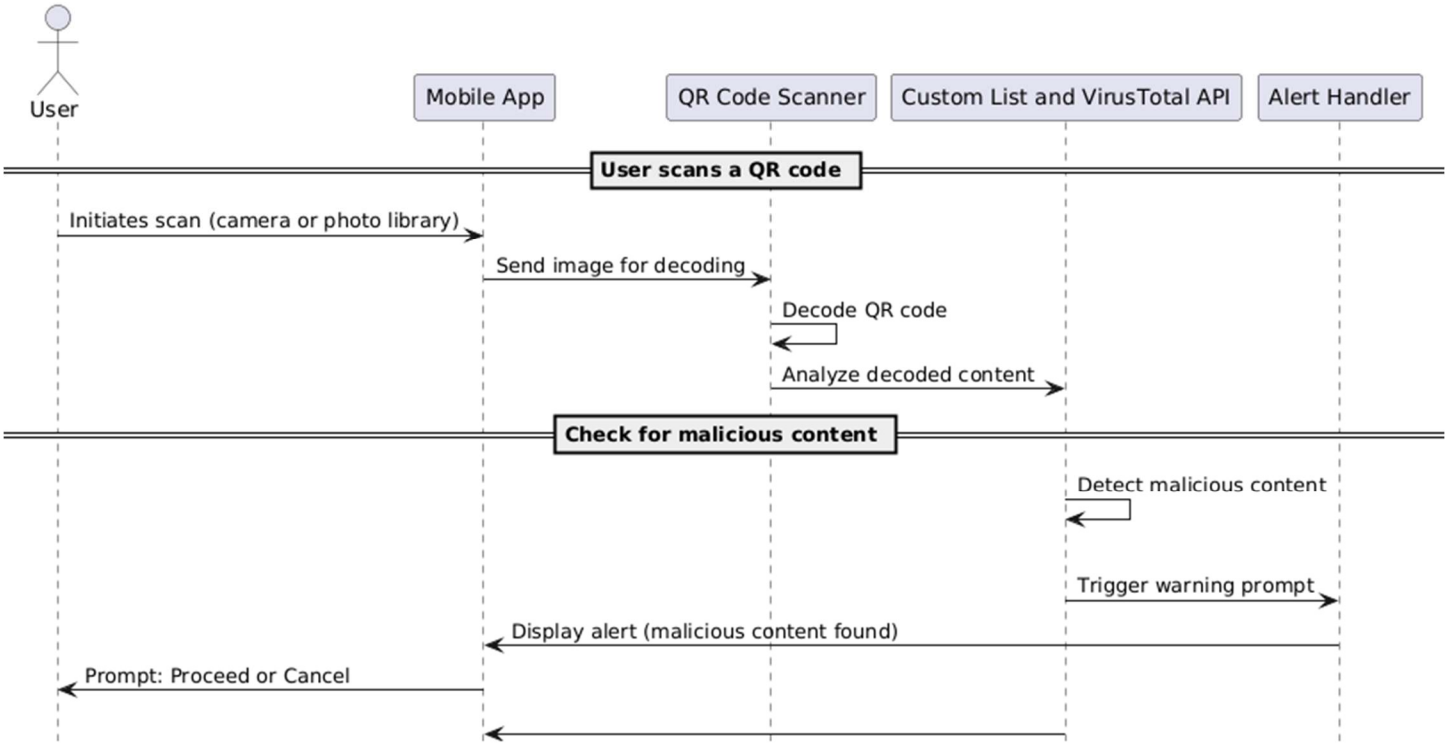
Scan QR Code from photo library



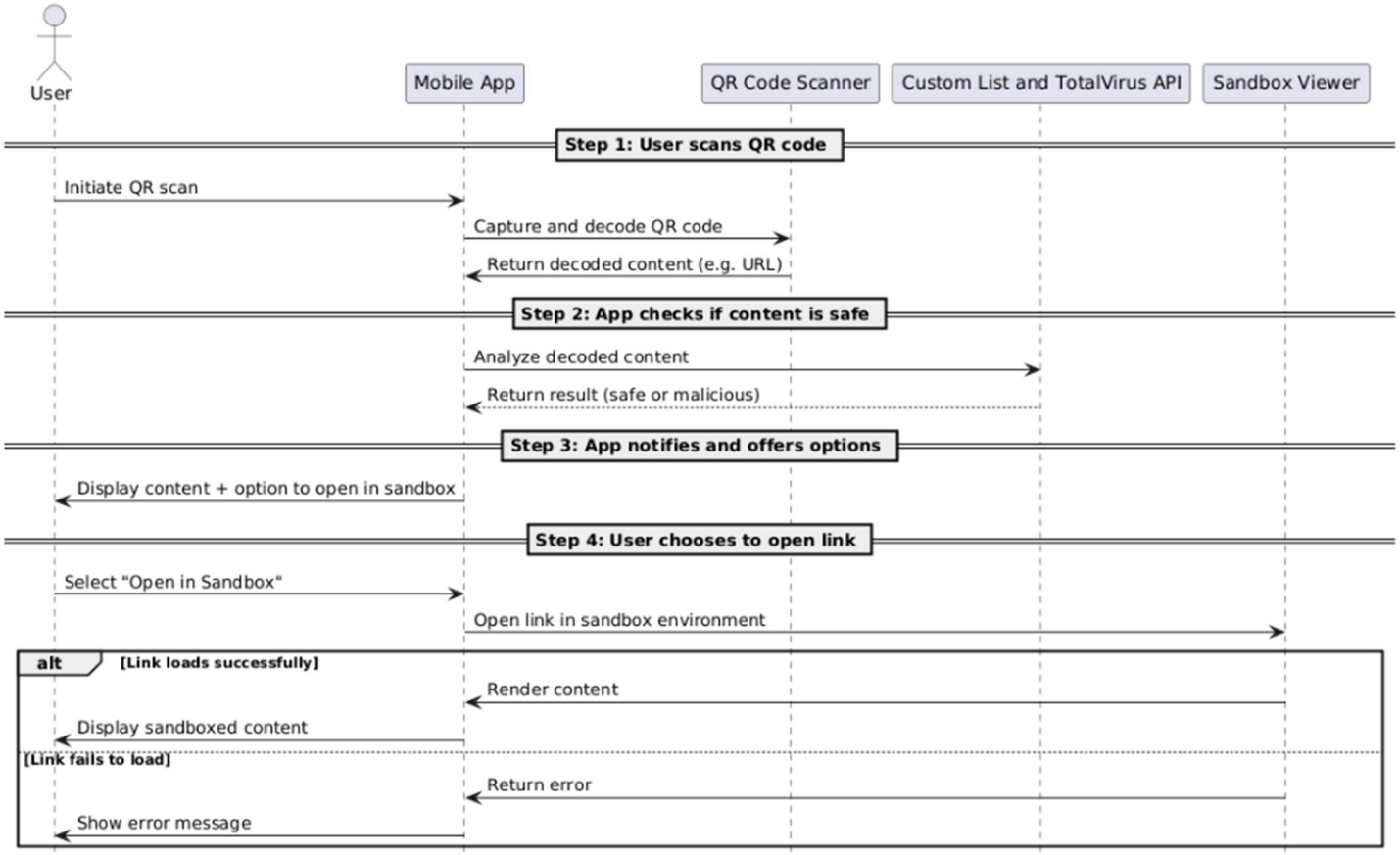
View History



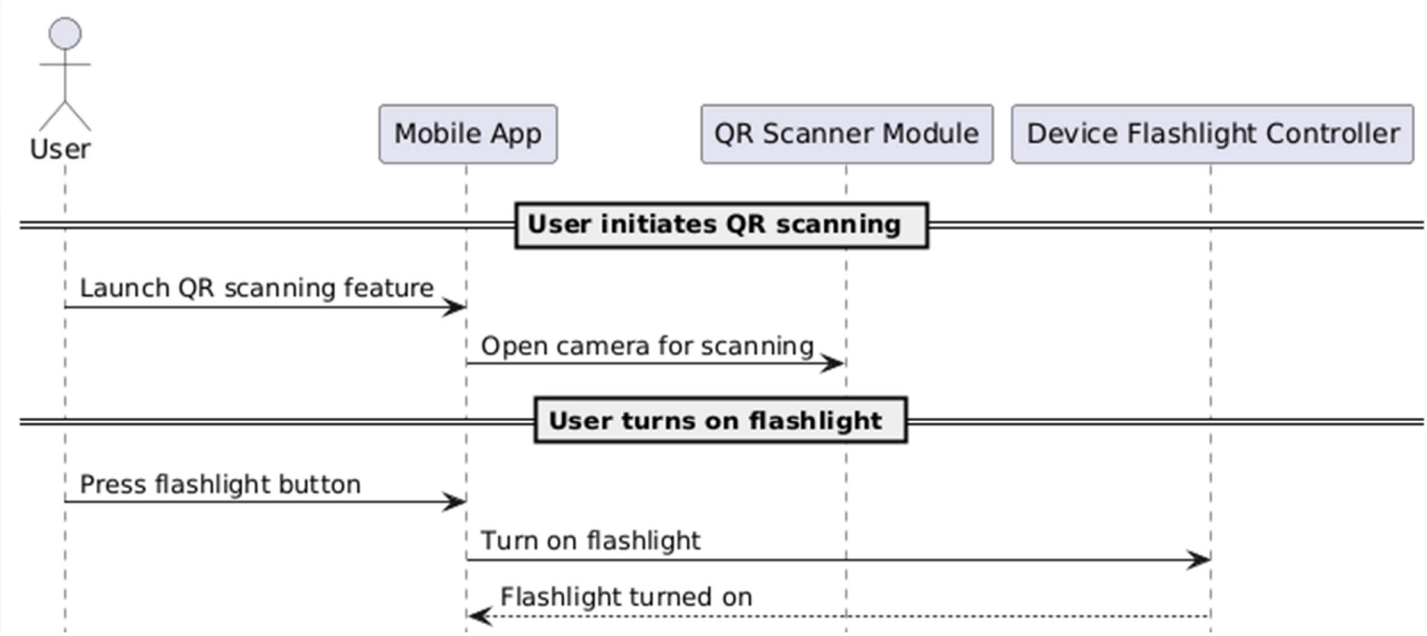
Action to take if QR Code is malicious



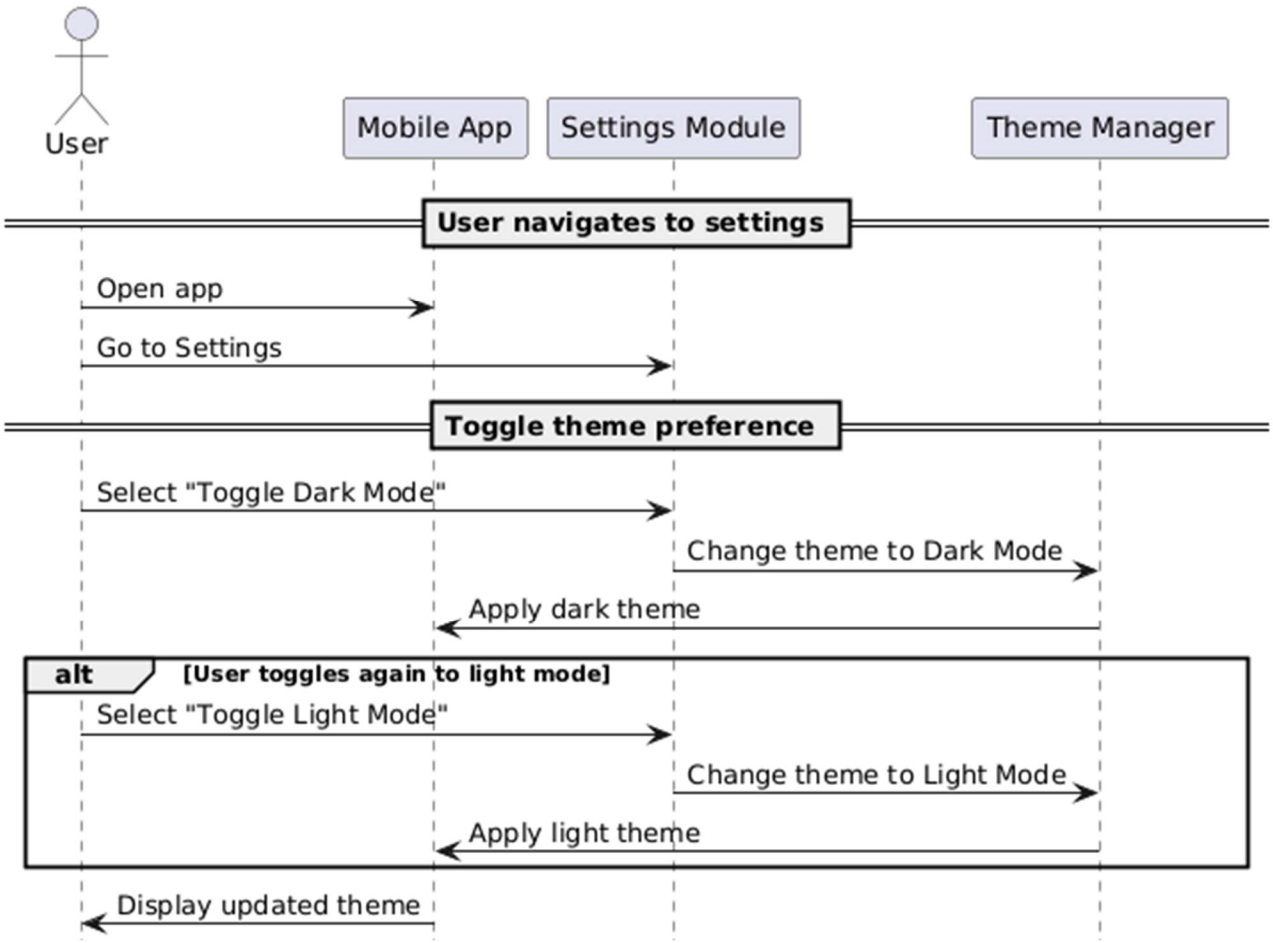
View contents of scanned QR within Sandbox Environment



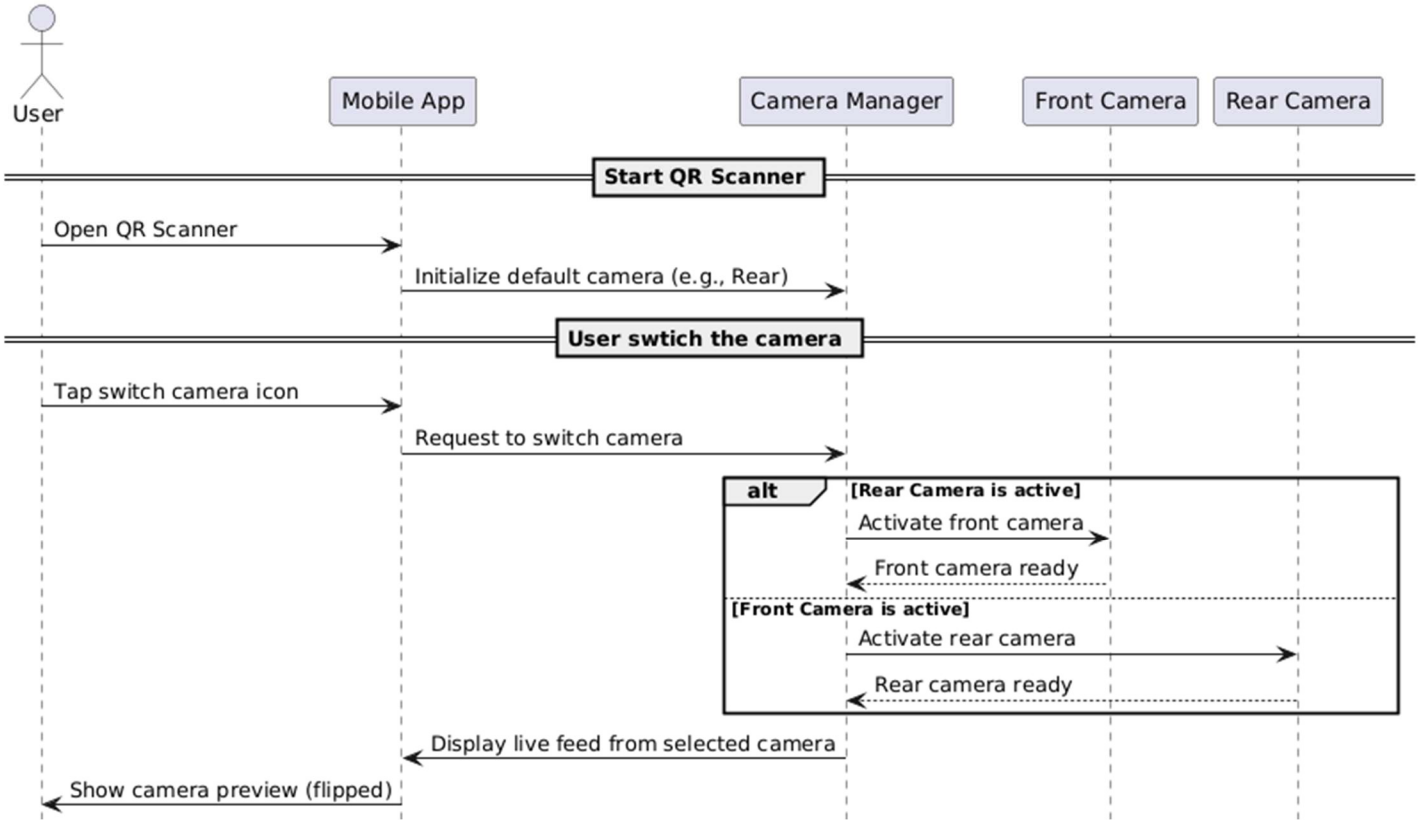
Turning on Flashlight



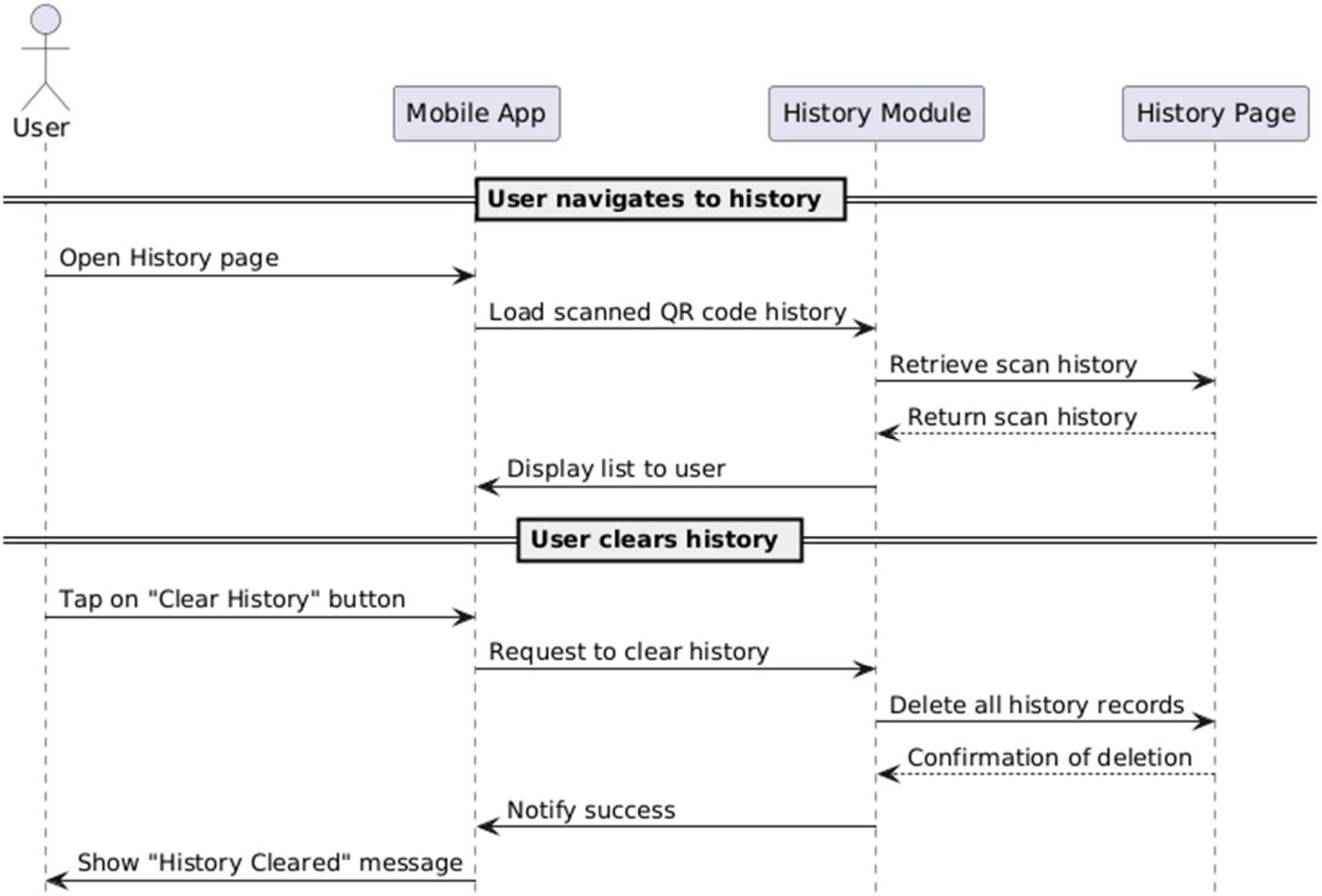
Turning on Flashlight



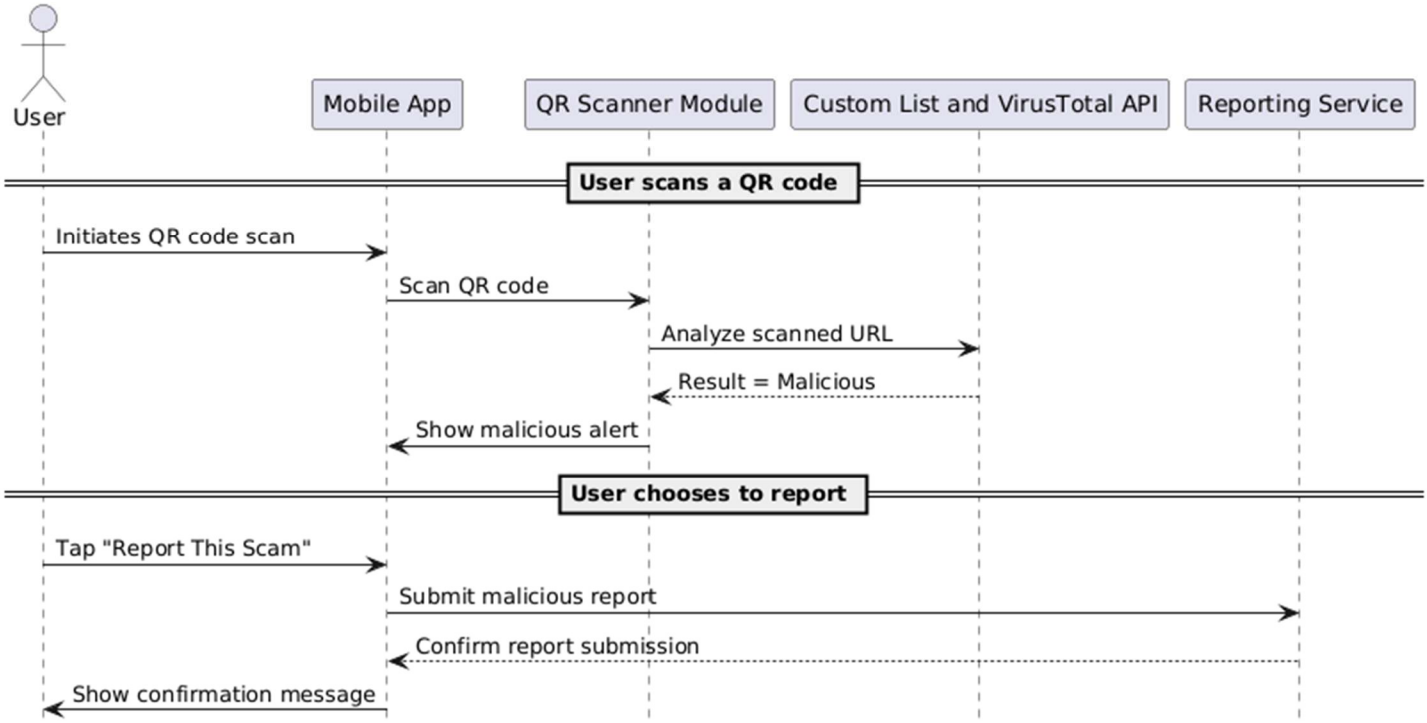
Switching between front and rear cameras



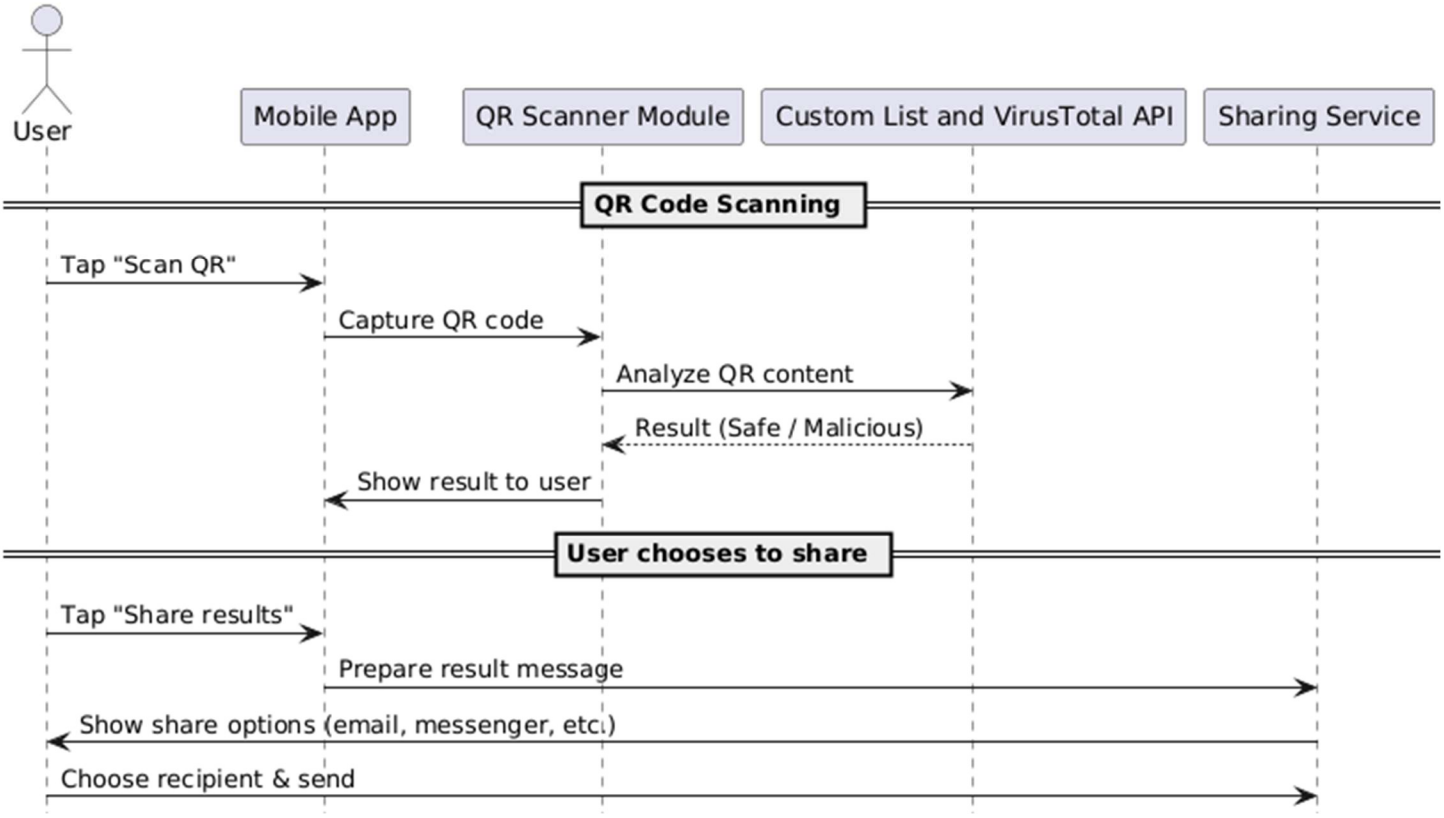
Clearing Scanned History



Reporting Malicious Link/QR



Share Scanned QR Results

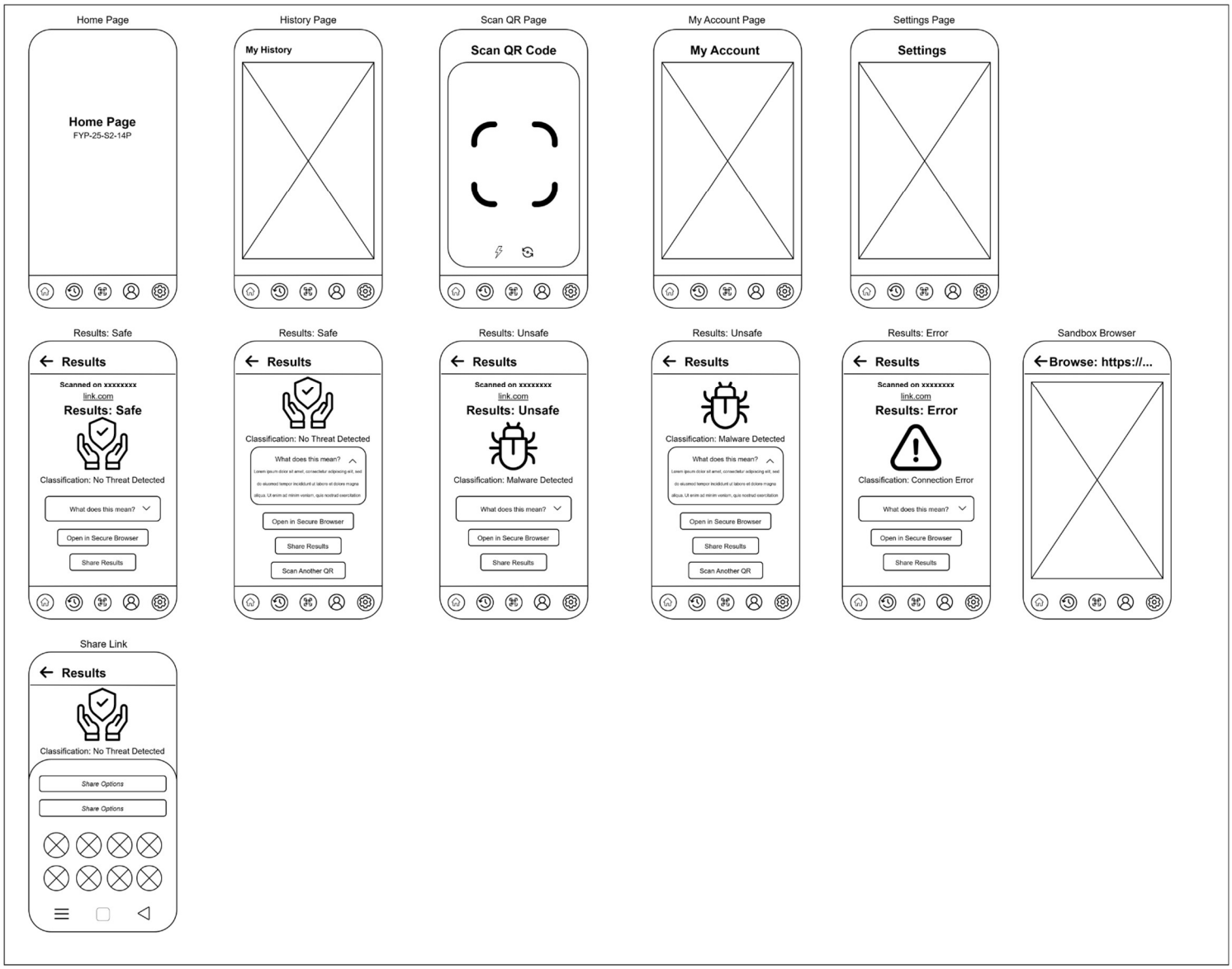


3. Wireframes

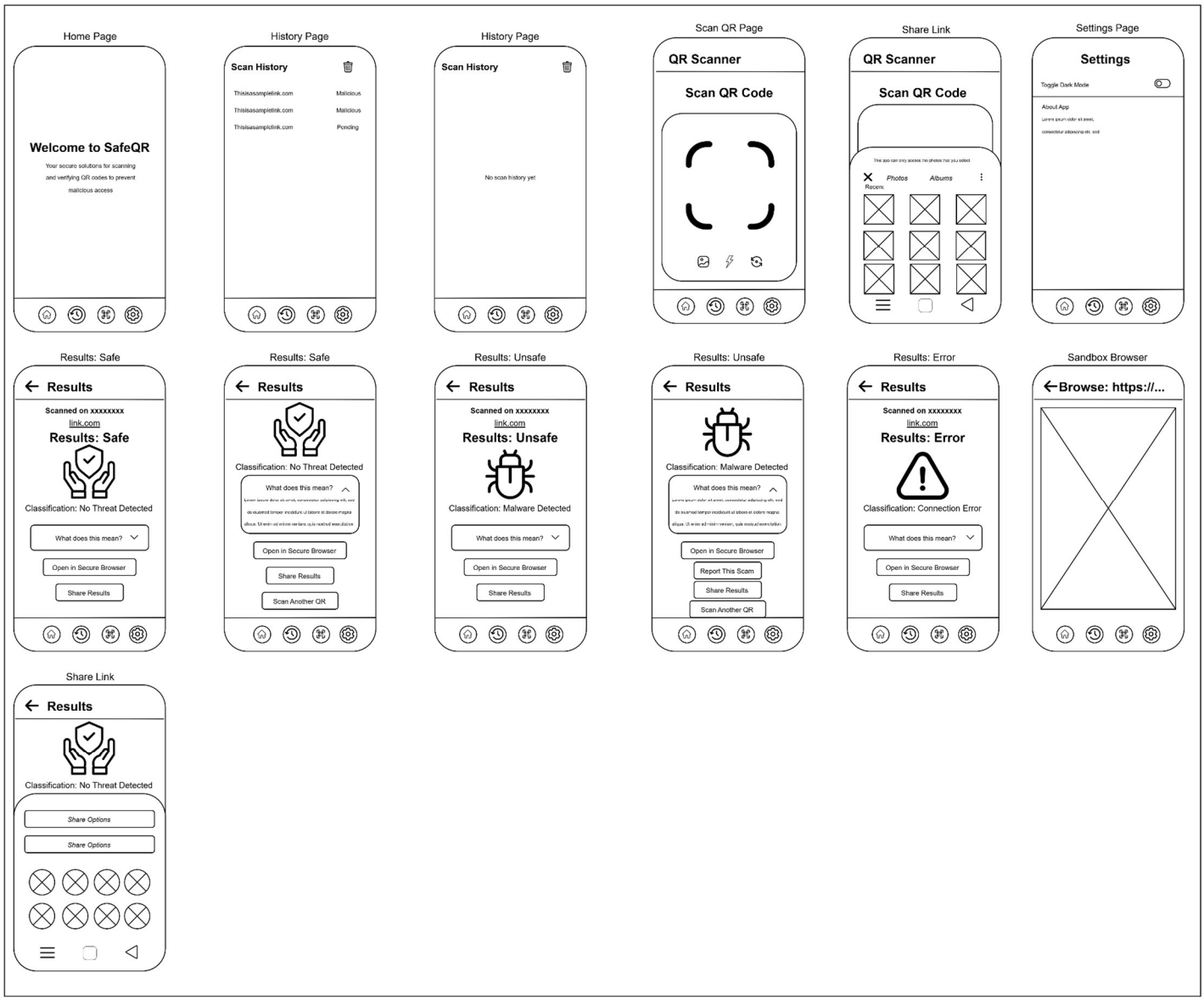
The wireframes in this document serve as a preliminary UI design to the design and structure of the final application. These

wireframes are not final and are subjected to changes based on updates on functions, as well as user feedbacks and technical considerations.

Preliminary Wireframe



Final Design Wireframe

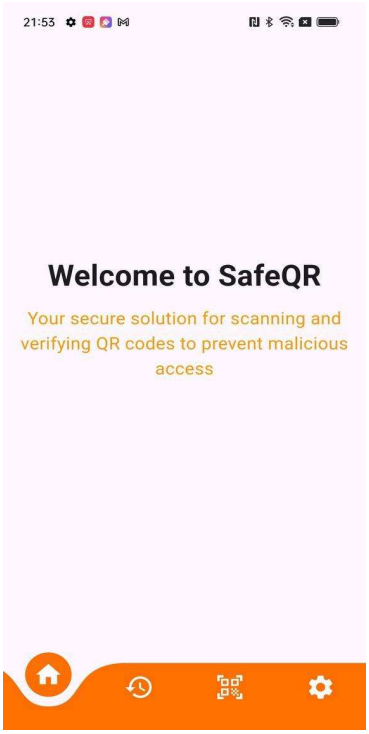

Home Page
Welcome to SafeQR
Your secure solutions for scanning and verifying QR codes to prevent malicious access
History Page
Scan History
Thisisasamplelink.com
Malicious
Thisisasamplelink.com
Malicious
Thisisasamplelink.com
Pending
History Page
Scan History
No scan history yet
Scan QR Page
QR Scanner
Scan QR Code
Share Link
QR Scanner
Scan QR Code
Photos
Albums
Settings Page
Settings
Toggle Dark Mode
About App
Results: Safe
Results
Scanned on xxxxxxxx
link.com
Results: Safe
Classification: No Threat Detected
What does this mean?
Open in Secure Browser
Share Results
Results: Safe
Results
Classification: No Threat Detected
What does this mean?
Open in Secure Browser
Share Results
Scan Another QR
Results: Unsafe
Results
Scanned on xxxxxxxx
link.com
Results: Unsafe
Classification: Malware Detected
What does this mean?
Open in Secure Browser
Share Results
Results: Unsafe
Results
Classification: Malware Detected
What does this mean?
Open in Secure Browser
Report This Scam
Share Results
Scan Another QR
Results: Error
Results
Scanned on xxxxxxxx
link.com
Results: Error
Classification: Connection Error
What does this mean?
Open in Secure Browser
Share Results
Sandbox Browser
Browse: https://...
Share Link
Results
Classification: No Threat Detected
Share Options
Share Options

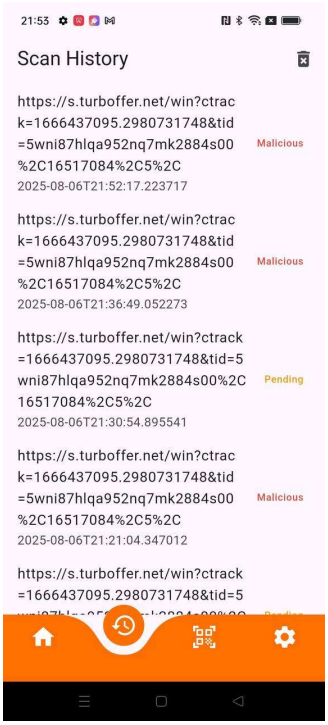
4. Interface Design

This section displays the final design of the SafeQR application, emphasizing the actual user interface of the completely created product. The pictures below depict the main displays and features that users will engage with during their experience.

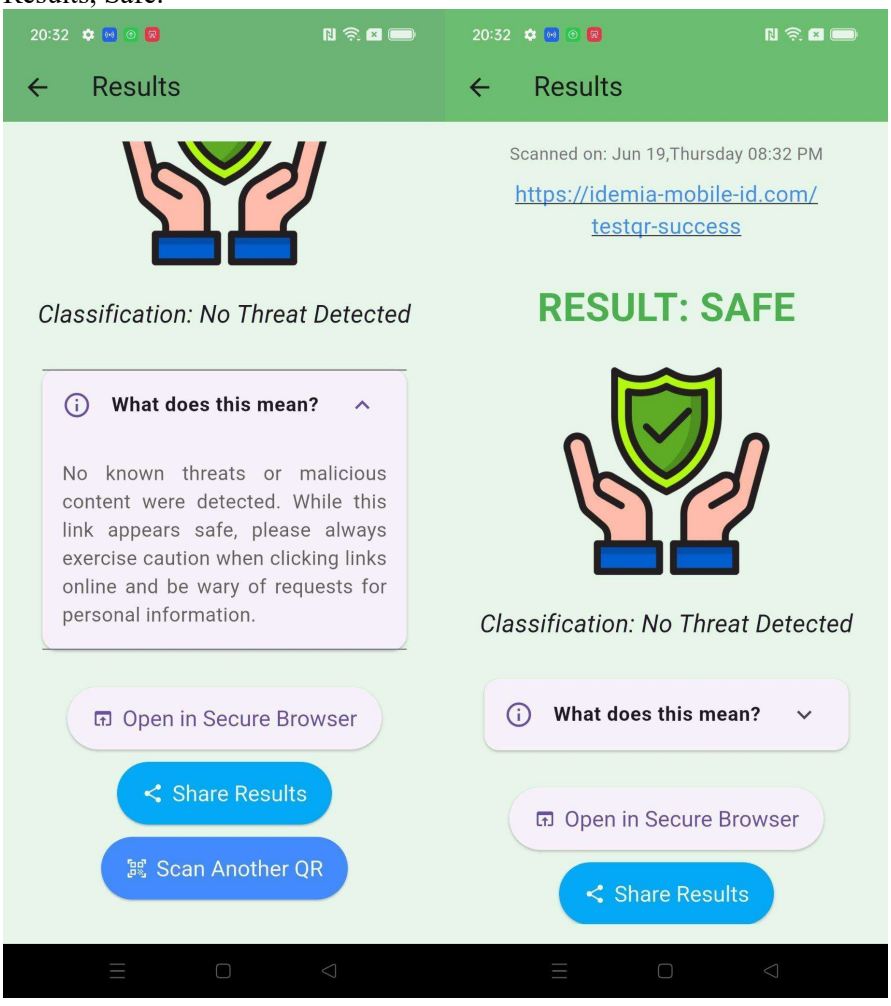
The User Manual provides thorough instructions on how to use all of the app's features. It contains step-by-step instructions, useful hints, and detailed descriptions of each interface and capability, ensuring that users can take full advantage of the application's capabilities.

Home Page, History Page:

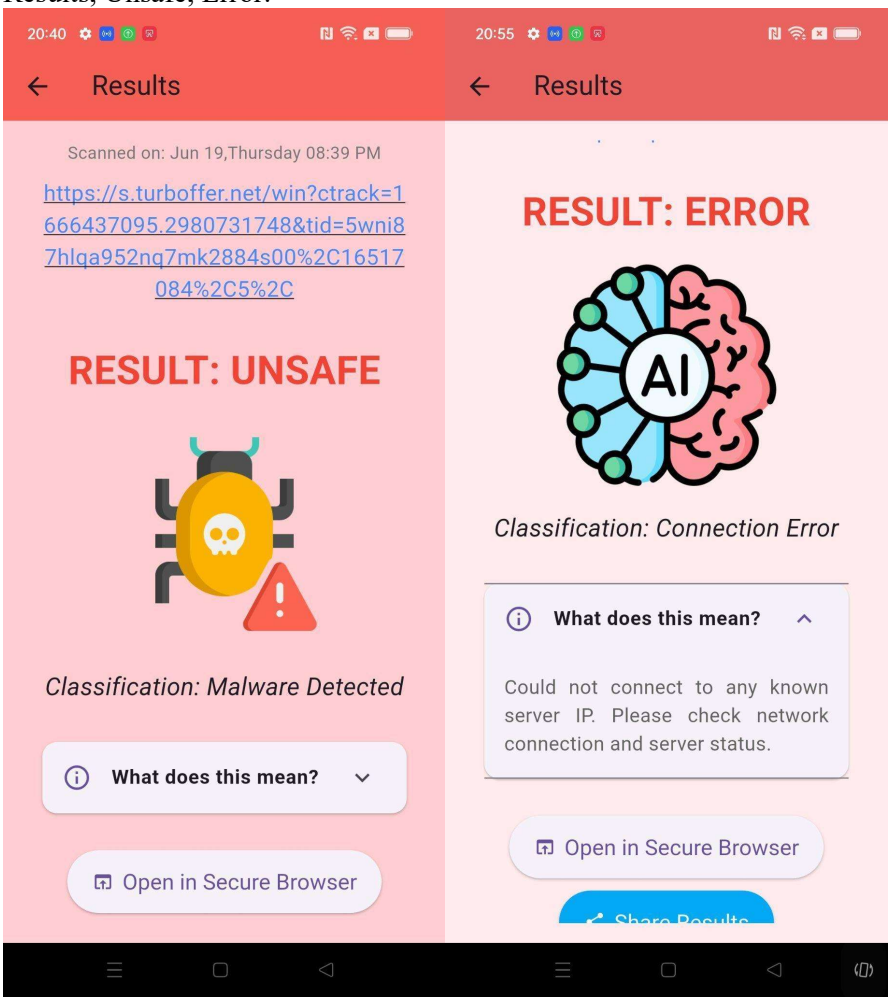


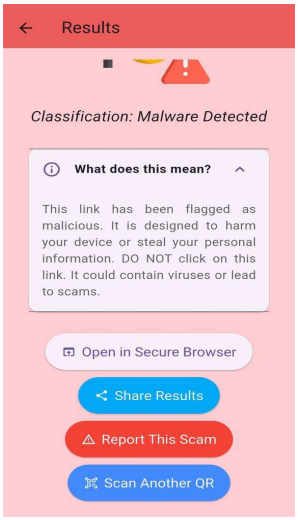


Results, Safe:



Results, Unsafe, Error:





5. Testing

5.1 Test Objective

Objective: The primary goal of this test plan is to assess and validate the functionalities and features of the Safe QR application modules.

This includes: Ensuring all features perform according to specified requirements with a strong focus on security protocols. Identifying, documenting, and resolving any security vulnerabilities, bugs, or areas for improvement to prevent malicious code exploitation.

Expected Outcome: A Safe QR application that is thoroughly vetted for security and functionality, ready for production use.

5.2 Test Cases

1.1 - FYP-25-S2-14P_TestCa

6. Backend APIs

VirusTotal API integration

Purpose

- The API is used to analyze URLs extracted from QR codes against VirusTotal's massive database of known malware, phishing sites, and other cyber threats.
- It provides a **first layer** of security screening before additional checks (like the custom AI model).

Key Features of VirusTotal API

1. URL Scanning
    a. The backend (Flask server) sends the extracted URL to VirusTotal's /urls/scan or /urls/report endpoint
    b. VirusTotal checks the URL against:
        i. 70+ antivirus engines (e.g., Kaspersky, McAfee, Symantec)
        ii. Web reputation databases (phishing, scam, malicious hosting)
        iii. Historical scan results (if the URL was previously analyzed)
2. Threat Intelligence
    a. Returns a comprehensive report with:
        i. Detection ratio (out of how many engines flagged this as malicious)
        ii. Threat categories (malware, phishing, spam, etc)
        iii. Timestamp of last analysis
3. Rate Limits & Caching
    a. VirusTotal's free tier has rate limits (eg. 500 requests/day, 4 requests/min)
    b. Flask backend caches results to avoid redundant API calls for same URL
4. Response Handling
    a. The backend parses VirusTotal's JSON response to classify the URL as
        i. Safe (0 detections)
        ii. Suspicious (low detection rate or mixed reputation)
        iii. Malicious (high detection rate)
        iv. Error (invalid URL or API failure)

Custom List

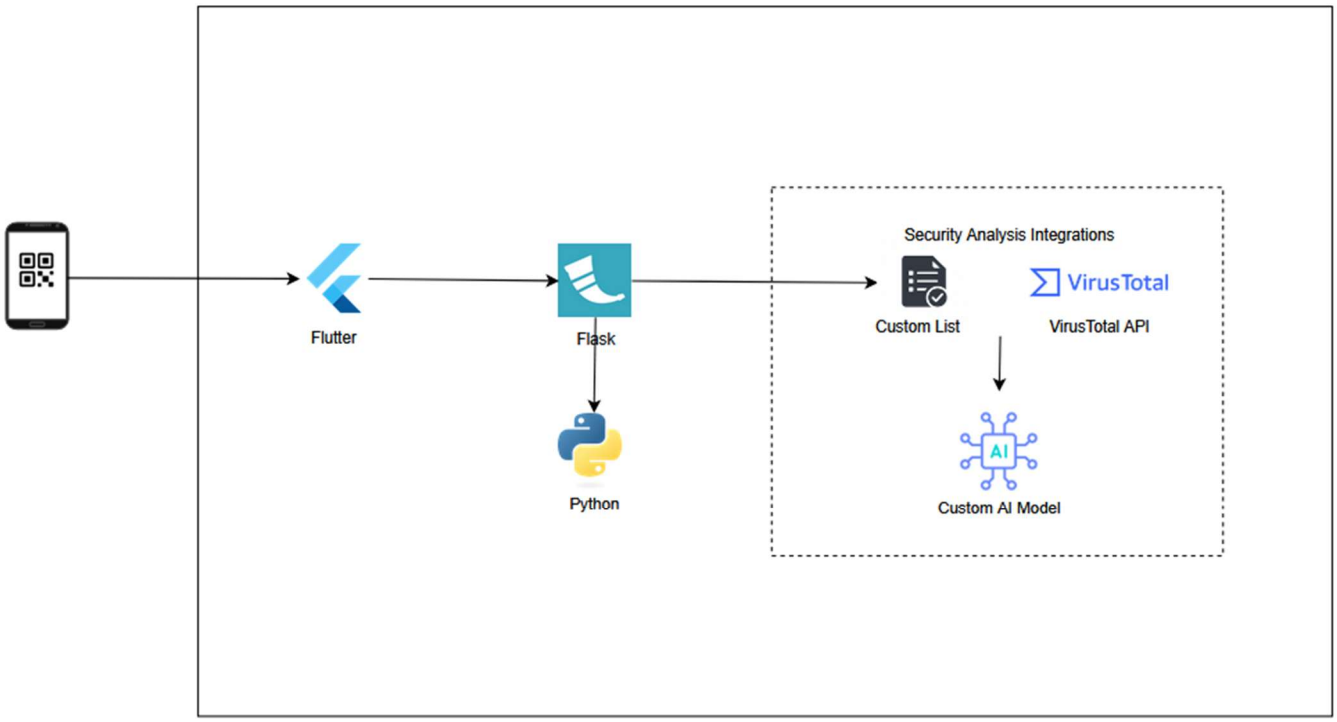
Purpose

- Acts as an additional first-line blacklist to detect malicious domains
- Help conserve VirusToal API calls by pre-filtering URLs against a trusted community-maintained threat list

Key Features of Custom List

1. URL Checking
    a. When the URL is extracted from a QR Code, it is first checked against the blacklist.txt blacklist file.
    b. If the domain matches the blacklist, the URL is immediately flagged as Malicious and VirusToal API Call is skipped
    c. If not found in blacklist, proceed with VirusTotal scanning
2. Caching
    a. Results from Custom List and VirusTotal checks are cached for efficient repeated lookups

7. System Architecture



7.1. Technical Design Considerations

1. Operational Effectiveness:

Operational effectiveness emphasizes consistency in execution and the smooth functioning of deployed systems. It focuses on clear process ownership, automation of repetitive workflows, and rapid incident detection.

To maintain high standards, systems should integrate health checks, auto-recovery mechanisms, and automated deployment pipelines. Regular evaluations and iterative refinements ensure alignment with evolving technical and business needs.

2. Security:

Securing applications and infrastructure is critical to protecting user trust and maintaining system integrity. This involves enforcing strong access controls, safeguarding data through encryption and deploying detection tools for potential threats.

A sound security approach uses defense-in-depth strategies, it combines authentication, logging, auditing and regular vulnerability assessments. These minimise attack surfaces and respond swiftly to anomalies.

3. Economic Efficiency

Optimising for cost ensures that the system delivers value without unnecessary spendings. This includes having sufficient compute instances, avoiding idle resource allocation, and leveraging cost-effective managed services.

Proactive cost reviews and usage analytics help identify inefficiencies and opportunities for automation to further reduce operational overhead.

8. Technology Used

Our QR Malware Detection project leverages a combination of robust software and frameworks across its development, frontend, backend, and security analysis integrations:

8.1 Development Environment

Android Studio: The integrated development environment (IDE) primarily used for building, debugging, and running the Flutter-based cross-platform application, especially when targeting Android devices and emulators.

8.2 Frontend Application

Flutter: A Google-developed UI toolkit enabling the creation of natively compiled, cross-platform applications from a single codebase. This ensures a consistent and seamless user experience for the QR code scanning, analysis results display, and secure browsing features [cite: image_7aad83.png].
Dart: The programming language used for Flutter development.

Core Flutter Packages:
http: For making network requests to the Flask backend and external APIs.
intl: For internationalization, specifically used for formatting dates and times.
webview_flutter: For embedding a web browser (secure sandbox) directly within the application to view URLs.
share_plus: For enabling users to share analysis results easily.

8.3 Backend Server

Flask: A lightweight and flexible Python web framework that serves as the project's backend. It handles incoming API requests from the Flutter app, orchestrates the URL analysis process by communicating with

external services, and provides results back to the frontend [cite: image_7aad83.png].
Python: The primary programming language for all backend logic, including API integrations, data processing, and the AI model (or its placeholder).
Python Libraries:
requests: For making HTTP requests to external services like VirusTotal.
python-dotenv: For securely loading environment variables (like API keys) from a .env file.
json: For handling JSON data parsing and serialization.
time: For managing time-related operations (e.g., timestamps in the analysis cache).

Security Analysis Integrations
VirusTotal API: A critical component for the initial and extensive layer of URL detection. The backend server queries VirusTotal to scan URLs against a vast repository of antivirus engines and website scanners for known threats [cite: image_7aad83.png].
Custom AI Model (Current State: Placeholder Logic):
Your project incorporates an "AI model for a second layer of detection" [cite: image_7aad83.png].
Currently, the AI model's functionality in server.py is implemented using placeholder string-matching logic. This means it classifies URLs based on whether the literal terms "malicious" or "suspicious" are present within the URL string itself (e.g., if "malicious" in url.lower():).

9. References

https://docs.virustotal.com/reference/overview

https://flask.palletsprojects.com/en/stable/

https://www.geeksforgeeks.org/python/how-to-run-a-flask-application/

http://flutter.dev/

https://cert.pl/en/warning-list/