Final Project Report

Executive Summary:

A QR code scanner application that puts the user's safety first. Using integrated APIs from trusted vendors, the app is intended to warn users of malicious intent behind the QR code they scanned.

Final Requirements and comparison with initial requirements:

Initial Requirements:
User friendly graphic display
Heads up alert system
QR code bookmarking
Malicious site database checks
Website sneak peeks
Anti Session hacking
Certificate Checking

Final Requirements:
User friendly graphic display
Malicious site database checks
Anti Session hacking

Final timeline and comparison with the initial timeline:

Initial Timeline:
Week 4-6: Sprint creation of app and camera abilities, review products
Week 7-9: Sprint utilizing API for HTTP header gathering and analyzation and database searches, review products
Week 10-12: Sprint utilizing API to obtain certificates and analyze it for security, review products
Week 13-14: Sprint to add additional features, review products

Final Timeline:
Week 4-7: Sprint creation of app and camera abilities, review products
Week 8-12: Sprint utilizing API for HTTP header gathering and analyzation and database searches, review products
Week 13-14: Sprint to add additional features, review products

Due to our issues with plugging in the Google Safe search API for many weeks we fell behind on many features we wanted to put in to begin with.

Project results compared with expectations:

a) Which of the initial use cases are functional?

Scanning a QR code
Scanner reads URL from QR code
User is redirected to URL

b) Which use cases have been removed/added/modified?

Preview Feature

Software evaluation:

a) Functionality: Which testing methods, tools and steps have been followed? What was the testing timeline? Which functionality issues are still open?

Functionality: Emulation and running the application through Android studio was our testing method,
which would follow the implementation of the API or code then the testing of it.

b) Security: Which security methods and tools have been employed? What were relevant observations? Which security issues are still open?

Security: Kotlin in itself has security aspects built in, as it is a branch off java that is more modernized.
We also stuck to using well known developers for APIs such as Google.

Work to be done:

Further development of features
Increased fluidity between chosen APIs
GUI feature to be further developed
URL redirect feature