SafeHouse

CA326 Technical Specification

Eamon Crawford and Sean Hammond

Table of Contents

1. Introduction

1.1 Overview

Safehouse is an app developed to offer users a convenient method of locking and opening doors through the use of the NFC functionality of their smart devices. It allows users to securely open their door by scanning an NFC card and inputting their fingerprint to authenticate the open request. Authenticated users can be added to the lock to ensure that the owner of the door knows exactly who has access, and has complete control over this.

The system is being developed to accommodate convenient management of access to specific buildings, apartments or doors. For example, this system would be useful for people who own an AirBnB and want to ensure that only the visitor can enter the property without the need to supply a key. Offices could also make use of this system, easily allowing access for a large number of people without compromising security.

1.2 Business Context

The primary business context for this product is the sale of the lock system that must be installed onto the desired door, and the appropriate NFC cards. While any NFC cards of the appropriate type would work, the product would be sold with cards together for convenience to the customer.

1.3 Glossary of Terms

- **NFC:**
  Near **F**ield **C**ommunication. Short-range wireless technology that enables simple and secure communication between two devices.

- **MongoDB:**
  An open source database management system that uses a document oriented database model, using documents and collections instead of tables and rows.

- **React Native:**
  React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android.

- **Raspberry Pi:**
  A low cost, extremely small PC that is capable of running a Linux OS.

2. System Architecture

2.1 Original System Architecture Diagram

The following diagram gives an overview of our systems architecture.

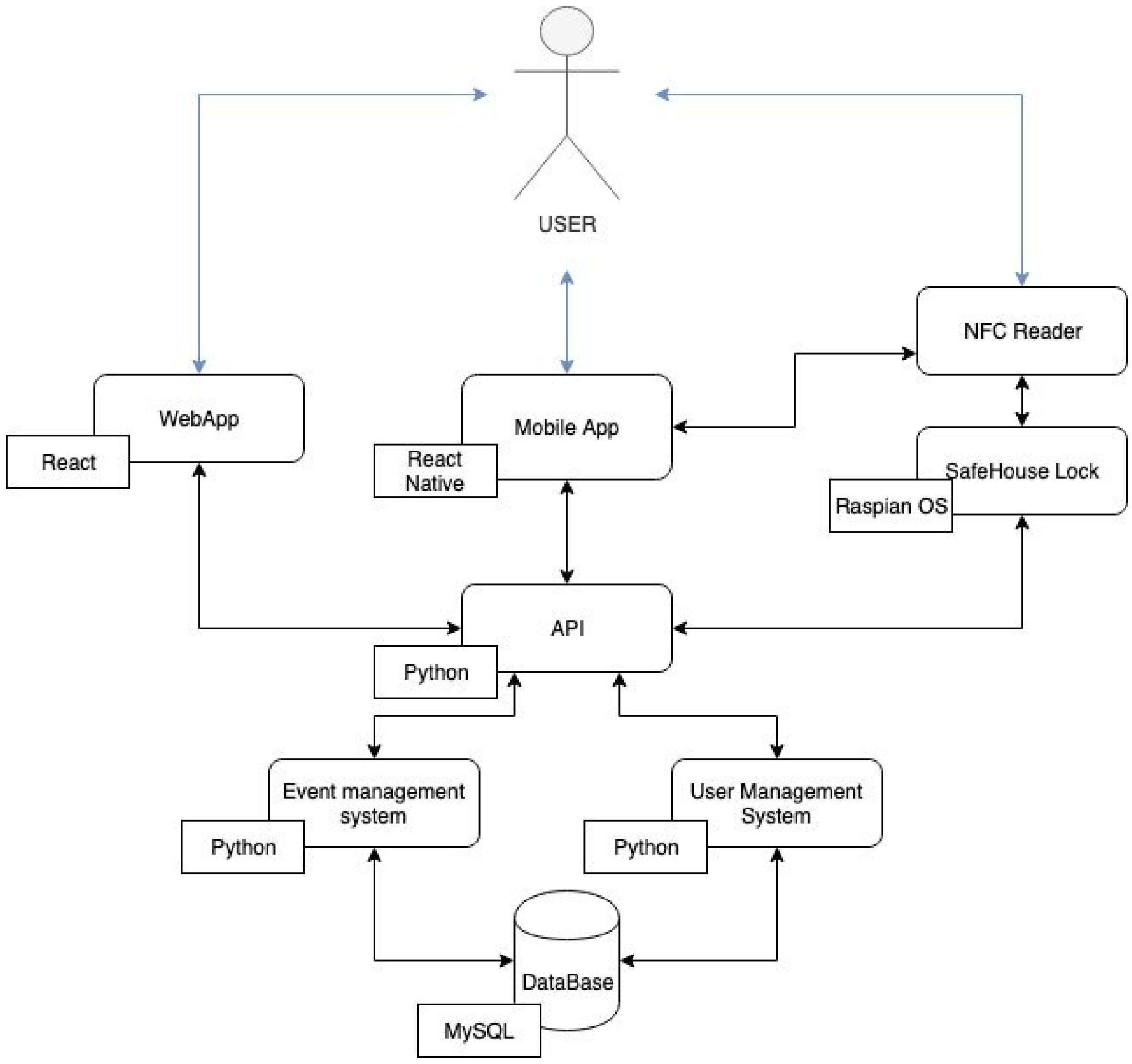

Fig. 2.1

2.2 Original System Architecture Diagram Description

Mobile App

The mobile app is the main interface the user will interact with. It will be an Android app, written in ES6, JSX and some Java using the React-Native Framework and Android Studio environment. There are also plans for an IOS, which will similarly be written in ES6 and JSX using the React-Native Framework. However, it will be written in the XCode development environment and may have to use some Objective-C. Using React Native will hopefully allow us to reuse a lot of code without modification for the two platforms.

Web App

The web app is an alternative interface a user can use. It will have most of the functionality of the mobile apps but will not contain the NFC functionality. It will be Written in React using JSX and CSS.

NFC Reader

The NFC reader will allow us to read NFC from the mobile app and NFC cards. We will use prebuilt drivers for this.

SafeHouse Lock

Using Python we will interpret messages from the NFC Reader on the SafeHouse Lock. The platform will be built on Raspbian OS on a Raspberry Pi.

API

The API will provide the mobile app, web app and SafeHouse Lock with requested data and will push events to these systems. It will be the broker between the backend services and these systems. It will be written in Python.

Event management system

The event management system will handle responding to, and firing events off events to, the rest of the system. These will mainly be triggered by user actions and come through the API but will also include scheduled jobs. It will be written in Python.

User Management System

This component will handle User data. This includes credentials and any other logic. It will be written in Python.

Database

The database will store all information on users and devices on our system. It will be queried by MySQL.

2.3 Final System Architecture Diagram

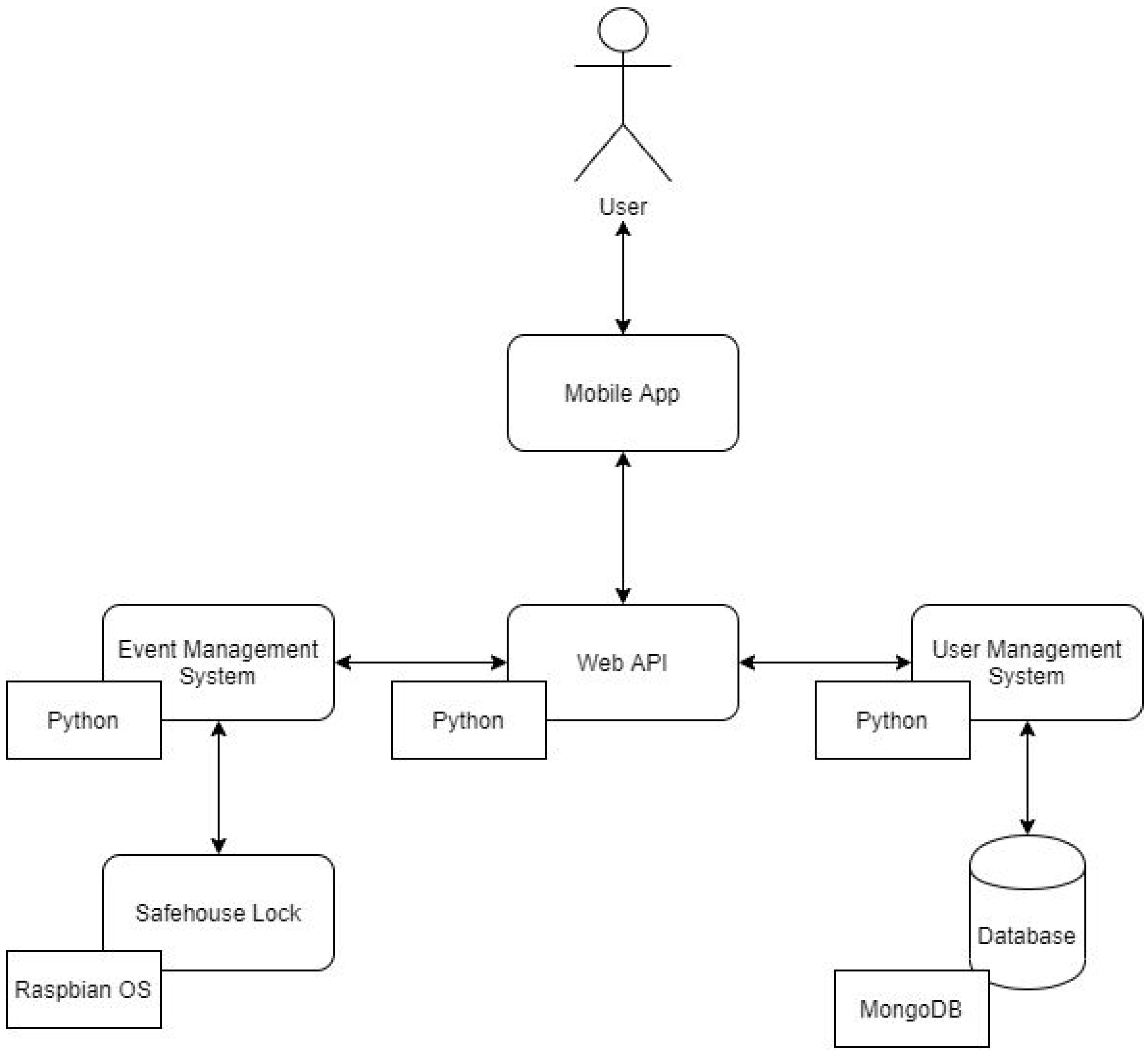

Fig 2.3

2.4 Final System Architecture Diagram Description

Mobile App

The mobile app is the main interface the user will interact with. It will be an Android app, written in ES6, JSX and some Java using the React-Native Framework and Android Studio environment.

SafeHouse Lock

Using Python the SafeHouse Lock checks the web API for messages. The platform is built on Raspbian OS on a Raspberry Pi.

API

The API provides the mobile app, web app and SafeHouse Lock with requested data and will push events to these systems. It is the broker between the backend services and these systems. Written in Python.

Event management system

The event management system handles responding to, and firing events off events to, the rest of the system. These are mainly triggered by user actions and come through the API but also include scheduled jobs. Written in Python.

User Management System

This component handles User data. This includes credentials and any other logic. Written in Python.

Database

The database stores all information on users and devices on our system. Queried and hosted by MongoDB.

3. High Level Design

3.1 Original High Level Design Diagram

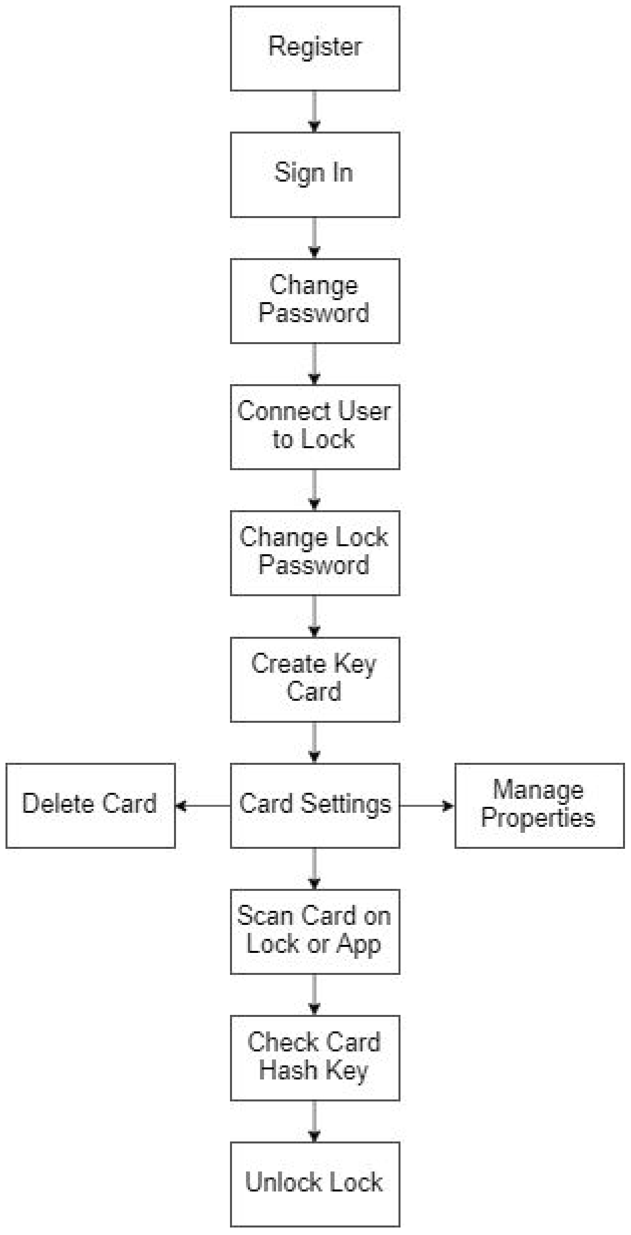

fig 3.1

3.2 Original High Level Design Description

Fig 3.1 is explained below.

- **Step 1: Register**
  Register an email and password to grant access to the app.

- **Step 2: Sign In**
  Log in to the app using the registered email and password.

- **Step 3: Change Password**
  Once the user has logged in they can change their password.

- **Step 4: Connect User to Lock**
  Connect user account to desired Lock by searching over network or scanning mobile device.

- **Step 5: Lock Settings**
  Once the user has linked their account and the Lock, they can change the password for the Lock.

- **Step 6: Create Key Card**
  Creates a key card by writing to an NFC card using the mobile device's NFC functionality.

- **Step 7: Card Settings**
  Once a card is created, the user can edit the card settings.
  - **Manage Properties**
    The user can change the card so that it expires after a certain number of uses, or after a certain amount of time has elapsed.

  - **Delete Card**
    The user can delete the card so that it cannot be used to open the Lock.

- **Step 8:Scan Card on Lock or App**
  The user can scan their card key on either the Lock or their app, using the NFC functionality of either.

- **Step 9: Check Card Hash Key**
  Once scanned, the card's contents are checked to see if the hash key stored is valid and if the key has not expired or been deleted.

- **Step 10: Unlock Lock**
  Once the Lock has confirmed that the key is valid, it will unlock.

3.3 Final High Level Design

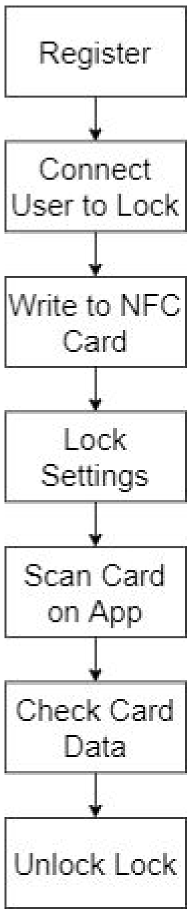

fig 3.3

3.4 Final High Level Design Description

Fig 3.3 is explained below.

- **Step 1: Register**
  Register an email to grant access to the app.

- **Step 2: Connect User to Lock**
  Connect user account to desired Lock by scanning given QR code. They will then name the lock.

- **Step 3: Write to NFC Card**
  Creates a scannable card by writing to an NFC card using the mobile device's NFC functionality.

- **Step 4: Lock Settings**
  Once the user has linked their account and the Lock, they can add users to the lock.

- **Step 5: Scan Card on App**
  The user can scan their card key on their app, using its NFC functionality.

- **Step 6: Check Card Data**
  Once scanned, the card’s contents are checked to see if the key stored is valid.

- **Step 7: Unlock Lock**
  Once the Lock has confirmed that the key is valid, it will unlock.

4. Problems and Resolution

During this project’s life cycle, we faced a number of issues and problems in terms of both our initial conceptual design, and the programming aspect.

Our first major problem that we ran into was that we felt as though our initial design for the project lacked a specific focus. We wanted users to be able to unlock the lock from both their door and on their phone using NFC cards but also we wanted to consider also allowing the user to use only their phone to unlock the door, both from up close by scanning the phone on a scanner at the lock, or by entering a password. With such a large number of redundancies, it was difficult to start at any one place, and we came to our supervisor with this issue. He managed to help us resolve this issue by putting forward a simpler, less convoluted design, which was to simply have one card per lock, that a user can scan their phone on and authenticate themselves with their fingerprint to open the lock. We felt this was a much more streamlined, and ultimately more convenient project than the previous idea.

Another problem that we encountered during the project was that we found we were unable to make HTTP requests from the app to the web API we had developed. This was crucial for our project to work and was extremely worrying, because if we couldn’t solve this problem we would have to completely rework the project. After researching as much as we could about Axios, the library we used for GET and POST requests in JS and trying every example we could find to discover what we had wrong, we finally realised that the issue wasn’t in the code. The Wi-Fi we were using was the DCU Guest Wi-Fi, which doesn’t allow devices to connect to Redbrick’s servers, which is where we were hosting the web API. After resolving the issue and successfully being able to make HTTP requests, we were made more aware of the importance of thinking outside the box when it comes to resolving problems.

5. Installation Guide

5.1 Hardware Components Required:

- NFC compatible Android device.

5.2 Provided Components:

- Safehouse Lock
- QR Code
- NFC Card

5.3 Step By Step Installation:

1. Install the Safehouse app by downloading the .aab file from the Safehouse GitLab.
2. Open the Safehouse app, and follow the on-screen instructions to register an email.
3. Plug in the Safehouse Lock for first-time installation.
4. On the app, scan the QR code using the scanner to link your user account with the lock.
5. Scan your fingerprint to complete the registration between the user and the lock.
6. If these steps were completed correctly, the lock should close and open once.
7. Now you may enter a name for the lock.
8. Scan the NFC card against your phone to write the unlock code to it.
9. You are now finished setting up the lock and app. You may now scan the NFC card on the app and scan your fingerprint to open the linked lock.
10. If you would like to add a new user to the lock, you can do so by navigating to “Share Access” and entering their email, if they have made an account already.

6. Appendices

https://reactjs.org/
https://react-native.org/
https://www.raspbian.org/