Property Notification System with Key tracking application

Student Name: Phoophanom Tanprasit
Student Number: C1226125
Supervisor: Dr George Theodorakopoulos
Moderator: Professor. Omer Rana
Programme Name: Computer Science with a year in industry (BSc)
Submitted: 06/05/2016

Abstract

The aim of this project is to build a bespoke property notification system that will assist commercial and residential property managers with specific managerial tasks. The web application will supply dynamic information to the occupiers of residential buildings, informing them of various events that will occur within the property.

Two mobile applications will be developed alongside the web application. The first will be an Android application that will display said notifications, which managers can modify at a moment's notice, through the online dashboard. The other is a mobile application that logs times when a master key gets taken and returned from a cabinet. Property managers will also be able to distribute pin codes through the application, replacing the current email system. Additionally, property managers will be able to view the times when a key had been taken and returned through the web application.

The web application had been developed on a PHP Framework called Laravel in conjunction with various web languages and libraries. Both mobile applications were developed on Android Studios, with Java and XML.

Acknowledgements

I would like to thank my supervisor, Dr George Theodorakopoulos for his help and support throughout this project. His weekly feedback sessions helped guide the project in the right direction. Also, I would like to thank my parents for providing me with various project ideas and supporting my time at Cardiff University.

Table of Contents

Table of Figures

Introduction

To most businesses, time is considered to be the most important resource available. In today's market in order to compete and succeed against other competitors, their workforce must work at optimal efficiency, be highly adaptable and provide rich digital experiences to their clients.

"Companies that successfully embraced digital transformation invested well in both technology-enabled initiatives and the leadership capabilities required. In return they grew revenues by 9pc and are, on average, 26pc more profitable than their industry competitors" (Freakley, 2015).

Businesses that lack digit integration will lose out on customers, as more companies are expected to provide fast and digitally rich services. As popularity of mobile applications is at an all time high, new and sophisticated applications are being developed to provide newer better solutions to existing problems.

In order to provide the best services, bespoke digital infrastructure will be needed to order to efficiently store and securely transmit data. Furthermore, by taking an agile approach the intended audience can provide constant feedback and improve the overall user experience by adding quality of life updates.

The property notification system was envisioned as a web application that property managers could access in order to supply up-to-date information to notification-enabled devices, located at various locations around the UK. The notification application can be deployed on to any Android enabled devices, as long as Internet and cellular connection are available.

Additionally, my client requested a key tracking application and he had asked for it to be integrated into the web application. Property managers who will now be known as the administrators will provide contractors with access to master keys via pin codes. The mobile application will audit the time the keys had been taken and returned to the cabinet.

Background

Property Management Solutions

Arthur Online

Arthur Online (Ltd, A.O, 2011) provides an online platform where property managers can connect with contractors, letting agents and owners via the web and mobile applications.

Users of the application can input and transmit data securely making sharing information safe and convenient. Allowing better more fluid explicit data transfers between users of the system.

Their interactive platform can audit all forms of data like financials and correspondence documents between users, allowing property managers to access all archives in the future.

The online dashboard is very minimalistic and simple to use. The user interface adjusts to fit the user's needs. For example, property managers can view their portfolios, while contractors can view their list of tasks and invoices.

Tenants on the other hand can view their monthly rent statements and raise any concerns that they may have with their lease. Personally, I felt that the application was well organised and the user interface looked very professional compared to other solutions available.

The mobile application contains most of the features available on the web platform. Similarly, the user interface was well organised like their web counterpart. Property managers can converse with tenants and contractors remotely through the app, in order discuss an issue that requires immediate attention. See Appendix 1.

Their one major downside that I had notice was the price plan of the application. It can be quite expensive, at £1.50 per property per month. They offer extra services like business hours helpline and live training. However, they are willing to lower the price per unit depending on the size of portfolio.

Additionally, users of the application must have a compatible smartphone. Arthur Online offers a complete package for residential management, however the application itself may be overwhelming for some tenants due to the amount of features that are available.

Buildium

Another good web based property management software is called Buildium (Buildium, 2016). They offer all the same services similar to Author Online but with a generic app. Therefore, tenants are expected to use the same portal as a property manager, as it is integrated into the management site. Unlike Arthur Online who are UK based, Buildium offers international features like time-zone support, currency conversions and country-specific address fields.

Their user interface is simple and easy to use but feature heavy. Therefore, it does not cater to novice users who are starting out with property management. Their self-help section is known to be out-of-date, thus a member of the customer service team is required for most work around. See Appendix 1.

Buildium allows tenants to submit issue tickets online via the tenant portal. Property managers can turn these tickets into tasks for contractors to be assigned to. Each task will be time stamped and allows tenants to attach any relevant photos that relates to the issue.

However, they charge setup fees for electronic payments and screenings. Which means that for small businesses it can be very expensive.

Key Tracking Solutions

TrackR (TrackR, 2016) is a Bluetooth tracking device, no larger than a coin that can be attached to small items. See Figure 2.

It connects to an app running on a mobile device. The application then tracks the power level of the emitted signals from the Bluetooth device. Depending on the threshold of the emitted signal, the application will alert the user if they stray too far from the linked item. Utilising Bluetooth low energy it consumes 50 times less energy than regular Bluetooth, allowing TrackR to last at least a year.

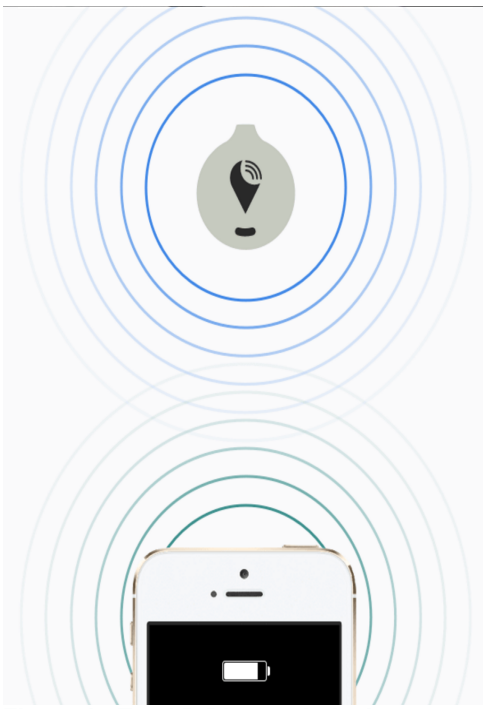


Figure 1 - TrackR Bluetooth device

TrackR includes a unique feature called Crowd GPS. It is different from traditional GPS, as it uses all devices that are connected to their network to locate missing items. Once the item had been located, its GPS location will be transmitted to the original owner of the item.

Each TrackR device costs $30 and considering that property managers can manage over 100 properties individually, this solution is a costly one. However, TrackR is open source, so developers can create custom applications that can interact with TrackR devices.

The problem

A simple Google search for a property notification system yielded zero result, it is considered very niche, a lot of property specialists can benefit from utilising this application, like removing the need to travel long distances to distribute notices to property residents.

Each solution from above was designed with the sole intention of managing properties and its residents. If a property manager wants to inform residents of an event, they will need to contact with the client directly through their email feature. A major problem with this approach comes from residents who do not check their email regularly. So the solution is to place several tablets at various locations where residents will take notice. For example, in a lift, as usually people do not have anything to do and have a few seconds to scan the notification screen.

TrackR is an alternative solution to tracking keys all together. The software tracks the location of the key rather than the time, which is what my client had wanted. However, since TrackR provides APIs to interact with their database, an application can be developed to track whenever a key leaves or enters a building using Bluetooth technology.

Stakeholders

The stakeholders of this project will be small to medium property specialists, as large organisations will have their own solutions to the problem sated previously. Allowing property managers with that organisation to distribute large amount of information to residents at various location. Additionally, the key tracking application will and web application will provide a platform for them to organise key distributions. Both notification and key tracking applications are lightweight mobile applications, so they can easily be deployed once the web application is configured

Aim and Objectives

The aims of this project remain the same, as specified in the first report. That is to build an online web application that can aid property managers with their managerial tasks. Such as, supplying information to occupier of residential buildings via digital devices.

A secondary aim is to create key tracking application. Its target audiences are contractors who had been contracted by a property manager to resolve an issue. A key will contain a pin code; this pin code can be used to unlock a master key cabinet. Whenever a pin code gets revealed, the application will automatically log the timestamp. Then when the contractor returns the key, he can log the time, so that he can no longer be accountable if the key was later stolen or missing.

The aim of this report is to explain the development processes that had taken place throughout the project. The project will require the combined knowledge of my time at Cardiff University and my industry experience as a developer. Describing how each application was built and the challenges I had encountered.

Three separate applications will be built. First, a web application that allows managers to login and view property and contractor information. In addition, managers will be able to create, edit and remove notifications that will be displayed within a property. The second a mobile terminal app that display said notifications and provide useful information like the local weather.

Lastly, a mobile app that contractors can sign in and view keys, each key will have a hidden pin code. These pin codes must be activated in order to unlock a master key cabinet. The application will track the time when a key had been taken and returned to a key storage box.

The first stage involves requirement gathering. The requirement gathering process for this project requires less work than most projects, as a property specialist gave the project idea to me.

He had already thought about the issues that exist and solutions to them. He had noted how they were still using paper to distribute information to property residents and that a display terminal or TV should be used instead in conjunction with a web app, enabling him to supply relevant information and up-to-the-minute notifications.

Additionally, he had mentioned that providing pin codes and keeping track of contractors can be time-consuming. Every time a contractor needs access to a property, a property manager will need to find the contractor’s contact information and email him a pin code in order to obtain a master key. A new system should be in place where they can view a list of keys that they had or will be assigned to them. An alert would also be desirable that will trigger when a contractor had taken a key and not returned within a specified time.

The second stage of the development cycle involves converting requirements into user case, user flow and entity and relationship diagrams. Additionally, I had designed various user interfaces for the three separate applications. However, live testing was only performed on the mobile applications. Furthermore, I had created test cases in order to test the functionality of my applications.

The third stage will include the implementation of my applications and the various approaches that I had taken in order to solve the problems that I had encountered. Additionally, I will specify the Frameworks that I had used and any libraries that I had integrated into my applications.

The fourth stage is carry various tests on the applications and noting down the results.

The fifth and final stage will involve taking the results from stage four and evaluate the web and mobile applications.

An agile approach will be used to tackle to project, in order to accommodate for the changes in requirements, time constraints and issues that may block my progress. Hopefully, this will allow three minimum viable products to be produced before the project's deadline, with the main features of each application working correctly. While a lot of time had been given to setup an environment to host the web application, there was not enough time to deploy the application on Amazon Web Services, which would allow the application to be accessed globally and tested at vast geographical locations.

Approach and Designs

Overview

To complete the aims and objectives that for this project, I had separated out the property management system into three separate applications:

- Web application for property managers
    - Must be able to assign notifications to devices
    - Must be able to distribute keys to contractors
- Android notification application for residents
    - Must automatically setup the device
    - Must be able to display notifications and local weather
- Android key tracking application for contractors
    - Must allow contractors to login
    - Must have professional looking interface
    - Must allow contractors to retrieve pin codes
    - Must be able to update the key taken and return times

Changes from initial report

- The notification application will no longer provide local traffic information, due to lack for free APIs
- Securing the data transmission became a secondary aim
- The notification application currently does not provide Android TV support, as the client had decided that this feature is no longer needed

Development Tools and Frameworks

Laravel

The web portal was developed on a PHP framework called Laravel 5.2. It uses a Model View Controller (MVC) framework; MVC paradigm enables developers to separate out the presentation like user interface from the object models that we perceived in the real world. These models were to be closed off and functions must be able to perform correctly without referencing the presentation.

For the backend, a version control type system was used to construct the database. They allow developers to modify the database schema and still keep their schema up-to-date. The reasoning behind it was to prevent any inconsistencies between developers, however with a project with one developer it enables easy expansions and modifications to the database in the future without having to redesign certain features

from scratch. Laravel also includes a schema builder, allowing for easy schema design and management.

The default objects relational mapper (ORM) that Laravel uses is called Eloquent. It is also very effective handling data set that have a lot of relations, providing multiple of helper functions to perform create, update and delete (CRUD) operations; In addition to making complex database queries. For the presentation, it uses a template engine called Blade, which is easier to use and make HTML code cleaner and easy to read. Laravel offers protections from cross-site scripting (XSS) and cross-sites resource forgery (CSRF) in addition to providing login capabilities straight out of the box.

Android Framework

Android was used to develop the two mobile applications. The reasoning was that Android devices are generally cheaper than the IOS counterpart. Meaning that the cost to deploy and maintain these devices will be low. Additionally, many open source libraries are available to use in Java, which developers can use in order to reduce development time. From a developer's point of view Java is ideal for this project, as it requires little time to setup the environment.

A superior solution is to build for IOS and Android at the same time using Xamarin. Xamarin, is a tool that allows developer to code Business logic, database uses and network access in C# once and reuse on any platform. To which developers can customise further depending on the end user needs.

One noticeable downside of using Xamarin is the cost, at $80 per developer on an on-going monthly basis. Second, Xamarin does not support all native features and user interfaces; therefore developers will not have access to the latest features and enhancements until Xamarin provides APIs to support them. Furthermore, having a smaller ecosystem than IOS and Android means finding support can be difficult, as documentations can be scarce. Lastly, the developer must be adept at C#, .Net, Android, IOS and Xamarin frameworks and Xamarin IDE. In addition, developers must have prior knowledge of each platform's lifecycle and user interface designs.

GIT

For this project GIT will be used as the version control system. By creating snapshots of the project and tracking changes made to the project. Git allows for branching, which means that a new feature or user story can be worked on independently. This enables the project to follow the agile approach by working on multiple stories at once and merge at the end of a sprint in order to produce minimum viable product. As bonus GIT offers rollbacks methods, which means that, any mistakes is negligible and reversible.

Bootstrap

To handle the visual side of the web application I had decided to use Bootstrap 3.3.5. Bootstrap employs the grid system by resizing contents depending the on the device's screen size. Thus creating a responsive website requires little to no extra work.

For a web developer a front-end framework like Bootstrap allows new websites to be made quickly. Instead of wasting time researching coding from scratch, Bootstrap provides examples on their documentation site. Additionally, it provides cross platform support a feature that will take several hours to complete.

Being a framework ensures that the web application looks the same regardless of the platform and it is highly customizable. Bootstrap provides a customizable page where developers can choose the features that they want their site to have. This can become useful during the optimization phase, where loading time can be improved even by a little.

Trello

A Kanban board was also used during the development, as a management tool. The application is called Trello. Using an Agile approach I had converted the project's goals and requirements into user stories. These user stories were then converted into cards on Trello, with the cards a user can place it under five lists applications web portal, notification app, terminal app, in progress or completed. The completed list refers the completed tasks for the current build not the finished product.

Choice of Development Methodology

There are two major software methodologies that most projects follow, Waterfall or Agile. Most developers will consider water as an out-dated concept the reasoning is that waterfall relies on well defined set of requirements Thus, any changes in the future will greatly impact the project progress. In theory it makes sense, however the reality is that nothing stays static. New or mainstream ideas gets introduced that can force changes in the project's requirements in order to compete or increase performance. This can have a major effect on the budget and time, as it was not accounted for during the requirement gather stage.
Agile on the other hand is the polar opposite. Where developers set out to accomplish a minimum viable build every two weeks. At the meeting the new build with new features will be demonstrated. Feedback will be given and used to guide the project ensuring that the product will be always be on track and relevant.

As mentioned before Git was the version control system of choice in order to guarantee that a working build is always available. Git uses branches to separate the master build from the ones that contains features that are under progress.

I had decided to use a hybrid of Waterfall and Agile, utilizing the advantages of both like the transparency of Waterfall and flexibility of Agile. First taking down the requirements, then reverting to Agile to adapt to the change in requirements allowing for continuous delivery. A hybrid model is very good at handling products where reusing of code is possible. Allowing for quick turnaround time.

System Architecture

The system architecture can be broken down into four parts (See figure 3):

- The first is a relational database that will house property and contractor information.
- Second is an Android application client that can display notifications and weather information to residents within a property.
- Third is an Android application that can be used by contractors to securely receive pin codes.
- Lastly, a web portal that property managers can log in and distribute notifications to devices and assign keys to contractors.

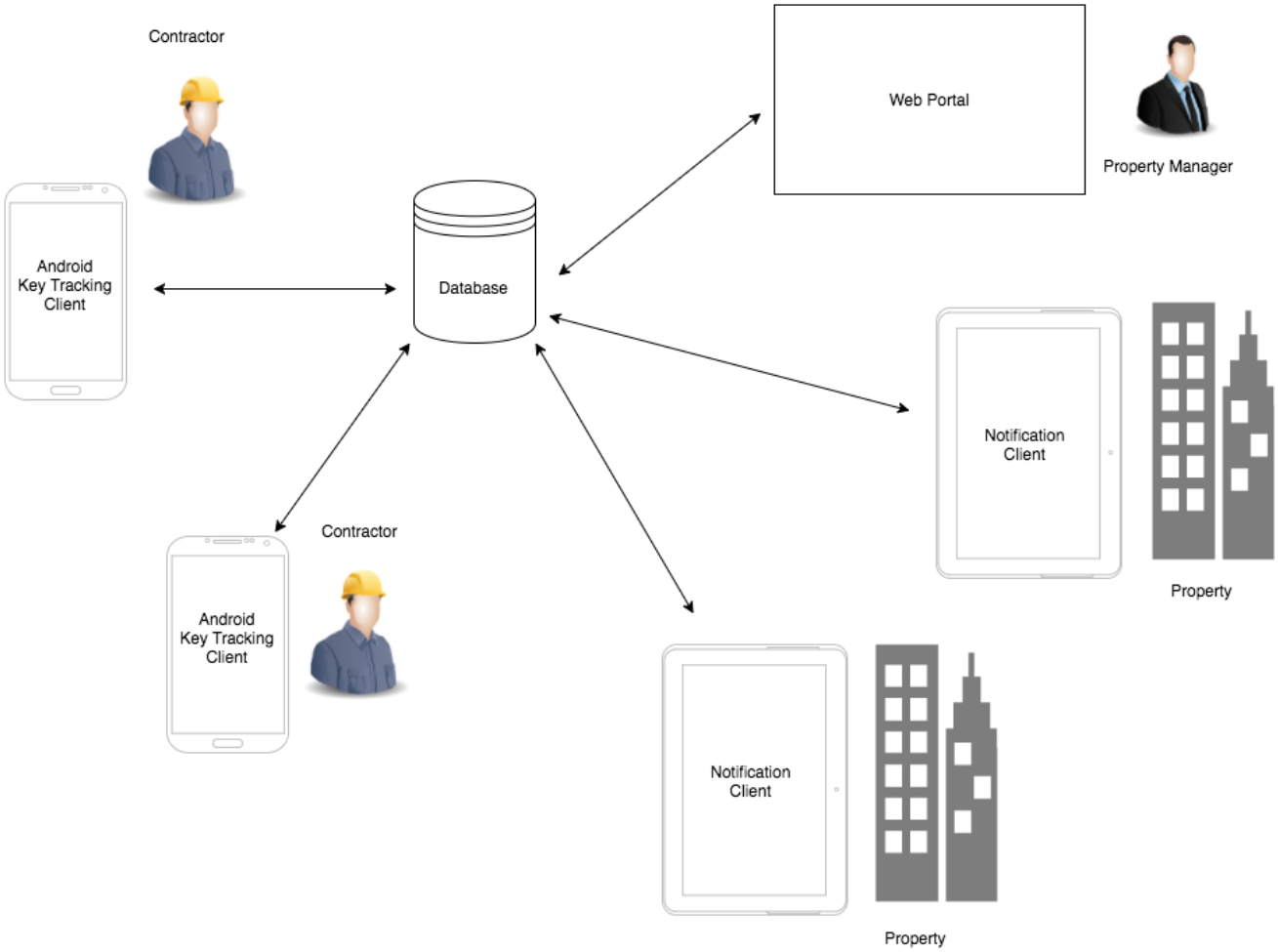


Figure 2 - Overview of the property notification system and key tracking application

Database Designs

There are two types of database structure that fit the project, relational and non-relational databases. Relational databases models data into one or more tables, consisting of rows and columns. Each entity will have it's own table and each row represents an instance of that object.

The a good reason behind choosing relational over non relational databases is it's ability to handle frequent short read and write operations. Additionally, the framework of choice is Laravel and its object relational model is Eloquent and both require the use of a relational database to be used effectively.

While scalability is better on non-relational databases, it is not a concern with this project as the number of properties will not exceed the limited need for it to out perform relational databases.

Entity and Relationship Diagram

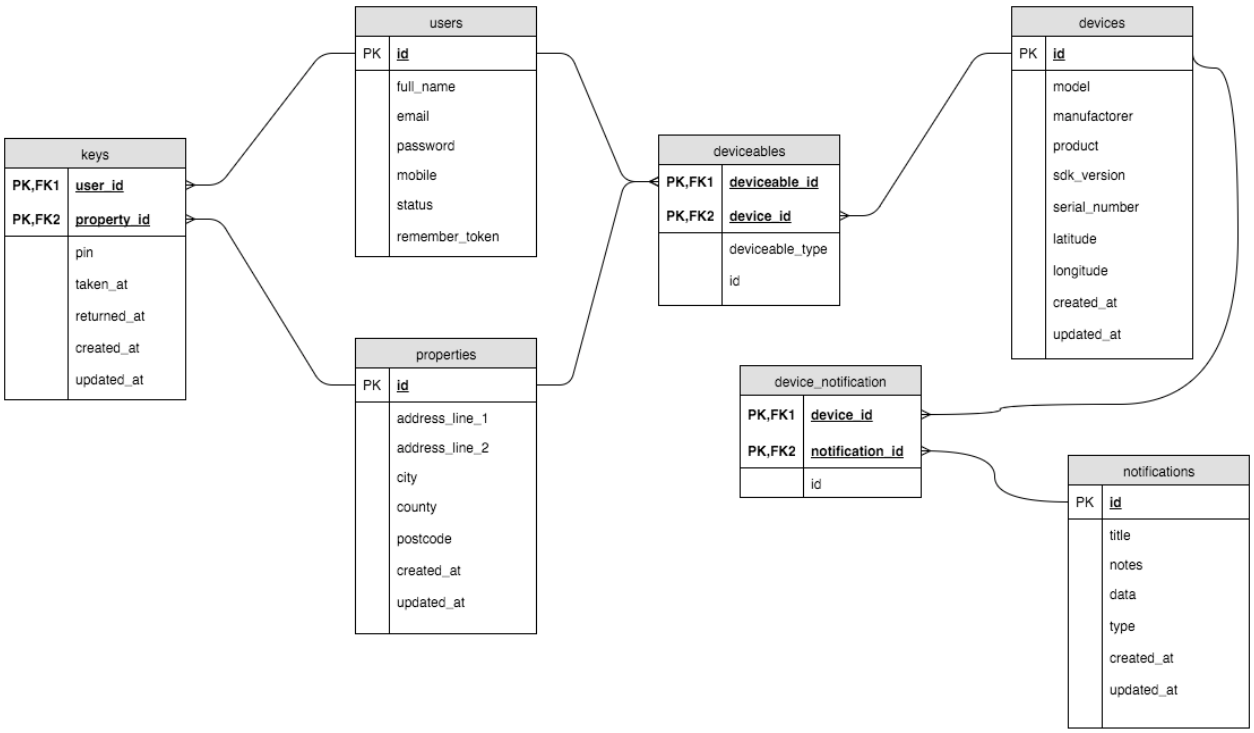


Figure 3 - Entity and relationship diagram of the notification and key tracking application

Every single model in Laravel will include a create_at and updated_at attributes. This is great for auditing data and informing the user of the time the last action had taken place.

Tables

Users - the records within this table will either be an instance of an administrator or contractor. It allows devices to be owned by a user and a contractor can have many keys as relationships. Laravel automatically hashes the password using a library called bcrypt based on the Blowfish cipher. The remember me field is for protecting against ‘cookie hijacking’. The value gets refreshed during login and logout process deeming hijacked cookies worthless, since it no longer matches.

Properties - Property management software will require a table to store property details. While the table seems very basic, it is enough. A property can have many keys in addition to multiple contractors. Multiple devices will be deployed at various locations within a property, thus a relationship between a property and device was needed.

Keys - Due to the nature to keys belonging to both a contractor and a property. This table is also known as a pivot table for properties and contractors. Describing a working relationship between properties and contractors. The taken_at and returned_at fields are for storing timestamps, when the key gets checked in and out from a cabinet. The pin field is for storing the pin codes needed to unlock the cabinet.

Devices – To attach notification to a property, a device must first be attached to a property. Once attached, multiple notifications may be assigned to the device. In order to define a device, a serial number must be present. Most if Android device will have one defined by default. Therefore, to prevent running to any problems only nexus devices will be used as they all have unique serial number. In addition to being the foundation most Android applications were built on. The latitude and longitude will be used to locate the devices. Administrators can use this alongside Google Maps API to generate a map for visual aid.

Deviceable - To prevent myself from creating a redundant device table, I decided that devices will share a polymorphic relationship with users and properties. This allows one model to be able to share a relationship with multiple models, in other words it means that a device can belong to a property and a user at the same time, but the relationships must be defined separately.

Notifications - For notifications, a title will be used to describe the notification. The type can be used to filter the list of notification i.e. alarms schedule, broken lift notice. Towards the end of the project it was decided there was not enough time to implement the filtering feature, as it is a quality of life feature and does not impact the major requirements. The data field will store the HTML text that CKEditor generates. The updated_at field will once again be used to audit the time the notification was modified.

User Interface Designs

Mock Ups

I had decided to use Balsamiq (*Balsamiq, no date*), an online wire-framing tool to plan out my user interface designs for the three applications. Balsamiq allowed me to mock up applications extremely quickly, using their drag and drop what you see is what you get editor. One downside is their lack of support for Android widgets and native user interface, however I made do with the IOS widgets. Full mock up designs can be found within Appendix 2.

Web Application

I found Balsamiq to be very useful when I want to keep the user interface consistent across the site. As each widget's look and feel provided me be with a standard design to follow. While the web application may look bland compared to the mobile applications. Users of the site will not care about the looks, only its functionality.

In order to save some time, I had only made mock-ups for pages that are unique or contains an important feature. For example, I found that the several missing pages, such as keys and notification pages were identical in design to the property's' in terms of design.

Key Tracking Application

As I had mentioned earlier, Balsamiq lacks support for Android widgets, therefore some user interfaces on the final product may differ from the initial mock-ups that I had produced.

To gather some design ideas, I had used an online visual discovery site called Pinterest (Pinterest, no date). Pinterest allowed me to search for visual trends and themes on Android, which is key when designing a professional looking user interface.

Notification Application

There was not point mocking up the notification application, as the look and look of each notification can be modified on through the web application. Additionally, the device will automatically setup itself. Therefore, it does not require any user interaction, removing any need to design the user interface.

System Requirements

Functional Requirements as User Stories

I had listed my main aims and requirements in my initial reports but back then project was at its early stages and additional requirements had requested and several had been down graded or removed.

Each of the core aims can be broken down further so that more details can be provided. As this is an agile project I will convert these requirements into user stories and provide a reasonable acceptance criteria.

Administrators

1. As an Administrator, I must be able to login to the web application so that only authorised users can gain access to the rest of the system.
   Acceptance Criteria - The administrators must be able to login using their credentials. If incorrect values had been entered they will be asked to retry.

2. As an Administrator, I must be able to logout of the web application so that administrators can switch accounts.
   Acceptance Criteria - Using the logout button this must clear the session and log the administrator off.

3. As an Administrator, I must be able to create new keys and send them to contractor providing them with access to a master key cabinet.
   Acceptance Criteria - If I am an admin, I can create new keys and assign them the contractors.

4. As an Administrator, I must be able revoke a contractor's access to a key cabinet, so that a cabinet can be kept secure.
   Acceptance Criteria - As an admin I can navigate to a contractor page, view a list of keys and remove one or several keys.

5. As an Administrator, I must be able to view the key taken and return times, so that I can contract a contractor if a key is missing.
   Acceptance Criteria - I can view a list of keys and the logged taken and return times that a contractor had made.

6. As an Administrator, I must be able to create new notification so that information can be distributed to property residents.
   Acceptance Criteria - If I am an admin I can create notifications and assign them to devices that resides within a property.

7. As an Administrator, I must be able to view the location of each terminal so that I locate the device if it needs maintaining.
   Acceptance Criteria - I can view the location of a device using Google maps.

8. As an Administrator, I only need to provide the notification with Internet connection to setup the device, so that I do not have to worry about configuring the device on deployment.
   Acceptance Criteria - The application will automatically register device details and can operate without user input.

Contractors

1. As a contractor, I must be able to login the application so that I can view a list of keys that had been assigned to me.
   Acceptance Criteria - If my credentials are valid, I should be able to see a list of keys.

2. As a contractor, I must be able to logout of the application so that only I can view the keys list.
   Acceptance Criteria - I will automatically logout of the application when it closes.

3. As a contractor, I must be able to view cabinet pin codes so that I can use it to unlock a cabinet.
   Acceptance Criteria - I can view a list of keys and their pin codes.

4. As a contractor, I must be able to log a key's taken and return times so that an administrator cannot hold me accountable if a key had be lost.
   Acceptance Criteria - I must be able to log the time taken and returned of key.

Residents

1. As a resident, I must be able to view the notifications that an administrator had made so that I am updated with the latest events.
   Acceptance Criteria - I can look at the notification device to get the latest information one fire alarm schedules and any on site maintenance.

2. As a resident, I must able to obtain the latest weather formation like the chance of precipitation and wind speed so that I can decide to bring an umbrella or not.
   Acceptance Criteria - I can view the daily weather on a notification terminal.

Non-Functional Requirements as User Stories

To design the behaviour of each system several non-functional requirements was created. Providing some constraint for the system focusing more on the quality attributes of the system, such as the performance and security of the application. As this is an agile project I will convert these requirements into user stories and provide a reasonable acceptance criteria.

Administrators

1. As an Administrator, I must be able to load all web pages under 2 seconds.

Acceptance Criteria - Making sure the time between a request and response is under the specified time.

2. As an Administrator, I must be able to operate the application on a mobile device so that I can work from away from a computer.
   Acceptance Criteria - The website can scale to fit the device screen.

3. As an Administrator, my user experience must be consistent throughout the application so that the application is easy to navigate.
   Acceptance Criteria - The user interface components like the search bar and delete buttons should be located in the same general area.

4. As an Administrator, I must be able to use the web application on different browsers.
   Acceptance Criteria - Changing browsers should not impact the features on the application, only the look and feel.

5. As an Administrator, I must be presented with a human readable message when an error has occurred.
   Acceptance Criteria - If an error occurs, it should fail gracefully and redirect to an error response page.

Contractors

1. As a contractor, I must be able to load and refresh my list of keys in less than 2 seconds.
   Acceptances Criteria - Loading and reloading speeds must be are under 2 seconds.

2. As a contractor, I must be kept informed when processing is happening in the background.
   Acceptance Criteria - A loading indicator will appear every time long processes are taking place.

3. As a contractor, I must be presented with a human readable message when error has occurred.
   Acceptance Criteria - An error message will pop up every time an error occurs, giving a reasonable explanation describing the problem.

4. As a contractor, the user interface I use should follow Android material design for so that it is pleasing to the eyes to use.
   Acceptance Criteria - The entire application will have consistent colours, text size and components.

Residents

1. As a resident, the notification displayed to be clear and easy to see.

Acceptance Criteria - The notification should scale the text size according to the device screen size. The text colour should change to complement the background.

2. As a resident, the notifications should rotate after a reasonable amount of time so that I have enough time to read everything.
   Acceptance Criteria - The notification will not switch between notifications and the weather information too fast.

Use Flow Diagram

To model the entire application I had created a user flow diagram. See Appendix 3. The diagram will describe a set of actions that a user of the system can perform and what is observable by other members of the application.

As stated previously in the requirements section, there are three actors' administrators, contractors and residents. Property managers are administrators of the system. Therefore, they are in charge of notification creation and distribution, in addition to managing keys for contractors. The contractors will only interact with the Android tracking application logging in the key taken and return times.

Use Cases

Field explanations

Title	The goal of the use case
Description	Description of the goal and context of the use case.
Actor	A person, software system that interacts with the system to achieve the specified goal
Pre-condition	Description of the system before an interaction had taken place.
Post-condition	Description of the system after all actions had taken place.
Main Success Scenario	Flow of events from the pre-condition to post-condition state.
Extensions	Extra scenario for this use case.

Use Cases

Title	Administrator can Login
Description	An admin can gain access to the rest of the application by providing their credentials.
Actor	Administrator
Pre-condition	Admin must have a register account
Post-condition	The admin is logged in and redirected to the property's listing page.
Main Success Scenario	1. Admin will need to enter their email address and password into the corresponding input boxes. 2. Admin clicks on the ‘Sign in’ button. 3. Admin get redirected to the properties dashboard page.
Extensions	2a. User incorrectly enters either the email or password. - 2a1 an error message will appear, informing the user of their error. - 2a2 Enter the correct credentials - 2a3 Clicks on ‘Sign In’ button

Title	Register new contractors
Description	Can create new contractors when given valid information.
Actor	Administrator
Pre-condition	Admin must be logged into the application.
Post-condition	A new contractor had be
Main Success Scenario	1. Administrator will need to provide the contractor’s full name. 2. Administrator will need to provide the contractor a password. 3. Administrator will need to provide the contractor an email address. 4. Administrator will need to provide the contractor a telephone number 5. Administrator will need to change the

	role to contractor. 6. Administrator clicks on submit to create a new user.
Extensions	6a. If the fields are left empty. - 6a1. An error message will pop up and tell the admin that they need to fill the missing fields. - 6a2. Admin fills in the missing fields and create a new contractor. 6b If an email address had already been used. - 6b1. An error message will pop up and inform the admin that they need to provide an alternative email address. - 6b2. Admin needs to enter a new email address or cancel the creation process.

Title	Deleting a contractor
Description	Removing a contractor record from the system
Actor	Administrator
Pre-condition	Admin must be logged in and on the contractor-listing page.
Post-condition	The deleted contractor will not be searchable
Main Success Scenario	1. Admin uses search bar to filter the table so that their desired contractors appear. 2. Admin clicks on the delete button. 3. The web page will then refresh the page and be updated with the latest list of contractors.
Extensions	None

Title	Assigning a key to a contractor.

Description	To generate a key for contractors to use to unlock a cabinet and retrieve a property master key. The newly generated key will appear within contractor's app when they refresh their key list.
Actor	Administrator
Pre-condition	Admin must be logged in and on the contractor-listing page.
Post-condition	Contractor will be able to view the key and the pin code that relates to it.
Main Success Scenario	1. Search for a contractor from the contractor table page. 2. Click on the row of the desired contractor. 3. Scroll down and click on 'Add Key' 4. Select a contractor name from the dropdown list. 5. Type in the pin code for the property. 6. Click on 'Add' to finalize the key.
Extensions	None

Title	Revoking a key from a contractor
Description	To revoke or remove an active key from a contractor. The removed key information will disappear from the contractor's key list once they refresh the list or restart the mobile app.
Actor	Administrator
Pre-condition	The administrator must be logged in and admin must be on the contractor page.
Post-condition	The contractor will no longer have access to the key and the pin code.
Main Success Scenario	1. Scroll down the active keys list 2. Type into the search bar the property name of the key. 3. Click on the 'Delete' button.
Extensions	None

Title	Assign notification(s) to a device
Description	Attaching a notification to a notification list, so that the notification app can display it to residents.
Actor	Administrator
Pre-condition	The admin must be logged in and must be on the device page.
Post-condition	The device will no longer display removed notification.
Main Success Scenario	1. Search for a device that admin wants to attach a notification to 2. Scroll down to the connected notification table 3. Click on ‘Add Notification’. 4. Select the notification from a list. 5. Click ‘Add’.
Extensions	None

Title	Removing a notification from a device
Description	Removing desired notification from the device so that it no longer displays it to residents.
Actor	Administrator
Pre-condition	Admin must be logged in and on the device page
Post-condition	The device will no longer display removed notification.
Main Success Scenario	1. Search for a device that admin wants to attach a notification to 2. Scroll down to the connected notification table 3. Search for the notification from the notification table. 4. Click ‘Delete’.
Extensions	None

Implementation

This section will outline development process that I had went through to implement the project. Including any noteworthy programming problems that I had encounters and the solutions to them. The first thing I had decided to do was setup the Laravel development environments, one locally and another on an Amazon AWS instance.

Web Application

Configuration

Since the introduction of Laravel 5, PHPDotEnv had been used to configure the environment variables. Environment variables separate sensitive data from the project's source code, as a rule of thumb, a developer should never keep their security credentials inside a Git Repo. Most of the time developers will have one configuration for their local machine and another for the production server.

Local

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=qCnQdiviL8BcjgMXaTa0tg1W302Ro2vP

DB_HOST=localhost
DB_DATABASE=dev_pms
DB_USERNAME=root
DB_PASSWORD=

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

REDIS_HOST=localhost
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Figure 4 - Print screen of the local environment parameters

For local development environment I had decided to display the error stack in the browser, as it was more convenient than searching through the Laravel error log file.

Production

```
APP_ENV=production
APP_DEBUG=false
APP_KEY=JG5h1V0Nkjpwqqrlyvvdrjm23BDvJu4e

DB_HOST=localhost
DB_DATABASE=pms
DB_USERNAME=root
DB_PASSWORD=thailand93

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

REDIS_HOST=localhost
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

Figure 5 - Print screen of the production environment parameters

For the production environment I did not want to display the error stack so I had set the *APP_DEBUG* variable to false, with different database credentials from the local version. Additionally, App_Key is required to encrypt session data and must be generated through a command line interface.

The database was also hosted on the same AWS instance in order to save on cost and like the local configuration file; I needed to specify the username and password. While at first it seems like a mistake to store the database password in plaintext, as long as the file is not placed within the public folder an outsider cannot access it, unless the entire AWS instance was compromised.

Database

The next step after setting up the environment is to store the various entities in a database, such as properties and contractors. Laravel allow developers to easily integrate databases to their web applications via the environment file. Performing queries was simplified and supports multiple database systems such as MySQL, Postgres, and SQLite and SQL server.

Migrations

Laravel uses a version control like system to manage the database. Allowing multiple developers share their database schema, following agile methodology means that the project must be able to adapt to changes and by using schemas some flexibility was added to the database. See figure 6.

```php
    public function up()
    {
        //
        Schema::table('notifications', function(Blueprint $table) {
            $table->mediumText('data')->change();
            $table->string('title');
            $table->string('notes');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
        Schema::table('notifications',  function(Blueprint $table) {
            $table->string('data')->change();
            $table->dropColumn('title');
            $table->dropColumn('notes');
        });
    }
}
```

Figure 6 - A database schema for the notifications table

A migration class will contain two methods ‘up’ and ‘down’. The up will allow new tables to be created, add new or change existing columns. The down method will need to do the reverse, conveniently for this example I wanted to change the ‘data’ column to type ‘mediumText, as it allowed more characters to be stored than what it was before. For the reverse function I changed the column type back to ‘string’ also known as ‘text’ primitively.

Routes and Middleware

For any modern web framework an application router will be needed to direct a HTTP request to an appropriate controller. The ‘app/routes.php’ is where the routes need to be defined. This was done to keep the entire route in one place and by separating the concerns future route modifications becomes easier to manage.

```php
Route::get('auth.logout', 'Auth\AuthController@getLogout');
```

With the above example the ‘/auth.logout’ requests will be handled by an authentication controller.

Request Verbs

Laravel likes to be more specific when post requests are performed. Normally, html forms only supports GET and POST request. Extra verbs like Delete and PUT verbs will not and requires spoofing by adding a hidden field named ‘Delete’. See code snippet below:

```
{{ method_field('DELETE') }}
```

Additional verbs were added so that multiple actions can be performed using the same URL.

For example:

http://192.168.0.8:8000/devices/152

The above link will load the device page with an id of 152, using the GET verb. However if a spoof POST request with a hidden field of PUT was made. The action will be to update the resource instead of loading it.

The following table lists all verbs that Laravel uses along with the action.

Verb	**Path**	**Action**	**Explanation**
GET	/resource	index	Gets the page with every resource loaded.
GET	/resource/create	create	Get the form page to create the resource
POST	/resource/	store	Request to create a resource
GET	/resource/{id}	show	Get resource page with the same id as specified in the path
GET	/resource/{id}/edit	edit	Get the form page to update the resource
PUT/ PATCH	/resource/{id}	update	Request to update a resource
DELETE	/resource/{id}	destroy	Request to delete a resource

Key Management

As specified as a project aim, property managers must be able to distribute keys to contractors. The decided solution was to have to a panel where an admin can assign keys to contractors, as well as specifying the property that key belongs to.

Assigning a key to a contractor

There are many ways to create keys, however I will be discussing the ideal method for key creation. This procedure will automatically create a relationship between a contractor and the created key, along with the property this key can be used on.

First, the administrator will need to navigate to a contractor profile page, scroll down to the active keys panel. See figure 7.

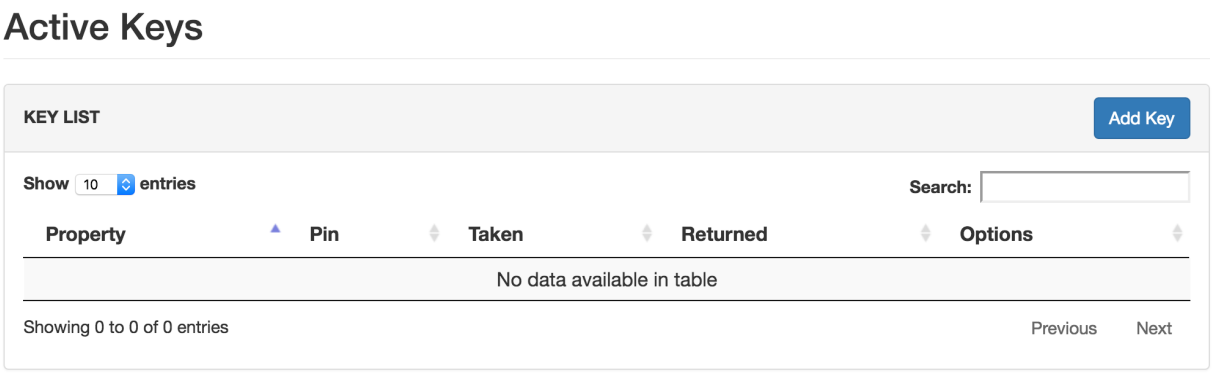


Figure 7 - An empty table of assigned keys on a contractor page

Clicking on the ‘Add Key’ button will trigger a model. Where admins can select a property from a drop down list and assign a pin code all contained within a form. See figure 8. However, from a user experience point of view this is very bad, as it is difficult for users to get a good overview of the list. If time permits I will modify this input box to use a postcode, where user can search for an address within the database.

Figure 8 - A bootstrap modal for assigning a key to a contractor

On submission, the POST request will be directed to the ‘addKey’ method within the contractor controller. See the code snippet below:

```
Route::post('contractor/{id}/addKey', 'ContractorController@addKey');
```

First the ‘addKey’ method will retrieve a property object using the submitted property id and the pin code. As it is a RESTful API, I can use the id in the URL to find the contractor. After creating a new keys and assigning the appropriate values for each field, then redirect the user back the page they were before.

```
// Add new key to contractor. Id is contractor id.
public function addKey(Request $request, $id) {
    $propertyId = $request->Input('property_id');
    $pin = $request->Input('pin');

    $contractor = User::find($id);

    $key = new Key();

    $key->pin = $pin;
    $key->user_id = $id;
    $key->property_id = $propertyId;
    $key->taken_at = "0000-00-00 00:00:00";
    $key->returned_at = "0000-00-00 00:00:00";

    $key->save();

    return Redirect::route('contractors.show', $id);
}
```

Within the key object both ‘taken_at’ and ‘returned_at’ fields was designed to accept null values in the schema. However, it is much better to assign it a default carbon value, so that I do not have to worry about null exceptions. Allowing for easier expansions in the future.

For example, the key tracking application maps this serialized key object to a java object via the use of GSON. GSON, doesn’t allow a mapping of null values if the timestamp value ‘0000-00-00 00:00:00’ was not used.

Revoking access to a key

The easiest method to remove a key is via the contractor active keys table. See figure 9. In the option panel the contractor will have the option to edit and delete a key. By deleting the key. The contractor will no longer be able to view key information on his or her device.

Figure 9 - An assigned key within a contractor's assigned key table

On submission, a POST request will be directed to the ‘removeKey’ method with the key_id as a parameter. See below:

```
Route::post('contractor/{id}/removeKey', 'ContractorController@removeKey');
```

The ‘removeKey’ method searches the database for a key matching the specified ‘key_id’ via Laravel’s eloquent model. Then proceeds to delete the record and redirect the user back the contractor page. See code snippet below.

```
public function removeKey(Request $request, $id) {
    $keyId = $request->Input('key_id');

    $key = Key::find($keyId);

    $key->delete();

    return Redirect::route('contractors.show', $id);
}
```

Notification Management

As mentioned earlier I had integrated an open source web editor called CKEditor into my application to handle the notification creation and modification. The code to integrate the editor was quite straightforward. I needed to create a webpage with a text area. Then tell CKEditor to replace the text area with the editor by using the following JavaScript code:

```
CKEDITOR.replace('notificationArea', { ...}
```

On CKEditor’s downloads page (http://ckeditor.com/download) they provide three versions of CKEditor basic, standard and full package. Essentially the versions you choose will decide the number of plugins it automatically include and ready to use straight of the box.

The developers have full control over the editor. They can add and remove the plugins that may or may not be needed. However, for an unknown reason, it does not provide a save button or plugin by default. Thus, I had to add to download and install the plugin separately.

```
// Enable save button
config.extraPlugins = 'save';
```

Creating and editing notifications

Creating a new notification is simple and straightforward because it is a **WYSIWYG** editor. They can style the text and add media to each notification in real time and the outcome

will be a document that can be displayed through a notification terminal. In terms of functionality it works like any word document.

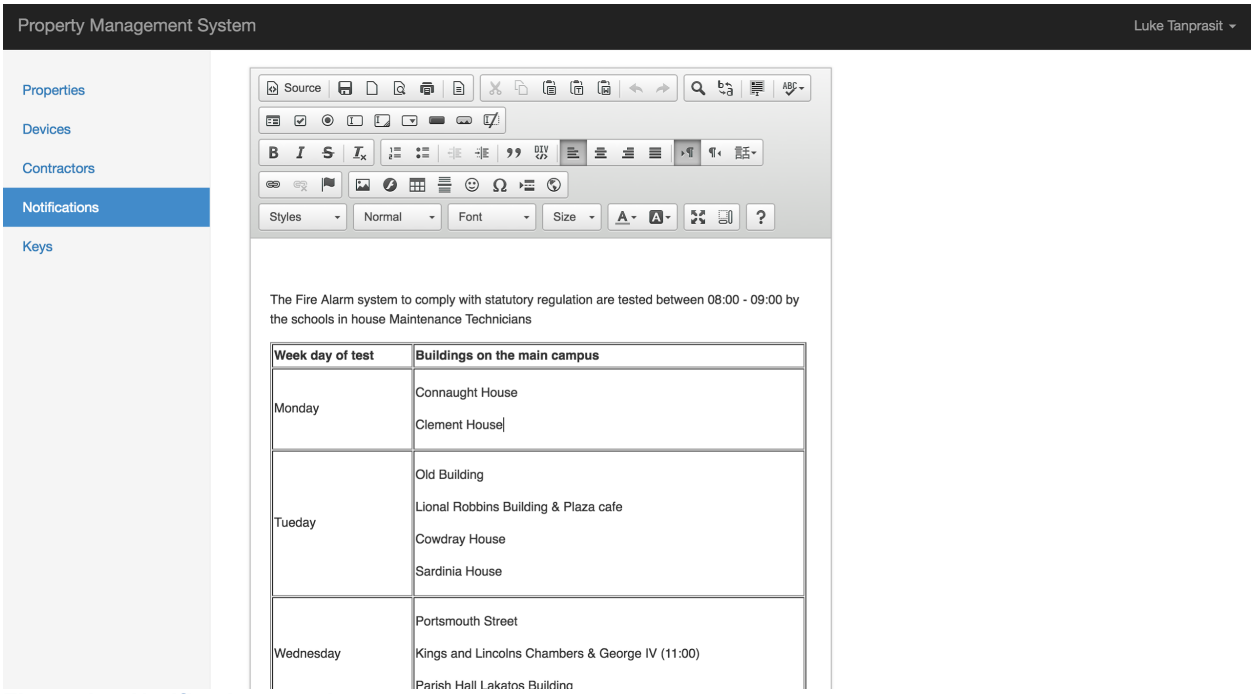


Figure 10 - Notification creation page

Currently, the notification app only supports portrait mode, so I had used an A4 paper size for the editor's width and height with 72 dpi. Ensuring that each notification will be structured roughly the same regardless of the device width and size in portrait mode. All modifications to the editor's user interface were done through 'public/lib/ckeditor/config.js' file see below.

```
// Adjust the height of the editor textarea
config.height = '842px';
config.width = '595px';
```

Saving the notification

In order to save the notification, CKEditor converts the textarea content into HTML code, before sending a request for Laravel to handle. The code to save newly created notification and edited notification are roughly the same, via AJAX.

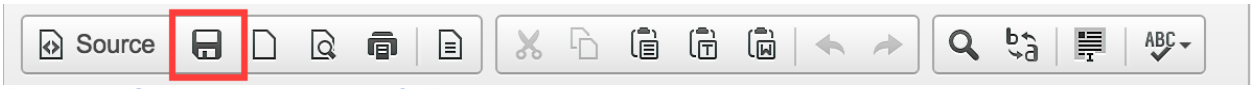


Figure 11 - Save button within the CKEditor toolbar

After adding the save plugin, I created an onclick handler for the save button. When the event triggers the HTML code that represent a notification, title, types and notes will be extracted from the textarea via JQuery. CSRF token was also injectected into AJAX in order to Laravel's security check. Then the contents of a notification will be encoded so that unwanted side effect are prevented For example, when using commas, AJAX will create parameters in a POST request. See the Ajax code below.

```
CKEDITOR.replace('notificationArea', {
on: {
    save: function(evt)
    {
        // Do something here, for example:
        var content = CKEDITOR.instances.notificationArea.getData();
        var CSRF_TOKEN = $('meta[name="csrf-token"]').attr('content');

        // Grab notification meta data.
        var type = $('#type').val();
        var title = $('#title').val();
        var notes = $('#notes').val();
        var data = encodeURIComponent(content);

        // Encode the data before ajax request. Otherwise, we will get multiple request headers.
        var dataString =
            '_token=' + CSRF_TOKEN +
            '&data=' + data +
            '&type=' + type +
            '&title=' + title +
            '&notes=' + notes;

        $.ajax({
            type: 'POST',
            url: '/notifications/',
            data: dataString
        }, "json").done(function (response) {
            window.location.replace(response.redirectURL);
        });

        // If you want to prevent the form submit (if your editor is in a <form> element), return false here
        return false;
    }
```

On arrival on the server side,

```
$this->validate($request, [
    'full_name' => 'required',
    'password' => 'required',
    'email' => 'required|unique:users',
    'mobile' =>'required',
    'status' => 'required'
]);
```

Then all input data will be used to construct a notification object and saved to a database. See code below:

```php
$title = $request->Input('title');
$notes = $request->Input('notes');
$type = $request->Input('type');
$data = $request->Input('data');

$notification = new Notification();

$notification->title = $title;
$notification->notes = $notes;
$notification->type = $type;
$notification->data = $data;

$notification->save();
```

Assigning notifications to a device

To assign a notification to a device the user needs to navigate to the devices list page. Then they will need to filter the table for the device they want to add notifications to. See figure 12.

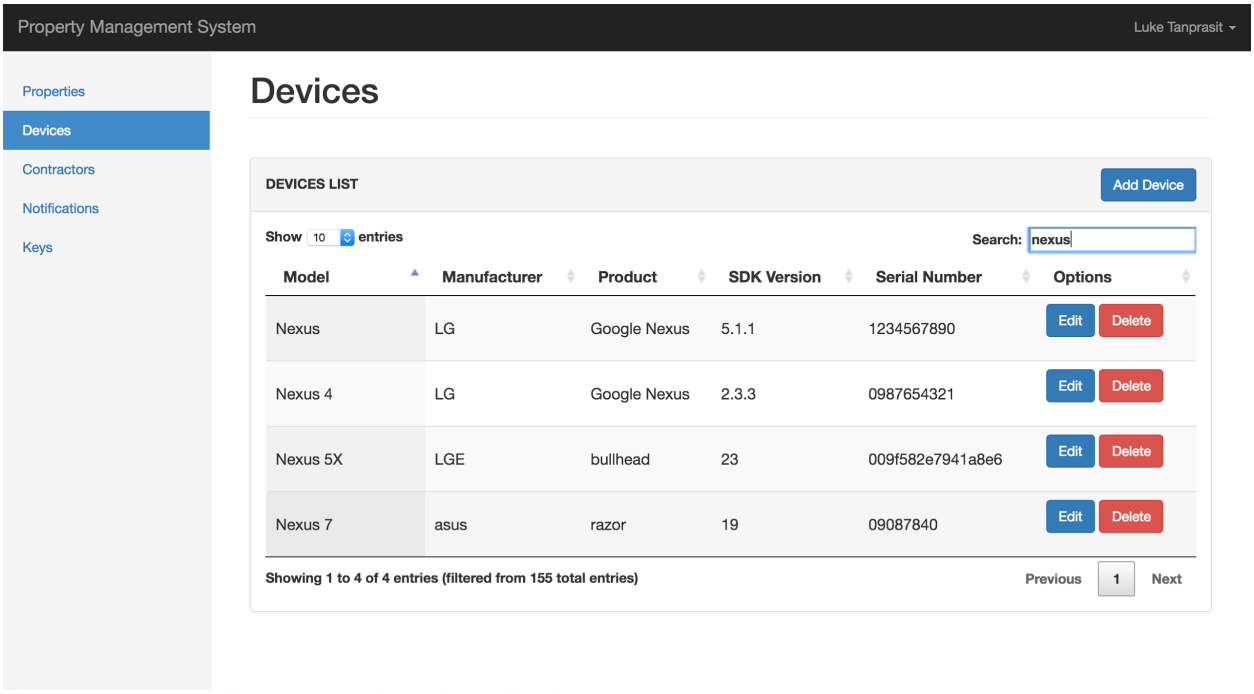


Figure 12 - Devices list page on the web application

After selecting a device and loaded the device page, scroll down to the connected notifications table, here the admin can add existing notification to the device via ‘Add Notification’ button. See figure 13.

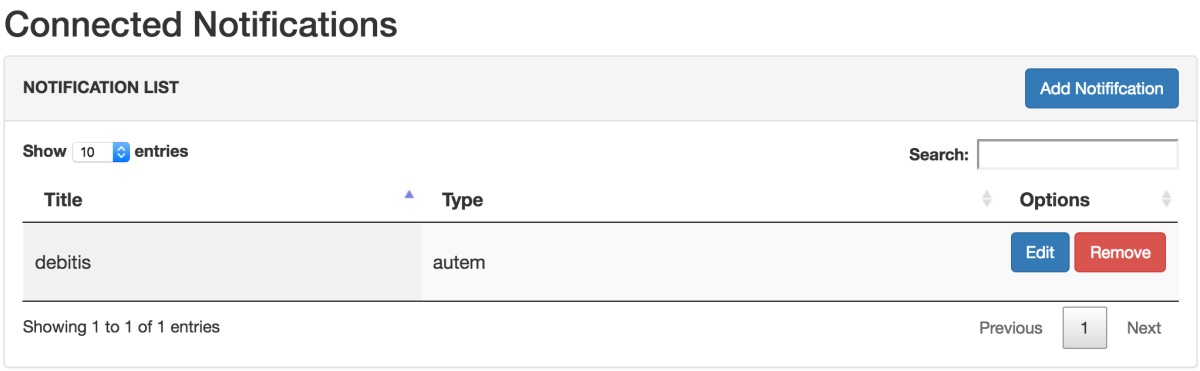


Figure 13 - Connected notifications table on a device page

Modal will appear, where admin can select a notification from the dropdown list. This can be done multiple times to chain notifications together, so that the device will rotate the notifications in order to display everything.

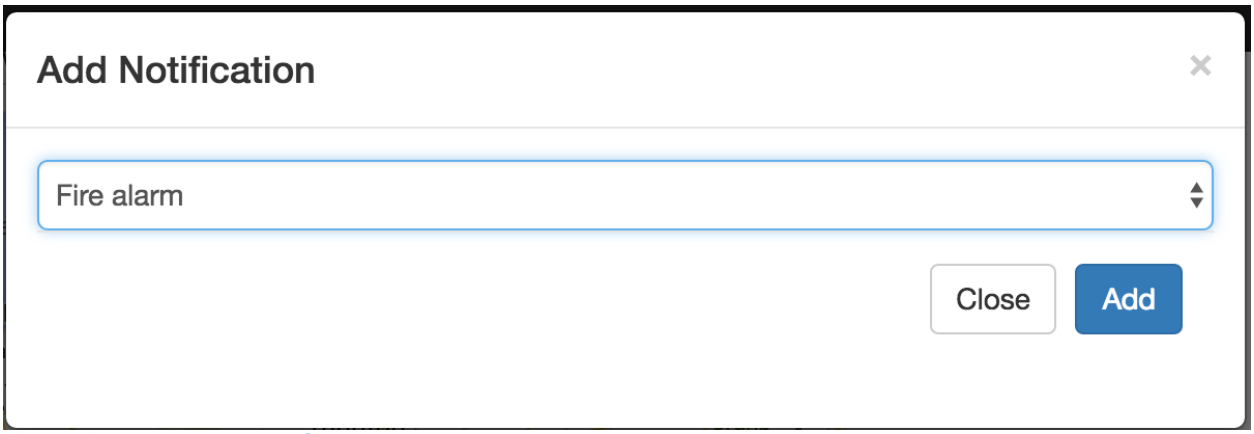


Figure 14 - Add a new notification to device modal

Removing a notification from a device

To remove a notification or notifications from a device, the administrator must navigate to each device page and scroll down to the connected notifications table. When an admin clicks on the delete button, a POST request will be sent to the server requesting the link between the device and notification to be destroyed.

```
<form method="POST" action="{{ URL::route('devices.destroy', [$device->id]) }}">
    {{ method_field('DELETE') }}
    <input type="hidden" name="_token" value="{{ csrf_token() }}">
    <a class="btn btn-primary" href="{{ URL::route('devices.edit', [$device->id] ) }}">Edit</a>
    <button class="btn btn-danger" type="submit">Delete</button>
</form>
```

Mobile Applications

Making requests

In order to communicate with the server and receive data on the Android device, a class was constructed to handle HTTP requests, both Android applications uses the same ‘RequestTask.java’ class implementation in order to save some development time. ‘RequestTask’ utilizes Android HTTP library Volley to make requests. The class does the following:

- Handles the creation of GET and POST requests.
- Handles retry policies of timeout requests.
- Handles error responses and generate human readable messages.
- Manages digest authentication routines.

For every request a new RequestTask object must be instantiated, along with a response listener and an error listener. On successful requests a JSON response will be returned, however if there was an error a VolleyError object will be returned instead.

The ‘RequestTask’ task handles GET and POST by creating two methods. StringRequest is request that expects a string response provided by Volley. For GET requests the method type does not need to be specified. However, for POST both post parameters and method type must be supplied.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST,
url, this.listener, new ErrorListener() {
```

Once a request had been created. A retry policy must also be configured in order to counter slow connections. By default the timeout value is set to 2.5 seconds and the max number retries is 1. If nothing was configured and the max number of retries had been reached, Android will throw a timeout error crashing the application.

Finally, add the request to a request queue. A Volley request queue singleton was made. My reasoning was that the application will make multiple requests and each time a request queue must be created. By not creating and destroying objects, the application will be much more efficient and the code produced becomes much cleaner.

VolleySingleton.java

```
    public static synchronized VolleySingleton getInstance(Context context) {
        if (instance == null) {
            instance = new VolleySingleton(context);
        }
        return instance;
    }

    public RequestQueue getRequestQueue() {
        if (requestQueue == null) {
            // getApplicationContext() is key, it prevents leaking the
            // Activity or BroadcastReceiver if the context was passed in accidently.
            requestQueue = Volley.newRequestQueue(context.getApplicationContext());
        }
        return requestQueue;
    }
```

Handling responses with GSON

All responses from the server will be in JSON format. Below is a typical response from the server.

A JSON response of keys

```
{
    "response": [
        {
            "id": 55,
            "takenAt": "0000-00-00 00:00:00",
            "returnedAt": "0000-00-00 00:00:00",
            "pin": 123,
            "property": {
                "id": 1,
                "addressLine1": "086 Alize Well",
                "addressLine2": "Suite 336",
                "city": "East Antoinette",
                "county": "Vermont",
                "postcode": "65358-9798"
            },
            "contractor": {
                "id": 1,
                "full_name": "Luke Tanprasit",
                "email": "tanprasitp@hotmail.co.uk",
                "mobile": "07479684526",
                "created_at": "2016-02-22 20:32:33",
                "updated_at": "2016-04-30 14:24:22"
            }
        }
    ]
}
```

The JSON was chosen as the response format because of its simplicity, ease of debugging and unlike XML uses less characters. Additionally, it is fully compatible with GSON, a Java library that can serialize and de-serialize JSON to Java objects. Combining the two removes the need for complex parsing methods.

To de-serialize a JSON to a Java object pass the JSON string and the class it maps to, to the *fromJson* method:

```
public void onResponse(String response) {
    KeyListResponse keyKeyListResponse = new Gson().fromJson(response, KeyListResponse.class);
    List<Key> keyList = keyKeyListResponse.getKeyList();
```

Both mobile applications do not need to serialize Java, however the method to do so is simple. See code below:

```
String serializedObject = new Gson().toJson(Object);
```

Conveniently, this allows any Gson serializable objects to be stored on the device, as preference manager allows strings to be stored. Once that object had been stored it can be retrieved at any point, removing the need to create a parcelable object to transfer objects between Android activities. However, it is less efficient this way.

Handling Date and Time on Android

Dealing with date and time can be difficult when time zones are involved. Splitting different functionalities into classes encapsulates problems, so that developers can focus on the problem. Additionally, I needed a clean and non-redundant way to format the date and time to fit certain situations. Thus, I utilized Joda time library to create a helper class to construct my own time object. The object can convert system's UNIX timestamp into Carbon time format, in order to be parsable server side.

Below is a method that converts Joda time into carbon time format.

```
public String getCarbonDateTime() {
   return dateTime.toString("yyyy-MM-dd HH:mm:ss");
}
```

Implemented with Agile in mind the mobile application can be configured to work at various location around the world, only requiring modification of the zone id.

A member variable of the TimeHelper.java class

```
private DateTimeZone timeZone =
DateTimeZone.forID(Constants.TIMEZONE_UK);
```

TimeHelper.java class's contructor

```
public TimeHelper() {
   this.dateTime = new DateTime();
   this.dateTime = dateTime.withZone(this.timeZone);
}
```

The following method was needed in order to convert the date and time object into a string that humans can read, for example 'Sun 01 May'. This is much more convenient than constructing the traditional *SimpleDateFormat* object, where a formatting and time zone are required every time.

```
public String getReadableDateForCard() {
   return this.dateTime.toString("EEE dd MMM");
}
```

Notification App

The notification application was designed with the sole purpose of supplying information to residents of a property, in addition to providing the local weather.

Permissions

The notification application requires Internet connection and location services. In order to use the APIs like get latitude and longitude, permissions must be specified in ‘AndroidManifest.xml’ file. Simply put a manifest file defines the structure and metadata about an application, permissions and intent filters are there to determine how they work with each other and other applications.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Device Setup

As specified as a user story, the device must be able to automatically register itself and update its location when moved from one property to another. The device must have Internet connection and the terminal owner must enable location service before the application will work.

```
setLoadingAnimation(true);

//Check if device info is exists in storage.
Context context = getApplicationContext();
SharedPreferences sharedPreferences = context.getSharedPreferences(Constants.DEVICE_DETAILS, Context.MODE_PRIVATE);

//Device object as a string.
String deviceString = sharedPreferences.getString(Constants.DEVICE_OBJECT, null);

// If not register the device to server
if (null == deviceString) {
    registerDeviceDetails();
} else {
    updateDeviceObject();
}
```

The code block above does the following, generate a loading animation, try to get the device object from local storage and determine whether or not the device needs registering or updating.

Registering the device

The first launch after installing the application, the device will register itself to the system. Using the ‘URLHelper’ class a URL is required to register the device.

```
URLBuilder urlBuilder = new URLBuilder();
String url = urlBuilder.getDeviceRegisterUrl();
```

Then the application extracts as much information as it can from the device. Such as the build, manufacturer and serial number. Location coordinates will also be extracted using the GPSTracker class.

```
private Map<String, String> generateDeviceParams() {
    Map<String, String> params = new HashMap<>();
    params.put("model", Build.MODEL);
    params.put("manufacturer", Build.MANUFACTURER);
    params.put("sdk_version", String.valueOf(Build.VERSION.SDK_INT));
    params.put("product", Build.PRODUCT);
    params.put("serial_number", Build.SERIAL);

    GPSTracker gpsTracker = new GPSTracker(getApplicationContext());
    final String latitude = String.valueOf(gpsTracker.getLatitude());
    final String longitude = String.valueOf(gpsTracker.getLongitude());
```

The latitude and longitude will be parsed into a string then compared with the string value of ‘0.0’, as it is the value that the default value that latitude and longitude returns when the location coordinates cannot be determined.

```
    if (!latitude.equals(String.valueOf(0d)) && !longitude.equals(String.valueOf(0d))) {
        params.put("latitude", latitude);
        params.put("longitude", longitude);
    }
    return params;
}
```

After the device successfully registers itself on the server. The response will then be stored locally for the rest of the application to use.

```
SharedPreferences sharedPreferences = getApplicationContext().getSharedPreferences(Co
SharedPreferences.Editor editor = sharedPreferences.edit();

editor.clear();
editor.putString(Constants.DEVICE_OBJECT, response);
editor.apply();
```

Updating the device

Updating the device information is similar to the registration routine. However, there is a check to see if previous device information exists on the device. If so, grab the device id and request a device update.

```
if (null != deviceString) {
    Device device = gson.fromJson(deviceString, Device.class);
    urlBuilder = new URLBuilder();
    url = urlBuilder.getDeviceUrl(device.getId());
} else {
```

With the above the GSON was used to convert the string device object into a use device object. Then the id number was used to construct the UPDATE request. If successful the updated device object will be replaced. This entire process will repeat every time the application restarts.

Displaying Notifications

Once a notification had been assigned to the device, the application will display the notification when the weather screen switches out. My implementation switches between weather and notifications every 20 seconds and updating occurs in between switching.

The limitation exists because of how the fragment switcher was implemented. By the time I had started building the application I was behind schedule, so decided to implement this functionality in the quickest and easiest way possible. I had decided to create a 'FragmentSwitcher' class that extends 'CountDownTimer'. 'CountDownTimer' is an abstract class that executes an event after a specified amount of time had elapsed.

.

Fragment switcher

When I extended my 'FragmentSwitcher' class from the 'CountDownTimer', I had to implement my own 'onFinish' and 'onTick' methods. I will leave the 'onTick' empty, as there is nothing I needed to do between one interval and the next. However, I implemented the fragment switching within the on 'onFinish' method because I wanted to switch fragments after the timer is up.

```
@Override
public void onFinish() {
    FragmentManager fragmentManager = this.activity.getFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

    int totalFragmentCount = FragmentList.getSize();

    switch (fragmentId) {
        case 1:
            fragmentTransaction.show(fragmentManager.findFragmentById(R.id.fragment_web));
            fragmentTransaction.hide(fragmentManager.findFragmentById(R.id.fragment_weather));
            break;
        case 2:
            fragmentTransaction.hide(fragmentManager.findFragmentById(R.id.fragment_web));
            fragmentTransaction.show(fragmentManager.findFragmentById(R.id.fragment_weather));
            break;
    }
```

The switching implementation works by assigning both notification fragment and weather fragment an id. The id number will increment after every switch and when the id number exceeds the number of fragment; the variable resets itself back to one. This implementation will simplify the appending of new fragments, like the traffic fragment for example; developers will only need to add a new case and hiding the existing fragments.

```
// If activity was already destroyed cancel the switch.
if (this.activity.isDestroyed()) {
    this.cancel();
} else {
    fragmentTransaction.commit();
    fragmentManager.executePendingTransactions();
    if (fragmentId == totalFragmentCount) {
        fragmentId = 1;
    } else {
        fragmentId++;
    }

    new FragmentSwitcher(20000, 1000, fragmentId, activity).start();
```

Displaying local weather

To supply the notification client with weather information I had built a weather 'IntentService' within the application. Simply put an 'IntentService' is a class had handles asynchronous requests, like weather update requests. All requests will be handled on a worker thread and stops itself when the job is done, which means that it will not hold up the rest of the application especially during long requests. Allowing both notifications and weather to update them simultaneously.

In order to update weather information as smoothly as possible, I had decided to update the interface when the weather fragment switches out and not viewable by the user. The logic is checks if it the fragment 'isHidden' or switched out. If so, check that the weather service is running and supply it with the URL and coordinates it needs.

```
    // When notification screen switches out. Because we are using onHiddenChanged this will
    // be triggered twice, once on appearing and another when disappearing.
    if (isHidden) {
        if (isWeatherServiceRunning()) {
            String url = new URLBuilder().getWeatherUrl(gpsTracker.getLatitude(), gpsTracker.getLongitude());
            Intent weatherIntent = new Intent(getActivity(), WeatherService.class);
            weatherIntent.putExtra(Constants.URL, url);
            getActivity().startService(weatherIntent);
        }
        this.isHidden = false;
    } else {
        this.isHidden = true;
    }
}
```

Getting the weather data follows the same procedure as a normal HTTP request, however I decided to use a callback method to separate out the code for readability purposes.

```
    getWeatherData(url, new VolleyCallback() {
        @Override
        public void onSuccess(String response) {
            Intent intentResponse = new Intent();
            intentResponse.setAction(Constants.WEATHER_RESPONSE);
            intentResponse.addCategory(Intent.CATEGORY_DEFAULT);
            intentResponse.putExtra(Constants.EXTRA_WEATHER_KEY, response);
            sendBroadcast(intentResponse);
        }
    });
```

When a response returns, the client broadcast the data back to the weather fragment. Telling it to update the user interface with the latest temperatures, wind speeds and so on. Below is the code that tells the weather broadcast listener to listen out for broadcasts with a ‘WEATHER_RESPONSE’ tag.

```
    // Register receiver
    IntentFilter intentFilter = new IntentFilter(Constants.WEATHER_RESPONSE);
    intentFilter.addCategory(Intent.CATEGORY_DEFAULT);

    this.weatherReceiver = new WeatherBroadcastReceiver();
    getActivity().registerReceiver(weatherReceiver, intentFilter);
```

Upon receiving a broadcast the receiver will extract the updated weather information and update the user interface components accordingly.

```
@Override
public void onReceive(Context context, Intent intent) {
    this.context = context;
    String weatherResult = intent.getStringExtra(Constants.EXTRA_WEATHER_KEY);
```

Key Tracking App

The key tracking application was designed to help contractor manage their keys. The application will only reveal pins when contractor activates their key, logging the time.

Contractor Login

In terms of registration every contractor will be given an account to use, with an email and password. The first thing the contractor will see is the login screen, where they can enter their credentials. See figure 14.

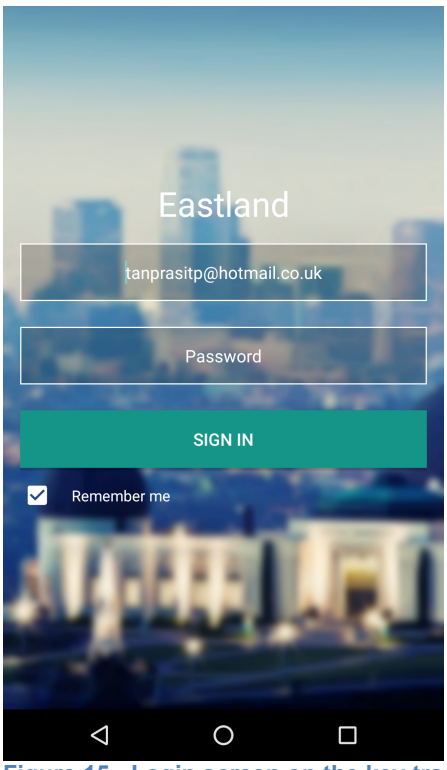


Figure 15 - Login screen on the key tracking application

I added a checkbox for remembering the username. In order to improve the user experience because the application automatically logs the user out when they close the application, forcing them to enter their email address every time. See code below:

```
private void setDefaultEmail() {
    EditText editText = (EditText) findViewById(R.id.login_username_field);

    if (this.sharedPreferences.getBoolean(Constants.REMEMBER_ME, false)) {
        editText.setText(this.sharedPreferences.getString(Constants.CURRENT_EMAIL, ""));

        CheckBox checkBox = (CheckBox) findViewById(R.id.login_checkbox);
        checkBox.setChecked(true);
    }
}
```

When the contractor entered their credentials and tapped on ‘SIGN IN’, a login request will be submitted along with their email and password. The request was made using the ‘RequestTask’ class that I had implemented.

```
final String email = usernameEditText.getText().toString();
final String password = passwordEditText.getText().toString();

if (!email.equals("") && !password.equals("")) {

    Map<String, String> params = new HashMap<>();
    params.put("email", email);
    params.put("password", password);

    String url = new URLBuilder().getLoginUrl();

    setLoadingAnimation(true);
    new RequestTask(new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Editor editor = sharedPreferences.edit();
```

A circle progress bar then appear providing feedback to the contractor, letting them know that they have submitted their request and are awaiting for a response. If the request was successful the client will save the user object that returns with the confirmation, otherwise a message will appear informing them that they had failed authentication.

Displaying Keys

Once the contractor successfully logs in. They will be presented with a key list that can either be empty or contain several cards. Each card contains information that will be useful to a contractor like the address, pin code and the key taken and returned timestamps. See figure 16.

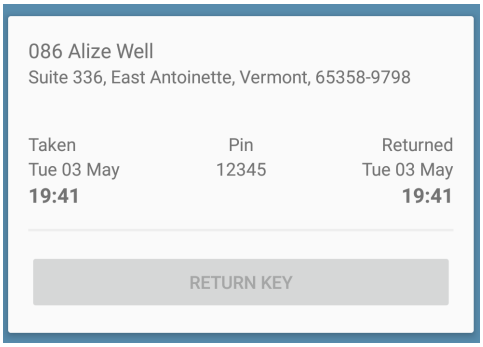


Figure 16 - A card containing an address, pin code, key taken and returned time

The key list was implemented using a widget called a ‘RecyclerView’. It is a more advanced and flexible version of a ‘ListView’. In Android a ‘ListView’ is a component that displays a list of scrollable items. A RecyclerView is Ideal for displaying medium to large datasets; users will be able to traverse through the dataset via scrolling.

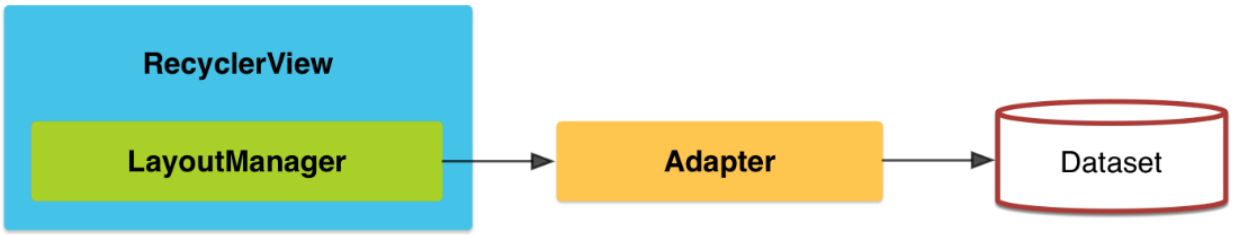


Figure 17 - Recylcer view diagram

RecyclerView

To use a RecyclerView, I added the xml code below to 'content_key.xml' file. The file will be used to render various components within the Key Activity.

```
<!-- A RecyclerView with some commonly used attributes -->
<android.support.v7.widget.RecyclerView
    android:id="@+id/activity_key_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical" />
```

Then I had attached a LayoutManager to the RecyclerView, allowing new cards to be created and attached to the RecyclerView. Setting the *hasFixedSized* to true means that when a new card gets added to the list, the contents like texts and images of that card does not force other cards within that list to re-render due to inconsistent width or height. Thus, improving the device performance, as it is expensive to re-render.

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.activity_key_recycler_view);
// use this setting to improve performance if you know that changes
// in content do not change the layout size of the RecyclerView
recyclerView.setHasFixedSize(true);

// use a linear layout manager
LayoutManager layoutManager = new LinearLayoutManager(this);
recyclerView.setLayoutManager(layoutManager);
```

Adapter

The adapter provides access to contents of a card that resides within a RecyclerView. An adapter is in charge of creating cards and updating contents of a card when it is out-dated.

Whenever a new card is created the onCreateViewHolder method is invoked. A LayouInflater is an Android component that allows new views like cards to be generated.

```
// Create new views (invoked by the layout manager)
@Override
public KeyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.key_card_view, parent, false);
    this.context = parent.getContext();
    return new KeyViewHolder(v);
}
```

Once a new card has been created. The contents like address and postcode can be injected into the view. The process is relatively simple when an adapter receives a list. The adapter indexes the items and generates the same amount of cards. Each card can then be inserted with texts, buttons or any other components.

```
// Replace the contents of a view (invoked by the layout manager)
@Override
public void onBindViewHolder(final KeyViewHolder holder, int position) {
    // - get element from your dataset at this position
    // - replace the contents of the view with that element
    final Key key = keyList.get(position);

    holder.firstAddressView.setText(key.getProperty().getAddressLine1());
    holder.secondAddressView.setText(generateSecondAddressLine(key.getProperty()));
```

Refreshing Key List

The ability to refresh the key list is a core feature of the key tracking application. Whenever an admin assigns a new key to a contractor, they must either restart the application or refresh the key list to obtain the new or updated keys.

To add the refresh feature, I added the ‘swipeRefreshLayout’ component to the ‘content_key.xml’ file. The xml snippet should wrap around the RecyclerView, while it does not affect rendering of either components; it will inform future developers that there is a refresh component that affects its child RecyclerView.

```
<!--Swipe refresh layout-->
<android.support.v4.widget.SwipeRefreshLayout
    android:id="@+id/activity_key_refresh_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- A RecyclerView with some commonly used attributes -->
    <android.support.v7.widget.RecyclerView
        android:id="@+id/activity_key_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical" />
```

Then I assigned a listener to the ‘SwipeRefreshLayout’, when a contractor makes a swipe down gesture the listener’s ‘onRefresh’ method will be invoked. On activation the contractor id will be used to send a key update request to the server.

```
swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
    @Override
    public void onRefresh() {
        int userId = contractor.getId();

        if (userId != 0) {
            String url = new URLBuilder().getKeysUrl(userId);

            new RequestTask(new Response.Listener<String>() {
```

If the request was successful, a list of keys will be returned. The response will then be serialized into a key list object. The key list adapter then uses the key list object to update the interface with the latest key information.

```
@Override
public void onResponse(String response) {
    KeyListResponse keyKeyListResponse = new Gson().fromJson(response, KeyListResponse.class);
    List<Key> keyList = keyKeyListResponse.getKeyList();

    if (keyList.size() > 0) {
        Collections.reverse(keyList);
        keyAdapter.swap(keyList);
    } else {
        keyAdapter.clear();
        runOnUiThread(() -> {
                Toast.makeText(getBaseContext(), "No keys to be displayed", Toast.LENGTH_SHORT).sh
        });
    }

    swipeRefreshLayout.setRefreshing(false);
```

Updating Key Taken and Return Times

When a new key gets assigned to a contractor, a card will be generated that card will not have a key taken or returned time. The pin code that the contractor needs to unlock a cabinet will also be replaced with the word ‘Inactivated’.

In order to retrieve the pin code, the contractor must tap on the ‘TAKE KEY’ button. On button click a user interface will update and reveal the pin code the contractor. See figure 18 and 19. I had designed the application this way, in order to force contractors to log the key taken time. When contractors are done with their key, they can tap the ‘RETURN KEY’ button to make a second request to log the key returned time.

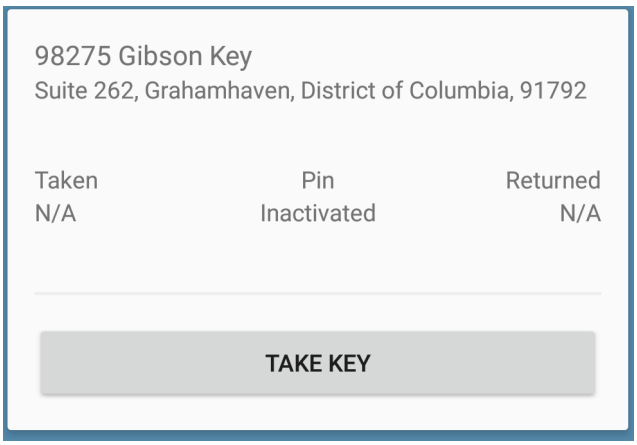


Figure 18 - An inactivated key displayed on a card

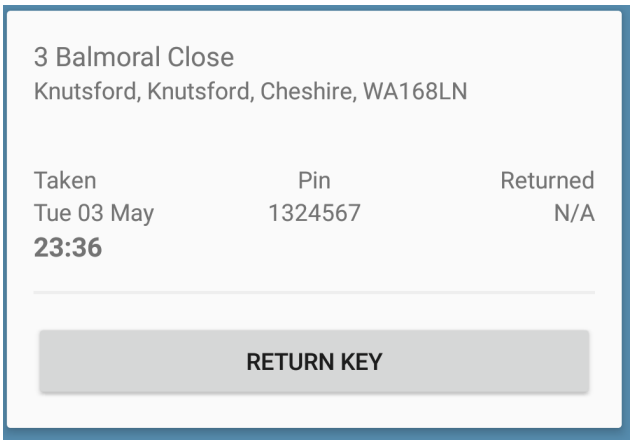


Figure 19 - An activated key displayed on a key

I experienced a nasty bug developing this feature. The issue was the key returned time was the same as the key taken time. I did not notice that the ‘TimeHelper’ object was extracted the current time of the system during card view construction and later during the on click execution.

To solve the issue I moved the logic around so that the client extracts the code just before a request submission, rather than during the card view construction, which was incorrect. See snippet below for the fix.

```
TimeHelper timeHelper = new TimeHelper();
final String currentDateTime = timeHelper.getCarbonDateTime();

// Checks if key had been taken yet, then update taken or return time accordingly.
if (!keyIsTaken) {
    postParams.put(Constants.TAKEN_PARAM, currentDateTime);
} else {
    postParams.put(Constants.RETURN_PARAM, currentDateTime);
}

if (keyIsTaken) {
    holder.keyPinView.setText(String.valueOf(key.getPin()));
}


new RequestTask((response) → {
```

Security

Authenticating Requests

One main feature that I wanted to add to my mobile application was a method to secure data transmissions. HTTPS prevents the user and the session from being exposed.

In order to decide which authentication method to use, I took into consideration the sensitivity of the data that was being submitted and the time left to implement it. For the notification system the devices will mainly be receiving non-sensitive information like a fire alarm schedules and advertisements. Thus, the method chosen does not have to be completely secure.

Digest Authentication

The first solution is to use Digest authentication method. This particular implementation prevents the shared secret from being transmitted in plaintext. Additionally, it uses nonce in order to prevent attacks from rainbow tables. Digest does not use SSL that means requests can be completed slightly faster.

There are also a few downsides from using Digest authentication. First every request that uses this method of authentication requires 2 requests, one without the credentials and one with. Second the HTTP Digest is vulnerable to man in the middle attacks, as it does not provide methods for clients to verify the server's identity.

Process:

1. Client makes a request to the server
2. Client receives a 401-authentication challenge from the server and with it a nonce.
3. The client sends back a hash response value containing username, realm, nonce, realm, URL, and a shared secret password also known as an API key. .
4. The server looks up the username and shared secret in order to carry the same hashing process.
5. Compares the client hashed response with it’s own. It is matches the client is authenticated, if not the password is wrong.

HTTP Basic Authentication using SSL

The obvious solution with securing data transmission is a priority is to use HTTPs. In addition to providing an added level of security, serving the application over HTTPS means that Google more likely to rank your site higher.

In order to use SSL, the server must be configured to use SSL there are multiple ways to obtain a valid certificate. Some will be free and others will require some sort of payment. However, the first step is to create a certificate-signing request using

OpenSSL. OpenSSL will create two files, a private key for decryption of SLL certificate and a certificate sign request.

After receiving a certificate from an issuer, it needs to be installed on a server that is hosting the application. After enabling SSL and its configuration, SSL connections will become possible.

SSL is slower than most non-SSL methods, only one API per request. The password on the server will be encrypted using bcrypt in order to make it harder for hackers to extract.

Process:

1. Client makes a request with a email and password in plaintext to the server.
2. If the credentials are valid the server will respond with the requested information or an error.

Chosen method of authentication

Given the amount of time I had left, I had decided to implement Digest authentication method, as the alternative was not viable for me to setup and use. The amount of work required in order to setup a valid certificate will undoubtedly add extra workload to an already large project.

For the notification side of the project the added level of security was an unnecessary overhead, as the information being transmitted was intended for the public eye. However, for the key tracking application it is a security flaw. I had stated prior that I ran out of time to complete the security aspect of the application. AES encryption will be used in the next iteration in order to encrypt JSON responses from the server. The response contains a lot of valuable information such the pin code that can unlock a master key cabinet and the location where the master key can be used.

Mobile Digest implementation

The below is the code that handles the authentication procedures on the Android device and can be located as a method in ‘requestTask.java’ file. The code handles the Volley

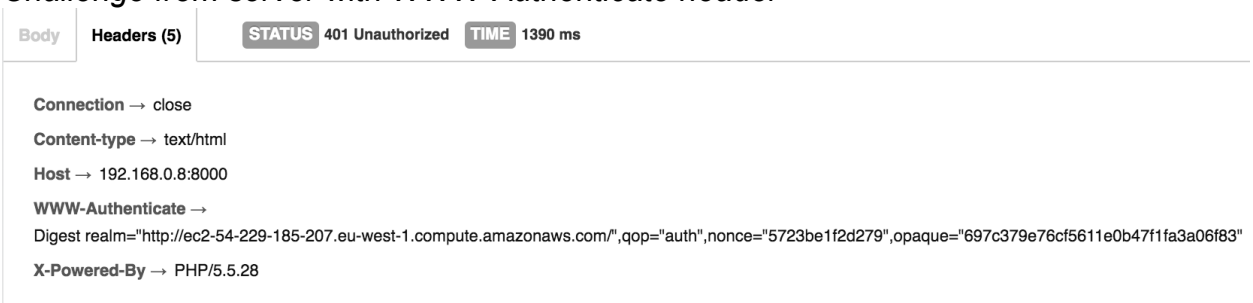
error response from the first HTTP request.

```
private void handleChallenge(VolleyError error, String url, String method) {
    Map<String, String> responseHeaders = error.networkResponse.headers;
    // Check if auth header exists
    if (responseHeaders.containsKey("WWW-Authenticate")) {
        String authHeader = responseHeaders.get("WWW-Authenticate");

        final Map<String, String> digestHeaders = parseHeader(authHeader);

        final Map<String, String> newHeaders = new HashMap<>();
        newHeaders.put("Authorization", generateAuthorizationHeader(digestHeaders, method, url));

        int methodInt = (method.equals(Constants.GET))
                ? Request.Method.GET
                : Request.Method.POST;

        StringRequest stringRequest = new StringRequest(methodInt, url, this.listener, this.errorListener) {
            @Override
            public Map<String, String> getHeaders() throws AuthFailureError {
                return newHeaders;
            }

            @Override
            protected Map<String, String> getParams() { return postParams; }
        };

        stringRequest.setRetryPolicy(new DefaultRetryPolicy(
                // 60 seconds timeout
                60000,
                DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                DefaultRetryPolicy.DEFAULT_BACKOFF_MULT
        ));

        this.queue.add(stringRequest);
    } else {
        // When user fails to enter correct credentials. WWW-Authenticate header will be missing.
        Log.e("Digest Error", "WWW-Authenticate header is missing");
        this.errorListener.onErrorResponse(error);
    }
}
```

The method checks whether or not a Digest challenge header ‘WWW-Authenticate’ exists. If the header is missing, it usually means that the user’s credentials were incorrect and that an error message had been returned.

However, if ‘‘WWW-Authenticate’’ header is accessible, it will require a method to parse its value, as it is one long string with each pair separated by a comma. The header content contains the realm, quality of protection, nonce and opaque.

Challenge from server with WWW-Authenticate header



WWW-Authenticate parse method

```
/**...*/
private HashMap<String, String> parseHeader(String headerString) {
    // Remove Auth scheme from string.
    String headerStringWithoutScheme = headerString.substring(
            headerString.indexOf(" ") + 1).trim();

    // Split each pair into an array.
    HashMap<String, String> values = new HashMap<>();
    String headerPair[] = headerStringWithoutScheme.split(",");

    // Loop through each pair and store the key and its value in a map.
    for (String pair : headerPair) {
        if (pair.contains("=")) {
            String key = pair.substring(0, pair.indexOf("="));
            String value = pair.substring(pair.indexOf("=") + 1);
            values.put(key.trim(), value.replaceAll("\"", "").trim());
        }
    }
    return values;
}
```

Now that 'digestHeader' had been parsed generateAuthorizationHeader(digestHeaders, method, url) method will generate a Digest response and appends the results to the second request.

There are two types of request the mobile application can make, either a GET or a POST request. Both requests can use this one method to handle all Digest responses, compression potentially two handlers preventing code redundancy and makes this class easier to maintain.

Volley POST request method

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, url, this.listener, new ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        NetworkResponse nr = error.networkResponse;
        if (nr != null && nr.statusCode == HttpURLConnection.HTTP_UNAUTHORIZED) {
            handleChallenge(error, url, Constants.POST);
        } else {
            errorListener.onErrorResponse(error);
        }
    }
}) {
    @Override
    protected Map<String, String> getParams() { return postParams; }
};
```

I found that the Digest handler to be problematic to implement because Volley does not provide methods to obtain the original URL or method type by default. Thus every request requires an URL and type to be passed down to the handler. Additionally, I had to manage two error listeners deciding how each error should be delegated.

For example to implement the login feature I needed to make a request in order authenticate the user. With every request an error listener must be instantiated to handle errors. When ‘RequestTask’ gets executed the first response will most like be the 401 challenges from the server. This error will be handled by a separate handler that resides in either ‘sendPostRequest’ or ‘sendGetRequest’ method. Then on the second response, if an error has occurred it will be delegated the first listener.

Server Digest implementation

Setting up the Digest authentication on the server was simple, as there are many libraries available. The package I had decided to use was called httpauth and it supports both digest and basic authentication.

To install the library the following command was executed.

```
$ composer require intervention/httpauth
```

Once installed and a package must be registered with an alias. These packages will be lazy loaded so that it doesn’t hinder the performance of the application. The term lazy loading refers to an event when a parent does not load any of its child objects.
After installation, I needed to create a filter that passes requests through the Digest authentication process. In Laravel a request filter is called a middleware; not only can it handle authentication processes, it can also be configured to audit requests.

```php
namespace App\Http\Middleware;

use Closure;
use Intervention\Httpauth\Httpauth;

class DigestMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $config = array(
            'username' => 'admin',
            'password' => '1234'
        );

        Httpauth::make($config)->secure();

        return $next($request);
    }
}
```

To enable a filter, the route configuration file must be modified. Using a group function and placing the routes inside it's closure allows a set of filters to be applied to selected routes.

```php
Route::group(['middleware' => ['digest']], function () {
    // API for retreiving notification codes
    Route::get('api/v1/notifications/{id}', 'NotificationController@apiShow');

    // API for retreiving device
    Route::get('api/v1/devices/{id}', 'DeviceController@apiShow');

    // API for registering new devices
    Route::post('api/v1/register/device', 'DeviceController@apiRegister');
```

Results and Evaluation

Testing

Initially, I had opted to create unit tests for all three applications, however I did not have time to complete them due to unforeseeable circumstance such as job interviews. I decided that it was best if I complete as much of the project as possible then performing manual unit testing.

While Laravel does offer some helper functions making tests for create, remove, update and delete operations was easy. The level of complexity increases dramatically once user interfaces are involved, as some design choices causes some presentation code to behave in an abnormal way. One example was when a label was used, as a synthetic button to submit a form. Thus, the testing framework cannot click on the label. Additionally, some implementations took far longer than anticipated including several less documented bugs.

The mobile applications was tested on two devices a Nexus 7 and a Nexus 5X. See full specifications below. The Nexus 7 was used to develop and test the notification app, as the screen size was preferable to a mobile phone's and mimics the tablet devices that will be utilized once released. The Nexus 5X was instead used to for the key tracking application.

Android Studio does offer the ability to emulate certain devices. There are drawbacks, for example not all user interaction can be replicated using a mouse or a keyboard. When testing for level of battery consumption, Android Studio also simulates the battery value meaning that it is not a true representation, as performance of field-tested device something that is hard to replicate. There are some features that Android can try and simulate the camera using the front webcam with poor results.

System Specifications

The web application was developed on a laptop and was designed to work across multiple browsers. The two mobile applications were designed and to be used on an Android devices see specifications below:

Laptop	
Operating System	OS X El Capitan 10.11 Beta (15A278b)
Model	MacBook Pro 2012
Peripherals	Laptop Built-in Screen Laptop Built-in Track pad Laptop Keyboard
Browser	Version 50.0.2661.86 (64-bit)

Mobile	
Operating System	Android 6.0.1
Model	Nexus 5X
Web Browser	Google Chrome 50.0.2661.89
Build	MHC19Q

Tablet	
Operating System	Android 4.4.2
Model	Nexus 7
Web Browser	Google Chrome 50.0.2661.89
Build	KOT49H

Unit testing – Web Application

In order to test the web application and services that it provides to the mobile application, it is critical that each route is fully functional. This, I had created several tests that I can use after completing each new implementation to ensure that everything is functioning normally.

However, I did not implement any tests for various APIs employed by the two mobile applications. Therefore, I will perform manual testing on them by firing requests and noting down the responses. See appendix 4.

While I tried my best to prevent bugs, the end product will have bugs that have yet to be discovered. As, it is extremely lengthy to test every possible variable, state and path within a system or application.

To create unit tests I had decided to use PHPUnit a PHP testing framework (https://phpunit.de/). PHPUnit can emulate HTTP requests, examine responses and fill out forms. I had created tests for every controller and the functionality of each method. Testing each method will ensure that:

- Each method works
- The response code is correct
- The response returned is as expected

Conveniently, PHPUnit is integrated into Laravel, allowing me to perform all tests using the following command within the project directory:

```
phpunit
```

Below is a screenshot from running PHPUnit. All source code will be supplied with this report.

```
Lukes-MacBook-Pro-3:property-managment-system-web luketanprasit$ phpunit
PHPUnit 4.8.23 by Sebastian Bergmann and contributors.

...............

Time: 3.65 seconds, Memory: 29.00Mb

OK (15 tests, 46 assertions)
Lukes-MacBook-Pro-3:property-managment-system-web luketanprasit$
```

User Testing – Key Tracking Application

Unlike the web application, I had opted to manually test the mobile application. In order to ease the testers in, I explained the scenario when the application will be used and that they will be carrying out a role of a contractor. All responses from testers will be noted, however I will not answer any questions or guide them through the tasks that I had set.

Functional Requirements

Task	**User**	**Responses**
Incorrectly enter the provided email address or password	User A	“It says incorrect credentials, as expected”
	User B	“A message popped up informing me that I had entered the wrong credentials”
Log in to the application	User A	“It works”
	User B	“It works like its supposed to”
Retrieve pin code to unlock a cabinet and log the time	User A	“The time was updated appropriately”
	User B	“The UI looks very good”
Refresh the key list	User A	“I don’t know how” “Pulling it down does it”
	User B	“Intuitive to pull down to refresh”
Return the key to the cabinet and log the time	User A	“Nice, the response tells me that the update was a success”
	User B	
Log out of the application	User A	“I don’t know how, There should be an option like a button ”
	User B	“I cannot see the logout button anywhere”

Non-Functional Requirements

In order to obtain feedback for non-functional requirements, I converted non-functional requirements into questions and asked them whether or not it had been accomplished, in addition to any improvements that they think should be included in the next iteration of the application.

Question	**User**	**Responses**
Was the application fast and responsive to use?	User A	“Yes, it updated fast enough.”
	User B	“Yes, very”
Did the application do a good job informing them that a long running task is in progress?	User A	“Yes, the spinner informed me that a task is running”
	User B	“Yes, the loading animation was useful”
Was the error message easy to understand	User A	“Yes, it was easy to understand
	User B	“Yes, it was simple and easy to understand”
Is the user interface easy to read and eye pleasing	User A	“App was eye pleasing the use of cards were good. The controls were bad because there are no indications.”
	User B	“Yes, more feedback to inform them that they need to pull down to refresh the key. It would be better to automatically refresh the list ”

Evaluation

To evaluate whether or not the implemented solutions were adequate, I will discuss the functional and non-functional requirements that I had set out to complete had been met. Including the results and conclusions that I had drawn from live user testing.

Functional Requirements

Administrators

As an Administrator, I must be able to login to the web application so that only authorised users can gain access to the rest of the system. I believe that I had met this requirement, as an administrator can login into the application using their account credentials. A session will also be established between the user client and server.

As an Administrator, I must be able to logout of the web application so that administrators can switch accounts. I believe that I had met this requirement, as logging out of the application is possible. The session will also be destroyed when the

As an Administrator, I must be able to create new keys and send them to contractors providing them with access to a master key cabinet. I had met this requirement, as it is possible for contractors to assign new keys to contractors and record the relationship within a database.

As an Administrator, I must be able revoke a contractor's access to a key cabinet, so that a cabinet can be kept secure. I met this requirement, as it is possible to remove a record from the pivot table that specifies a relationship between a contractor and a key.

As an Administrator, I must be able to view the key taken and return times, so that I can contract a contractor if a key is missing. I had met this requirement, as admins will have access to all contractors page, including the active keys table where the times taken and returned were displayed.

As an Administrator, I must be able to create new notification so that information can be distributed to property residents. I had met this requirement, as admins can create new notifications via the notification creation page and assign them to devices using the device page.

As an Administrator, I must be able to view the location of each terminal so that I locate the device if it needs maintaining. I had met this requirement, as admins can view the exact location of each device using the GPS coordinates extracted from individual devices.

As an Administrator, I only need to provide the notification with Internet connection to setup the device, so that I do not have to worry about configuring the device on deployment. I met this requirement by implementing an auto registration feature on the

notification application. It checks the server for its serial number; if it exists that means that it is registered. If it does not exist a new device will be created with the submitted serial number and saved in a database.

Contractors

As a contractor, I must be able to login the application so that I can view a list of keys that had been assigned to me. I met this requirement, as it is possible for contractors to sign in to the application using a valid credential.

As a contractor, I must be able to logout of the application so that only I can view the keys list. I believe that I had met this requirement, albeit partially. At the moment the user will need to close the application in order to logout. My reasoning was that I wanted to automate redundant processes like logging out, in case they the contractors forget.

As a contractor, I must be able to view cabinet pin codes so that I can use it to unlock a cabinet. I met this requirement, as contractors are able to view their assigned keys on the mobile application

As a contractor, I must be able to log a key's taken and return times so that an administrator cannot hold me accountable if a key had be lost. I met this requirement, as the mobile application can send update requests containing key taken and return times.

Residents

As a resident, I must be able to view the notifications that an administrator had made so that I am updated with the latest events. I met this requirement, as residents are able to view new and old notifications via the notification app. Administrators are able to add and remove notifications at will, allowing information to be distributed quickly and efficiently.

As a resident, I must able to obtain the latest weather formation like the chance of precipitation and wind speed so that I can decide to bring an umbrella or not. I met this requirement, as residents can view live weather data on within the application. The application updates its local weather data every 20 seconds, thus the weather data displayed will always be correct and up-to-date.

Non-Functional Requirements

Administrators

As an Administrator, I must be able to load all web pages under 2 seconds. I had met this requirement by testing the speeds for all pages and timing them and they were all under 2 seconds.

As an Administrator, I must be able to operate the application on a mobile device so that I can work from away from a computer. I did not complete this requirement, as the login page background image did not scale properly. However, the rest of the application scaled correctly on a mobile device.

As an Administrator, my user experience must be consistent throughout the application so that the application is easy to navigate. I met this requirement by using Balsamiq to first design my application ensuring consistency between pages.

As an Administrator, I must be able to use the web application on different browsers. I met this requirement by testing web application on Chrome and Safari. I found the web application to be very stable and responsive.

As an Administrator, I must be presented with a human readable message when an error has occurred. I met this requirement by building validation checks whenever, I store or update an object within the database. If an admin makes a mistake, they will be directed back to their previous page, with a custom message telling them errors and solutions to them.

Contractors

As a contractor, I must be able to load and refresh my list of keys in less than 2 seconds. I met this requirement, as I had tested the application on a standard Wi-Fi network and timed the response to confirm that it the request and the user interface updated within 2 seconds. Additionally, during live user testing and both testers agreed that the application was fast and responsive.

As a contractor, I must be kept informed when processing is happening in the background. I met this requirement by using a progress bar on long running tasks. During live user testing both testers mentioned that the application provided adequate feedback.

As a contractor, I must be presented with a human readable message when error has occurred. I met this requirement, as I had created custom error and success messages that will provide additional information to the user. During live testing both testers agreed that all messages were informative and appropriate.

As a contractor, the user interface I use should follow Android material design for so that it is pleasing to the eyes and easy to use. I believe that I did not initially meet. Testers requested basic instructions to be added to the application, like the ability to refresh the key list and some indications to where they can log out. In response, I added some instruction to the key list and added a log out button.

Residents

As a resident, the notification displayed to be clear and easy to see. I met this requirement by allowing carefully by using scaling text unit in Android.

As a resident, the notifications should rotate after a reasonable amount of time so that I have enough time to read everything. I had met this requirement by, changing the rotation speed and asking my testers whether or not enough time was given to read each notification. I found that 20 seconds was a reasonable amount of time, according to my testers

Future Work

While I believed that the client had met all core aims and requirements, there are still a lot of features left to be implement. Considering that the client is an acquaintance of mine, I will continue to develop this product in the future and hopefully deploy it in his organisation.

Web Application

As of this moment, the user interface of the application is very bland and some features are far too basic to be used effectively. For example I would like to replace the current table library with something that is more robust. Currently, when searching for a particular record the library only filter out records that does not contain the search term. This means when common words or phrases are used like first names for example, it will not filter enough of the dataset to become useful.

One key requirement I did not really fulfil was the security aspect of the application. As, I had mentioned before in the security section, I wanted to switch my requests to use HTTPS. Allowing my application to encrypt the data between the server and client, using symmetric cryptography. Encryption will prevent against man in the middle attacks as the server keeps its key secret so that the man-in-the-middle cannot use the site's real certificate. Nowadays if anyone tries to use an invalidated certificate they will be detected almost immediately.

Due to the lack of time by the end of the project, several automated tests were missing from the web application. At the moment no test exists to test that can test the API login method within the contractor controller. In the future I will try to use test-driven development as a counter measure, in order to take an even more Agile approach. It will help focus the implementation process to only produce relevant code.

Key Tracking Application

The first thing I would like to do within the key tracking application is to convert it into an IOS app. I did not suggest this earlier as a requirement because the project large enough without adding an additional application. One major factor was my lack of knowledge of IOS development, which unlike Android I have never used Swift the IOS native language of choice.

There are several features that I would like to implement but ran out of time. The first would be a prompt for users telling them to enable 3G or Wi-Fi, as the application at currently does not check for this eventuality. The next feature I would like to work on is the ability to sign up on the device itself, as currently contractor registration can only be done on the web application. As a result the feature will save administrators a lot of time, removing the need to manually enter the data himself

An additional feature that my client suggested was the ability for the contractor to call the property manager that had assigned the key to them. Essentially, adding a method of communication between contractor and the property manager like an in app messaging screen for example.

In terms of code, I would like to spend more time refactoring various components of the applications. There are some snippets of code that I could take out and replace with something better. Currently, when an error occurs an error message pops up and the message can be rather generic. I would like to replace it with a global error handler that users can interact with to submit error reports along with the error stack.

While I did some live testing personally and with a couple of tester, I would like to distribute the application to various contractors within my client's company to test whether or not they prefer the application over what exists currently. Testing the application with real intended users would mean better feedback, as the responses would be more genuine.

Another interesting observation that I had made was a lack of tutorial within the tracking application. Testers that do not use an Android on a daily basis were not aware of the pull down to refresh feature. Which lets me to believe that nothing can be too basic,

One bug issue I have with the key application is the lack of use case testing within the application. Android studio IDE provides it's own testing framework for me to use, however I did not have time, as I needed to focus on solving major issues that exists elsewhere.

An area that I had wised I worked on more, was automated testing for the notification application. Between working on three applications, I did not have enough time to create test cases for the application. User tests that I had carried out with testers should be converted into tests, as it is simple to implement.

Notification Application

One main feature that I did not have time to carry out was the ability to offer local traffic information. I had a discussion with my supervisor and we had both agreed that existing solutions were not worth the effort or costs, as most traffic APIs are paid services. Additionally, the value the feature would add to the application would be minimal compared to the implementation of the notification screen.

Personally, I would like to change how I had implemented the notification screen. At the moment each notification was displayed through a web view. Whether there is more than one notification the web view switches out the HTML code to display a different notification. I felt that this was ineffective resource wise, so I had decided to a *ViewPager* instead. Unlike the current implementation, each notification will be a unique page within a *ViewPager* and therefore would not need to switch the notifications in and out of a web view.

Conclusion

The goal of this project was to produce three applications, a web application, an Android notification app and an Android key tracking application. All three had been completed by the project deadline.

Initially I began working on establishing the core requirements of the three applications. By researching and analysing existing solutions I was able to produce both functional and non-functional requirements that were needed to progress to the implementation stage. Once I had established the actors and their roles within the system. I was able to design the entities and relationship diagrams that were used to create schemas and in turn the database.

After the design stage ended, I started working on the web application that enables key and notification management. Then I moved on to the mobile notification application. Once basic functionalities like displaying notification and weather data had been implemented, I began working on the key tracking side of the project with testing in between implementations.

All together the project was a success, the notification application can automatically register itself, communicate and pull notifications from a server. In addition to displaying local weather using the device's GPS location. On web side administrators can create, edit and delete notifications as well as assigning them to registered devices.

The key tracking application can potentially be deployed to real contractors once the web application interface for distributing keys has been streamlined. The user interface is fluid, ran smoothly and functioned as intended.

Reflection On Learning

The project gave me an opportunity to work with Android applications, something that I had enjoy working on. I was able to discover more of my strengths and weaknesses as a developer.

One major weakness that I had discovered was my judgement of project complexity. Throughout the dissertation, I had underestimated the amount of time that was needed to implement simple features. This was not an issue during my year out, as I was working within a team, therefore it was easy to move resources to speedup development when certain features are behind.

Another weakness was my loose attitude towards testing. I had started the project strong created tests for all the methods within a controller. However, as the project size grew it became exponentially difficult complete deadlines and create tests for automated testing. This highlighted the needs for me to learn how to develop using test-driven development. TDD will allow me to spend less time debugging and more time developing and solving problems.

I have previous experience working with Laravel 4.2. Yet, I had decided that it would be better for employment if I were to learn and use Laravel 5.2, the latest version. The MVC principles remained the same but several tools and libraries that I had relied on in the past had either been removed or replaced with a new version. Therefore, many commands and paradigms that I had memorised became unusable, causing slow downs, as more time was needed to read and understand the framework.

As for my strengths, I felt that I was good at picking up and using new technologies. One good example was when I had implemented the GET and POST requests on the Android applications.

Previously, I had used Apache HTTP client to make my requests. However, since Android version 6.0 the library had been removed, as it was considered to be less efficient than Volley. Then I had decided to spend a couple of hours going over web tutorials and Google documentations in order to master it. By the end of that day I was able to construct the ‘RequestTask’ class that was able to make both GET and POST requests and able to delegate errors to the right error listener.

Another strength was my ability to judge whether I should be implementing my own solutions or use existing implementations. For example I implemented the digest authentication feature on the server, I knew that there were several libraries available that can perform the digest routine. However, on the Android Digest authentication libraries exist but most implementation looked very ad hock, lacking in structure and documentation.

I found that during the designing phase that it was a good idea to take a look at existing mobile applications and the user interface. Analysing the good and bad allowed me to construct a better-looking application than I would have otherwise.

Several attempts were made to have meetings with my client, however due to scheduling conflicts and job interviews I found it difficult to do so and obtain feedback from him. However, being able to discussing requirements initially allowed me to pin point exactly what he wanted out of the application. In addition, I was able to discuss in more detail the scenarios where the application will be utilize.

Project Management

Weekly Meetings

I met with my supervisor on a weekly basis, where we had discussed the progress that I had made that week, issues that I had encountered and set of goals. He would often provide me suggestions and guidance that I needed, so that I do not fall behind schedule.

Agile Methodology

Having experienced the benefits of both Agile and Waterfall, I had decided to use them both. The sequential design process was during the requirements gathering and design stage allowing me to carefully design code can be reused between applications, however once I reached the implementation stage I switched to full Agile. I needed to product three separate applications in the shortest amount of time possible, so when the deadline arrives I wanted to have a basic system that I can show to my supervisor and moderator.

I believe that my decision to use Agile can be justified. Once week 9 arrived, I was able to demonstrate the notification application and by week 10 I had a working key tracking application to show and test. The remaining time I had left was used on writing this report and tests various aspects of the system.

References

Freakley, S. (2015) Businesses must embrace the digital world. Available at: http://www.telegraph.co.uk/finance/comment/12022636/Businesses-must-embrace-the-digital-world-to-compete-or-be-left-behind.html (Accessed: 22 April 2016).

Ltd, A.O. (2011) Property management software - Arthur. Available at: http://www.arthuronline.co.uk/ (Accessed: 22 April 2016).

Buildium (2016) Property management software. Available at: https://www.buildium.com/ (Accessed: 23 April 2016).

TrackR (2016) Lost it? Find it. TrackR helps you find lost items in seconds using your iPhone or Android. Available at: https://www.thetrackr.com/ (Accessed: 23 April 2016)

Keytracker, T. (2016) Home page - Keytracker - secure key and asset management. Available at: https://www.keytracker.com/ (Accessed: 24 April 2016).

Pinterest (no date) Available at: https://uk.pinterest.com/ (Accessed: 6 May 2016).

Balsamiq. Rapid, effective and fun wireframing software (no date) Available at: https://balsamiq.com/ (Accessed: 6 May 2016).

Appendix

All appendices can be found within the attached zip folder and will be ordered the same as the following list:

1. Background Research Documents

2. Balsamiq Mock-ups

3. User Flow Diagram

4. Manual Testing Documents