

An Application that promotes safe behaviour Through Nudge-based Notifications

Student Name: Christopher Tregear

Student Number: 160516317

Supervisor: Professor Aad van Moorsel

Deadline: 25/08/2017

Word Count: 15656

Abstract

Recent increases in violent crime across England and Wales is worrying when considered in the context of increased budget cuts on police forces around the country. In light of this mounting pressure on police forces, it is increasingly important to look at ways in which individuals can protect themselves against becoming a victim of crime. This paper will focus on an approach to promoting behaviour that is conducive to remaining safe while out in public.

Acknowledgements

I would first like to express my gratitude to my supervisor professor Aad van Moorsel for his guidance during the course of the dissertation. I would also like to thank my girlfriend Celine Wang whose love and support over the years has not only helped through poor health, but has been a source of encouragement during all stages of my degree.

Table of Contents

1.0	Introduction.....	8
	1.1 Aims and Objectives.....	9
	1.2 Scope of the Project.....	9
	1.3 Project Structure.....	10
2.0	Background and Research.....	11
	2.1 Choice Architecture and Nudges.....	11
	2.2 Key Technologies.....	12
	2.3 The SafeZone App.....	14
	2.4 Summary to Background and Research.....	15
3.0	Requirements Elicitation and Analysis.....	16
	3.1 Requirements Analysis.....	17
	3.2 Functional Requirements.....	18
	3.3 Non-Functional Requirements.....	20
	3.4 Risk.....	21
	3.5 Conclusion to Requirements Elicitation and Analysis.....	25
4.0	System Design.....	26
	4.1 Agile Development.....	26
	4.2 Sequence Diagram.....	27
	4.3 Use Case Diagram.....	28
	4.4 Wire Frame Diagram.....	30
	4.5 Designing the Nudge-based Notifications.....	31
	4.6 Summary of the System Design.....	33
5.0	System Implementation.....	34
	5.1 Orange3- Data Mining Package.....	34
	5.2 The K-means Algorithm.....	35
	5.3 Calculating New Centroid Values.....	38

	5.4 Building the Android Application.....	40
	5.5 Implementation Summary.....	45
6.0	Testing.....	46
	6.1 Black and White Box Testing.....	46
	6.2 System Integration Testing.....	46
	6.3 Functional Requirement Testing.....	47
	6.4 Acceptance Testing.....	49
	6.5 Testing the Geofences.....	51
7.0	Evaluation and Results	52
	7.1 Meeting Functional Requirements.....	53
	7.2 Meeting Non-Functional Requirements.....	54
	7.3 Focus Group for Program Evaluation.....	54
	7.4 Results.....	55
	7.5 Results and Summary.....	56
8.0	Conclusion.....	58
9.0	References.....	60
	Appendix (A) Questions used to guide the focus group system evaluation.....	63

List of Figures

Figure 4.1: sequence diagram.....	27
Figure 4.2: use case diagram showing users' interactions with the system.....	28
Figure 4.3: wireframe diagram showing menu screen.....	30
Figure 4.4: wireframe diagram showing notification screen.....	30
Figure 4.5: wireframe diagram showing nudge-based notification.....	30
Figure 4.6: wireframe diagram showing nudge-based notification.....	30
Figure 4.7: wireframe diagram showing nudge-based notification.....	30
Figure 4.8: wireframe diagram showing nudge-based notification.....	30
Figure 4.9: wireframe diagram showing nudge-based notification.....	31

Figure 4.10: wireframe diagram showing nudge-based notification.....	31
Figure 4.11: wireframe diagram showing nudge-based notification.....	31
Figure 5.1: orange3 setup for the processing of the project's data.....	34
Figure 5.2: uploading data to orange interface.....	35
Figure 5.3: k-means algorithm.....	36
Figure 5.4: configuring k-means.....	36
Figure 5.5: visualised representation of the processed K-means output.....	37
Figure 5.6: preview of processed data.....	38
Figure 5.7: formula used to calculate the arithmetic mean.....	38
Figure 5.8: centroid data	39
Figure 5.9: location tracking permissions.....	40
Figure 5.10: play-services-location dependency.....	40
Figure 5.11: notification builder.....	40
Figure 5.12: notification issued.....	41
Figure 5.13: list of images in drawable.....	41
Figure 5.14: getRandom() method.....	42
Figure 5.15: getMessage() method.....	42
Figure 5.16: splash screen.....	43
Figure 5.17: multi-threaded program for splashscreen.....	43
Figure 5.18: menu scroll screen.....	44
Figure 5.19: string file.....	44
Figure 5.20: setting up scroll screen.....	45
Figure 6.1: state Transition Diagram of the new safety application.....	47
Figure 6.2: setting mock location.....	50
Figure 6.3: receiving notification.....	50
Figure 7.1: gantt chart.....	52

List of Tables

Table 3.1: Shows the results of the focus group.....	17
Table 3.2: Functional requirements.....	19
Table 3.3: Non-functional requirements.....	21
Table 3.4: lists the potential risks to the software application project.....	23
Table 3.5: categorizes the risks in order of severity as either low, medium, or high.....	24
Table 3.6: offers a list of risk mitigation strategies.....	25
Table 6.1: results of the functional requirements testing.....	47

1.0 Introduction

As the exponential growth of technology continues to touch every facet of our lives in deeper and ever meaningful ways, we have begun to look to technology to solve some of societies' most profound problems. From healthcare to transport, computer technology has become an inescapable fact of life in the 21st Century. The application of technology to social issues such as crime is just another example of how technology can help to improve quality of life and well-being. Byrne and Marx [9] distinguish between two different categories of technological innovations – those that are information based and those that are material based. These can be referred to as soft technologies and hard technologies respectively. Both soft and hard technologies are increasingly being applied to crime prevention and police, however, they are being applied in different capacities. The police tend to use hard technologies for increased police protection such as vests, cars, riot gear, guns and helmets while the same category of technology is increasingly used in crime prevention in the form of CCTV and extra street lighting [9]. Soft technologies, however, have increased the use of technology in the police through crime mapping, information sharing between departments, and new technologies to monitor communication. In crime prevention, soft technologies are being used for crime threat assessment instruments, profiling potential offenders and facial recognition software [9].

Another area of research into the application of soft technology to crime prevention include the study of how information-based technologies can reduce the fear of crime (FoC). FoC can be summed up as the fear of becoming a victim of crime as opposed to the actual probability of becoming a victim of crime [17]. There are conflicting theories in the literature regarding the causes behind increased FoC in the public. According to Baba and Austin [4] earlier theories have accounted for FoC as being directly related to the experience of criminal victimization while other theories claim that FoC has very little to do about past victimization and instead has more to do with attitudes about what is going on in the community [4]. The impact of FoC in individuals can have a detrimental effect on an individual's well-being and can lead to '...heightened interpersonal trust, withdrawal of support from the systems of formal authority devised to control crime, and decreased levels of social interaction' [4]. Reducing FoC is one of the state's primary concerns [4].

Recent literature has focused on ways in which soft technologies can be applied to Neighbourhood Watch Schemes to increase participation and thus reduce FoC. Such a system was proposed by Arief et al. [2]. in their paper, 'Towards the implementation of an internet-based neighbourhood watch scheme' it is pointed out that 'reduced participation can lead to an increase in fear of crime' and propose the use of internet technology to expanding the reach of the Neighbourhood Watch Scheme to those who might be most vulnerable [2]. With continued cuts to the police service across the UK [3] which has been closely accompanied by a 24% increase in violent crime [29] can the further application of soft technologies to crime prevention be used to confront increasing levels of violent crime? The focus of this paper will be on the application of soft-based technologies to the building of an android software application that aims to change the behaviour of its users so that they behave more safely without increasing FoC.

1.1 Aims and Objectives

This project seeks to successfully create an android software application that will encourage its users to behave more safely while in public and thus help to reduce the chances of becoming a victim of crime. This will be achieved through notifications to a user's phone when they enter areas that have higher recorded rates of crime. It is also important to note that this application seeks to avoid increasing unnecessary fear of crime, and so has chosen to adopt a strategy of nudge-based notifications that will encourage precautionary behaviour instead.

The following objectives have been carefully selected to provide the project with a logical sequence of tasks that will facilitate the successful progression of the solution from beginning to end.

- Research will be carried out into android and its development environment as well as the implementation of geolocation and geofencing.
- Research into how choice architecture and nudges can be used to influence behaviour and how similar techniques might be applied to a safety application.
- Host a discussion group to aid in requirement elicitation.
- Design and implement an android application that successfully utilises and processes data from location and crime in the Northumbria area.
- Look at machine learning algorithms that might help in the processing of crime data.
- Test and Evaluate finished system
- Make improvements based on final tests and evaluation.

1.2 Project Scope

- The android application will aim to support the most popular android devices above android version 4.1 Jelly Bean.
- Organise and group crime data to create geofences
- Effectively track a user's location and set triggers every time the user enters or leaves a geofence.
- Provide nudge-based suggestions on how to change behaviour to avoid becoming a victim of crime.
- Once a notification is selected, the user will be guided to the main interface screen where the notification will be displayed with either a picture, a written message, or both.
- All nudges can also be viewed on the main interface menu in the form of a list.
- Due to time constraints, this application will offer only a simple interface that can be easily navigated.

1.3 Project Structure

This project will be made up of nine chapters that will include the introduction, background research, requirements & analysis, system design, system implementation, testing & results, evaluation, conclusion, and bibliography respectively. Chapter 1 will give an introduction into crime in England and Wales, fear of crime (FoC) and the proliferation of software applications as a potential tool against the fight against crime. The first chapter will also cover the aims and objectives of the project. Chapter 2 will focus on the background research regarding choice architecture, nudges, and similar applications. Additionally, this chapter will also explore key technologies used in the project such as geolocation services, geofences, android studio, and data mining techniques. Chapter 3 will discuss the process of requirements gathering and will formally list both functional and non-functional requirements. Chapter 4 will cover the actual design of the system in the form of written descriptions and diagrams. Chapters 5, 6, 7, and 8 will detail the development processes involved in the implementation of the system, testing of the end software application, evaluation, and conclusion. The final chapter will be a bibliography and will include a list of reference material consulted for the successful completion of the project.

2.0 Background Research

This chapter will offer a brief overview of some of the methods and technologies that will be implemented in the building of the main software application. The chapter will then conclude with a review of a similar application that is already on the market, the SafetyZone App.

2.1 Choice Architecture and Nudges

One might be forgiven for the over-confidence placed in spontaneous decisions made when little to no information is available. Indeed, some of the wider literature suggests people have a natural ability to make accurate and reliable spontaneous judgments without full concentration and the need to consult large volumes of data. Malcolm Gladwell [18] refers to this miraculous ability of our intuition as ‘Thin-Slicing’. This, he goes on to define as the ‘...ability of our unconscious to find patterns in situations and behaviour based on very narrow slices of experience’ and the ‘...part of what makes the unconscious so dazzling.[18]. There is a prevailing belief, however, that individuals, in fact, make very bad decisions without having complete information and the full concentration that might otherwise be available to them. Kahneman [22] is one such scholar who advocates reason and rationality over the power of intuition. Kahneman distinguishes between two systems of the mind. System 1, he describes as the automatic system that with little effort is quick to make decisions. System 1 can be compared to what we might often refer to as intuition. System 2 or the reflective system, on the other hand, Kahneman explains, is associated with regulating some of the thoughts that are manifested from system 1 and is involved in rational thinking and concentration. When system 1 is permitted to execute unchecked as it often does, it becomes susceptible to bias and is prone to mistakes [22].

It is possible to take advantage of system 1 and the mind’s natural disposition to take shortcuts to help make better decisions. Thaler and Sunstein [40] describe the ways in which it is possible to improve people’s lives and solve some of society’s biggest problems while at the same time maintaining their freedom of choice. According to Thaler and Sunstein [40], there is no such thing as neutral design and it is in fact the choice architect’s role to organize the context in which decisions are made. This can be achieved by way of what Thaler and Sunstein call a ‘nudge’. More specifically, a nudge is ‘...any aspect of the choice architecture that alters people’s behaviour in a predictable way without forbidding any options or significantly changing their economic incentives.’ [40]. It is also important to keep in mind that a qualified nudge is one in which it must be cheap and easy to avoid. By examining some of the weaknesses of system 1 or the automatic system, it is possible to use nudges to direct decisions and behaviour. Some weaknesses include the availability heuristic, anchoring & adjustment, framing, priming, and the status quo bias. The availability heuristic states that people will be influenced by incidents that are easily recalled. In a study that involved applying behavioural insights to organ donation it was found that by taking advantage of the availability heuristic it was possible to significantly increase the number of people registering for donation. This was done by providing information on the large number of people who had already joined the organ donor register. This proved to inflate the perceptions of those visiting the site about what other people do, giving them an exaggerated sense of the number of people registering as an organ

donor which in turn increased the chance of the visitor of the site registering as an organ donor themselves [10].

Our automatic systems can be exploited through anchoring and adjustment. This often occurs when a number is given and future estimations are then made on adjustments of that number. This is seen clearly in studies of how to increase charitable giving. One such study shows that donors can be influenced to increase their donations by the way possible donation amounts are listed. This is achieved by anchoring the donor to higher amounts [10]. The same study also demonstrates the effective application of the status quo bias. This method utilises default options as a powerful nudge to increase donations over a long period of time. In such a scenario donors are asked whether they would prefer to join a scheme where donations are increased year by year but with the choice of opting out at any time. It is found that these donors will donate three times more than expected over their lifetime [39]. Another way in which anchoring and adjustment may be used to influence behaviour and choices is by changing the order in which questions or statements are presented. This is achieved in the organ study by way of asking a question which serves to anchor the potential donor to a certain way of thinking: “If you needed an organ transplant would you have one? If so please help others” [10].

The automatic system is also vulnerable to priming which can serve to change a person’s behaviour. Priming can come in the form of asking someone what they will do in the future thus making them more likely to carry out the stated action. This is often more accurately referred to as the mere-measurement effect. Another form of priming can be used through the presentation of a stimulus that can later have a direct influence on a person’s choice [40]. In the organ donor study, researchers found that by adding a visual cue of a crowd of people to a piece of text they were also able to significantly increase the number of people signing up to organ donation.

2.2 Key Technologies

When deciding the principle language to be used in the writing of the application, it was important to choose a language that the developer was most comfortable and proficient in. Due to extensive experience and instruction received in java, this became the obvious language of choice. Additionally, java is a powerful programming language that is used on a wide range of devices and operating systems.

2.2.1 Android Studio IDE

Upon deciding on java as the code of choice for this project, it was then necessary to choose a development environment for java that would allow for the development of a mobile application.

After some research, it was decided that Android Studio IDE would suit the project’s needs. Some of the benefits of using android studio include:

- Uses java as the core development language.

- Android Studio is a fast and efficient IDE that is designed specifically for the development of android applications.
- It is quick and simple to start a new android project for different types of android applications.
- Android Studio allows you to use a live-layout feature which allows the developer to see what the application will look like on their chosen android device. This is achieved through real time app layout rendering. This was very helpful in the development of the new application.
- Android Studio also utilises modules to manage code. Each code module also possess their own Gradle build files which allow them to define their dependencies.

2.22 Geolocation

The increasing pervasiveness of context-aware mobile applications are becoming ever reliant on geolocation technologies and services. Geolocation services allow for the determining of a device's precise location based on longitudinal and latitudinal geographical coordinates. The wireless geolocation system is usually comprised of three key components: a location sensing device, a positioning algorithm, and a display system [35]. Although there are several different location technologies that may be used for geolocation this paper will briefly focus on two of the most relevant technologies used in the new application and then discuss some of the advantages and disadvantages of each.

2.2.3 Global Positioning System (GPS)

Global Positioning System or GPS is a popular location technology and often regarded as one of the most accurate location technologies available [35]. GPS takes advantage of satellites containing highly accurate clocks that are accurate to within 3 nanoseconds. This accuracy is essential as the receiver is required to calculate exactly how long it takes for a signal to travel between GPS satellites [35]. However, there are disadvantages to using GPS location technology which include the necessary use of a GPS chip that uses large amounts of power and therefore quickly depletes mobile phone batteries. Also, the accuracy of GPS largely depends on the quality of the GPS signal which can be negatively affected by dense cloud cover and tall buildings.

2.2.4 Network Location Provider Services (CellID lookup/ WiFi MACID lookup)

This strategy uses cell towers and WiFi access points to determine location. The advantage of this technique is that no location chip is required and power consumption is minimal. However, unfortunately, unless there are many WiFi access points, there is a marked drop in accuracy (Android Developers, n.d.).

2.2.5 Geofences

Geofences are virtual perimeters that surround a geographical area using point of interest (POI) data. Examples of POI data might include parks, universities or other places of special interest. Devices with GPS capability may be used in conjunction with geofences so that if the device enters or exits these virtual boundaries then a trigger is created. The new application will use longitude and latitude together with a radius so that a boundary can be created around the point feature.

2.2.6 Data Mining

Data mining is the process of discerning patterns from large datasets with the view of gaining new knowledge and insights into data that might aid in making better, more accurate predictions [45]. Data mining processes are often automatic or semi-automatic leading to some sort of advantage. With the rapid accumulation of data across a vast number of fields, data mining has been applied in areas such as healthcare to predict the volume of different categories of patients, fraud detection, understanding purchase behaviour, and in customer relationship management just to name a few. With the exponential growth of data, new technologies and techniques are needed to manage and analyse these vast datasets. During this project, data mining techniques were explored to find an appropriate solution to extrapolate patterns from large crime datasets. As a result, various cluster algorithms were explored as a potential solution to grouping the crime data.

2.3 The SafeZone App

The SafeZone App is a personal safety application designed and built by an Australian company CriticalArc Pty Ltd. that allows its users to be connected directly to the university security team when in need of help while on campus. The application is available free of charge to both students and staff from a range of universities including the University of Newcastle Upon Tyne and is available for download through both the iOS App Store (apple) and Google Play (android).

The SafeZone App offers a comprehensive array of features. If users feel unsafe they have the option of manually checking-in to certain locations around the university campus. The application also allows for high-speed push messaging so that the emergency management team can inform users of emergency or critical incidents. However, the strengths of this application mainly lie in the capability of directly connecting its users to the emergency management team while on university campus or nearby student accommodation. This is achieved through three separate clearly defined options: emergency alert, first aid alert, and help call. Once an option has been selected, the emergency management team are able to track the user's location on campus and provide rapid response. The application also provides options for the user to view their location and the safety zones.

In order for the emergency management response team to effectively respond to emergency alerts the application tracks the user's location using a combination of cell-network, Wi-Fi and GPS positioning technologies. Safe zones are defined strictly with the use of Apple and Google services and the implementation of geofencing technologies. By using geofences users can be automatically signed out of safe zones once they leave the area.

Despite the obvious success of this application, unfortunately it's features and functionality are limited by areas that the application defines as safe zones. Once a user leaves a given safe zone area, the application is left largely redundant. It is easy to see that emergency response would be very effective on smaller campuses, however, on larger campuses with only one centralised security office it might be argued that in certain situations it would be equally or more effective to call emergency services directly as opposed to going through the security team. The SafeZone application tends to focus largely on response to emergency incidents and seems to provide little in the way of preventative measures. It is hoped that my application might add value to the SafeZone app by increasing focus on ways in which it could further prevent its students from becoming victims of crime.

2.4 Summary of the Background Research

This chapter started by first introducing choice architecture and nudges; describing various techniques used by choice architects to help people to make better decisions. Some of these nudges included the use of the availability heuristic, mere-measurement effect, priming, status quo bias, and framing. Some of these techniques will be utilised in designing messages for the new safety application. Research was carried out into relevant key technologies before carrying out a review of The SafeZone App. It was concluded that The SafeZone App was a very useful application with good safety features, however, it did have its limitations which included lack of functions outside of university safe zones and provided little in the way of preventative measures. The new application can add value to existing applications by offering nudge-based notifications that will help to prevent its users from becoming a victim of crime. The following chapter will look at the process of requirements elicitation and Analysis.

3.0 Requirements elicitation and Analysis

Requirements analysis is the first stage of the software development lifecycle. It is at this stage that the expectations of the customers are gathered and clearly defined. Any issues regarding the complexities of the requirements are discussed and resolved [29]. Once the requirements have been gathered and analysed, it is then necessary to finish the requirements engineering process by producing requirements documentation. The requirements documentation will be informal and use natural language to produce an agreed statement of the system requirements. This chapter will first describe the requirements gathering process then list both the functional and non-functional requirements of the new safety application. The final part of this section will explore the risks that might affect the build of the new software application along with steps to reduce the chance of potential problems from occurring and possible mitigation strategies.

The requirements documentation can then be used by both the customer and software developer to:

- **Resolve contractual disputes**

Having a requirements document signed by both parties ensures that any conflicts that might arise regarding the requirements of the system can be legally resolved by referring to the document as evidence.

- **Provide a schedule**

A detailed requirements analysis will further provide both parties, the client and the software developer, a clear understand of the software project's schedule. This can serve to prevent any misunderstandings regarding key deliverables and prevent possible contractual disputes.

- **Prevent scope creep**

Scope creep is the tendency of a project's requirements to grow as a project progresses. Small incremental adjustments to the software project may lead to a situation where a project which initially had five basic requirements finishes with having ten. This may often be due to misunderstanding at the requirements elicitation stage or due customer's changing needs. A comprehensive requirements analysis may help to minimise the possibility of scope creep by clearly defining functional and non-functional requirements. Change requests may then be further recorded and communicated to prevent final deliverables taking longer to complete and diverging further from the initial project objectives.

3.1 Requirements Gathering

For the purpose of gathering requirements for the new safety application, requirements elicitation was facilitated through hosting a small focus group with the intention of gaging the thoughts and opinions of key stakeholders. In this way, it was hoped to find out what kinds of functionality potential users would like featured in the new safety application. The focus group involved meeting three potential users of the application over tea and coffee. The results of the focus group are outlined in the table below:

Positives	Negatives
Provides extra information about journeys in terms of areas of dense clusters of crime	Location services always on
Works with minimal interaction	Notification spam
Innovative notifications that attempt to avoid raising fear of crime	Risk of getting robbed while receiving notification in area of high crime
Anonymity in terms of there being no need to sign in and to enter personal details	Possibility of increasing fear of crime

Table 3.1 shows the results of the focus group.

Following the focus group, it became evident that one of the main concerns of the group was that of being constantly tracked via location services. This, however, was not a concern that was linked especially to the new safety application but instead to all applications in general. During the software engineering project, attempts have been made to include an option to disable location services for the software application. A second prevailing concern was that of notification spam. All participants commented about how they would find a safety application that constantly sent push-notifications to their mobile devices very tedious. As a response, much effort has been put into the project so that notifications are timed and do not send out notifications too frequently. One participant mentioned the irony in becoming a victim of crime while checking a safety notification. Although the point was made in humour, however, in working on the application, the developer has made effort to increase the safety of the safety application so that the probability of such an incident occurring is largely negligible. During the focus group a controversial topic was mentioned – fear of crime. Having seriously considered the matter of the potential for increasing the fear of crime during the inception process, the idea of nudges and nudge-based notifications as a potential solution, once explained to the participants, became one of the most liked aspects of the application. Due to this response, it became evident that time and effort needed to be invested in the design of the notifications in order to maximise effectiveness. Not only were participants able to give their opinions concerning what they perceived as benefits and disadvantages of the proposed system, they also proposed requirements that would make the system better. One suggestion was to include the possibility of being able to scroll through nudge-base notifications at the user's

leisure without having to set off a geofence. This was considered by the developer and effort was made to implement the suggested requirement. The focus group greatly aided in the requirements elicitation process and allowed for the design of more detailed and comprehensive functional and non-functional requirements.

3.2 Functional Requirements

The functional requirements of a system state explicitly what the system will perform [24]. More specifically, a comprehensive list of functional requirements will clearly state the system's inputs, outputs, functions, and exceptions [37].

The table below is a list of functional requirements for the new safety application. An important aspect of the table is the use of a priority scale used to indicate the relative importance of the requirement. Various ranking schemes exist, however, for the purpose of the new safety application, a categorical scale was used where H represents high priority, M for medium priority, and L for low priority. Once a list of requirements had been created with each requirement paired to its corresponding priority, the requirements were then arranged and ordered accordingly.

Requirements	Priority
The user's location must be monitored as they walk through areas of Northumbria	H
Predefined geofences must be triggered when entering areas of higher levels of crime	H
Once a geofence has been triggered, a notification must be sent to the user's device	H

The notification must be a nudge-based notification that avoids increasing fear of crime	H
Each nudge-based notification should be random and not specific to each geofence.	H
User's should be able to disable location tracking for the application	M
The application should not excessively spam its user's devices with too many notifications	M
User's should be able to click on a notification to view a nudge in full screen.	M
User's should be able to scroll through a list of nudge-based notifications upon opening the application	M
The application should offer some sort of welcome screen once the application has been successfully opened	L

Table 3.2 Functional Requirements

The functional requirements could then be implemented in order of priority. The advantages of such an approach included being able to show stakeholders iterations of the application at various stages of the development process. This also allows for regular input into the progress

of the system. Also, if the project had fallen behind schedule then the most essential functional requirements would have been met.

3.3 Non-Functional Requirements

Non-functional requirements focus on how well a system performs its tasks. According to Bajpai and Gorthi [5], non-functional requirements are rarely considered in the late phase of development and as a result in some instances have led to serious consequences. If non-functional requirements are not properly accounted for then it may lead to software failures. The reason for the neglect of non-functional requirements in the past, Bajpai and Gorthi assert, is due to the lack of support from any language and tools. If they are stated then they are often stated too informally [5].

Mariza et al. [24], as well as suggesting a unique way in which non-functional requirements may be categorised based on types of systems and application domains, also points out the five most commonly considered non-functional requirements that are used across a variety of software. The new safety system application will focus on these popular categories of requirements to form a comprehensive list of non-functional requirements for the application. These include: performance, reliability, usability, security, and maintainability.

	Requirement	Priority
performance	<p>The application will respond swiftly to direct user interaction without delay.</p> <p>The application will take minimal space on the user's device</p>	M
Reliability	<p>The application can be relied upon to function consistently when required and with reliable accuracy.</p> <p>The application will rarely fail once in use.</p>	H
Usability	<p>The application will take minimal effort in learning to use</p>	

	<p>Nudge-based notifications will be easily understood</p> <p>The application will be regarded as useful by its users.</p>	H
Security	<p>Minimal amounts of personal data will be stored on this application</p>	H
Maintainability	<p>The code used in the building of this application will be clearly written and commented out</p> <p>An MVVC framework will be used so that the code can be easily modified or added to</p> <p>The code will be easily tested</p>	M

Table 3.3 Non-functional requirements

3.4 Risk

Risk, in the context of software engineering projects, may be defined ‘...as the uncertainty that has an impact on project objectives’ and can be described in terms of the likelihood and impact of occurring [24]. Optimistic enthusiasm combined with the Neglecting of risk in software engineering projects may eventually lead to the project’s downfall [7].

3.4.1 Risk Management

Risk management is critical to the success of a software engineering solution. It involves predicting the risks that might negatively influence a project’s schedule or the quality of the software being developed, and acting to prevent these risks from happening [37]. While developing the new safety application, risk management was divided into four main stages: risk identification, risk analysis, risk planning, and risk monitoring as suggested by Sommerville [37].

3.4.2 Risk Identification

Risk identification is the first process involved in risk management and involves identifying all the possible threats and opportunities facing the software engineering project [26]. Moran distinguishes between two different categories of risk: internal and external. Internal risks describe those risks that are more easily controlled while external risks refer to risks that are more difficult to control. Risk identification often involves the collation of project-specific risks that might compromise the project's success [7]. This can be presented in the form of a risk register or checklist.

The new safety application is also exposed to the same possibility of risk that might impede the success of the development of the software application. Therefore, it has also been necessary to create a risk table that clearly shows the potential risks as they relate to this project. This risk table has been designed to include six categories: Technology risks, people risks, organizational risks, tools risks, requirements risks, and estimation risks respectively [37].

Risk Type	Identifiable Risk
Technology	<p>Database may not be able to process the large volume of data</p> <p>'Orange' machine learning and data mining software unable to successfully run k-means algorithm on large data set</p> <p>Location services unable to locate user's position</p> <p>Geofences not triggered and nudge-based notification fail to send</p>
People	<p>Software developer unable to work on project due to illness</p> <p>Inability of the developer to understand software key to success of project</p>
Organisational	<p>Project leader / supervisor or external expertise not available to software developer at critical time.</p>
Tools	<p>Database and data mining tools do not seamlessly work together in android environment</p>

	Emulator or android testing device ceases to work
Requirements	Inadequate understanding of the requirements Changes in requirements that require large changes to existing code Incorrect implementation of requirements
Estimation	Time required for project completion severely underestimated Costs associate with completion of project underestimated

Table 3.4 lists the potential risks to the software application project.

3.4.3 Risk Analysis

The second stage and important part of risk management is risk analysis. There are two main approaches to risk analysis: qualitative and quantitative. The quantitative approach to risk analysis involves the analysis of probability distributions to characterise risk probability and impact. Quantitative risk analysis often makes use of various software applications to model effects on schedules and cost estimations. However, due to lack of experience working with similar projects such data was not available. For the new safety application, a qualitative approach to risk analysis was taken. This involved prioritizing risks while, at the same time, considering the probability of each risk occurring together with their potential impacts.

After having successfully identified the possible risks to the successful completion of the new safety project, it was then necessary to categorise each risk according to the likelihood of the risk occurring and the possible impact it would have on the successful completion of the project.

Risk	Probability	Effects
Database may not be able to process the large volume of data	Moderate	Serious
‘Orange’ machine learning and data mining software unable to successfully run k-means algorithm on large data set	Moderate	Serious
Location services unable to locate user’s position	Moderate	Catastrophic

Geofences not triggered and nudge-based notification fail to send	Moderate	Catastrophic
Software developer unable to work on project due to illness	Low	Serious
Inability of the developer to understand software key to success of project	Very Low	Catastrophic
Project leader / supervisor or external expertise not available to software developer at critical time.	Moderate	Serious
Inadequate understanding of the requirements	Very Low	Serious
Changes in requirements that require large changes to existing code	Moderate	Tolerable
Incorrect Implementation of requirements	Low	Serious
Time required for project completion severely underestimated	High	Tolerable
Costs associate with completion of project underestimated	Low	Tolerable
Database and data mining tools do not seamlessly work together in android environment	Moderate	Tolerable
Emulator or android testing device ceases to work	Moderate	Tolerable
Accidental loss of project and software	Moderate	Catastrophic

Table 3.5 categorizes the risks in order of severity as either low, medium, or high.

3.4.4 Risk Planning

The next stage in risk management is that of risk planning. It was at this stage that it was necessary to consider the key identified risks and then to develop strategies to manage these risks. Key risks were identified and defined as those risks whose effects were classified as either serious or catastrophic. Strategies were then devised based on Somerville's [37] classification of risk strategies: avoidance strategies, minimisation strategies, and contingency plans.

Risk	Strategy
Database Performance	Set aside time to research different databases and select one based on the project's specific requirements

Server and Algorithm Performance	Research servers capable of running algorithms on large amounts of data on the server Compare advantages of using different algorithms
Location and geofence performance	Use reliable location services such as Google API
Software developer illness	Ensure easy access to surgeries and hospitals
Lack of key software knowledge	Thorough research and practise with relevant technologies
Project leader/supervisor availability	Ensure list of appointments for coming weeks so availability is known about well in advance. Arrange other alternative forms of communication such as e-mail and skype if necessary
Misunderstanding requirements	Focus groups and inspections to further ensure clarification on requirements
Accidental loss of project and software	Regularly backup work
Requirements changes	Maximise information hiding in design
Wrongly implemented requirements	Use agile-like approach to project that allow requirements to emerge and evolve. This will ensure that the end solution will reflect the requirements.

Task 3.6 offers a list of risk mitigation strategies.

3.4.5 Risk Monitoring

The risk monitoring process involves regularly checking whether the probability of the previously defined risks have changed. At this stage, it is also important to assess whether the impacts of each risk have changed also. One factor that might be indicative of changes in the probability of risks and impacts is the number of requirements change requests [37]. The risk monitoring stage of the process ensures a closed-loop process providing risk-reduction can be continuously tracked and corrective measures applied [7].

3.5 Summary of the Requirements and Analysis

This chapter discussed the process of requirements elicitation for the new safety application. Requirements were gathered, listed, and then sorted in order of priority. This helped during the build process to insure the most important functions of the application were included before attempting the lower priority ones. The last section contemplated the risks that might prevent the project from either being successfully completed or hinder its progress. These risks were listed, categorised, and then strategies were designed to mitigate these potential risks. The next section will consider the design of the new safety application.

4.0 System Design

In this chapter, a software development strategy will be chosen and diagrams of the system will be presented. The chapter starts by introducing agile development and the benefits of such a development methodology. Sequence diagrams are then introduced to show how events switch between objects in the new application. Use case diagrams were used in this section to create a diagrammatic view of the user requirements and wireframes were designed to provide a description of how the user will interact with the interface. It is in this chapter that the nudge-based notifications are then designed with an explanation of the techniques used and expected outcomes. The system design will guide the developer in the implementation of the new safety application.

4.1 Agile Development

Agile was the choice of development methodology for this software project. Agile development places emphasis on delivering good quality working software in short periods of time. It handles changes to the requirements throughout the length of the software development life cycle and allows each developer to work to their strengths with the success of an agile project being largely measured by the output of working software [30]. The development process encourages regular customer input which can ensure that by the end of the software contract, the software is likely to fulfil its functional and non-functional requirements and thus be accepted by the customer. Due to the time constraints placed on the project, agile was good choice of development strategy as it ensured that requirements could be achieved at regular intervals. After each requirement had been met, it was then possible to discuss with potential users their opinions about how well the requirement had been met and then make further adjustments accordingly. After having successfully fulfilled one requirement, work on the next requirement could then be started. Even if by the end of the project some requirements had not been met, by using the agile development methodology, it was possible to ensure that a working software project could be delivered.

4.2 Sequence Diagram

Sequence diagrams are behavioural models that show how events can cause transitions between objects as a function of time [30]. The arrows in the sequence diagram as shown below represent an event and the way in which the events switch between the objects in the new application.

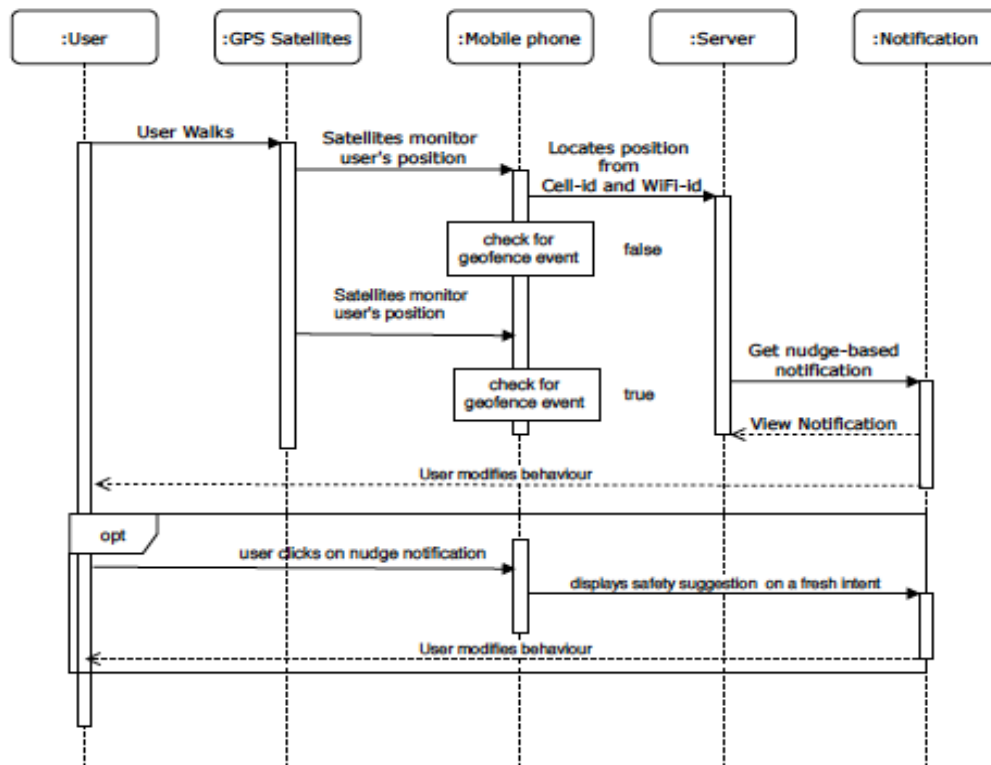


Figure 4.1 A sequence diagram showing events switch between the objects in the new application.

Each vertical narrow rectangle in the sequence diagram represents the time spent processing a given activity while the vertical dotted lines represent changes in an object's state.

- The first event originates from the user object. While the user ambulates to their desired location, the behaviour is channelled to the GPS satellites object where their position is continuously monitored.
- A request lookup is then passed to the mobile phone object which in turn queries a database on the server object to check for a geofence event.
- If a geofence event is returned then the flow continues to the notification object where a relevant nudge-based notification is returned.
- Upon viewing the nudge-based notification, the user will then adjust their behaviour in order to further reduce their chances of becoming a victim of crime and thus increase their own safety.

- The user also has the option of clicking on the notification so that they can view the nudge which will allow the intent to be viewed in full screen on their android device.

The sequence diagram was very useful in modelling the way in which events cause transitions between the system application's objects which then allowed for a clear representation of the system's input and output events. The design of the sequence diagram was highly beneficial for the system application's successful development.

4.3 Use Case Diagram

A use case can be used to elicit precise requirements for a system [36]. In this project, the use case was largely helpful in further elucidating the potential users' expectations of the system and allowed for the documentation of the requirements that the system should fulfil. Seidl et al. [36] explain that use cases can be very important for detailed technical designs and if they are neglected or incorrectly implemented the consequences can be severe. Such consequences, it is explained, might include user dissatisfaction, increased maintenance costs, and the incorrect use of the system that might prevent expected returns.

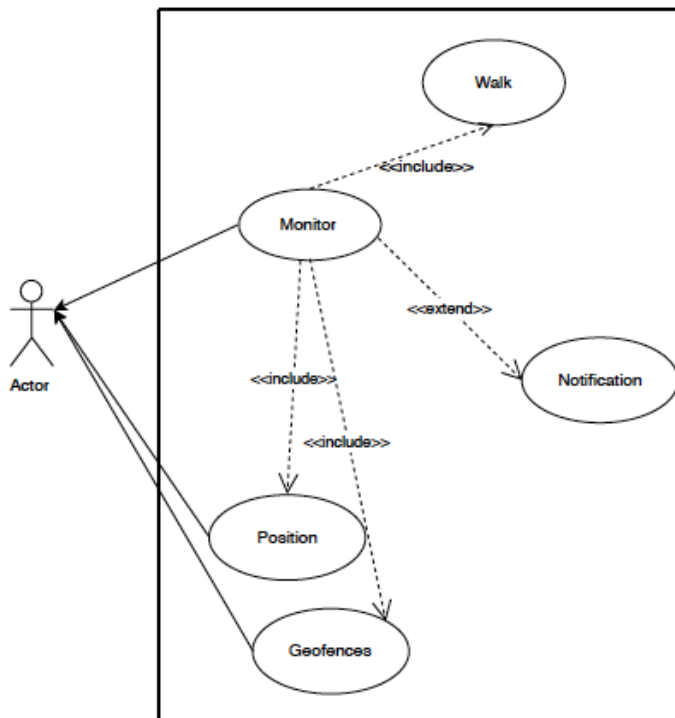


Figure 4.2 A use case diagram showing users' interactions with the system

The above is a user case diagram that was used to build the new safety application.

Use case: Sending Nudge-based Safety Notifications

Primary actor: Mobile user

Goal in Context: To ensure that when a geofence event is triggered, a nudge-based Safety notification is sent to the user.

Preconditions: Mobile device must give permission for the new safety application to send notifications to the phone. The user must also ensure that the application is able to access GPS location services.

Trigger: Once the user enables geolocation services, whenever the user enters a boundary that is defined by the geofence, a nudge-based safety notification will be sent to the phone.

Scenario:

1. User: Allows the android application continued access to GPS location services.
2. User: Opens application and allows it to continue running in the background.
3. User: Enters boundary defined by geofence.
4. User: Receives nudge-based notification.
5. User: Influenced to modify their behaviour to better avoid becoming a victim of crim.

Exceptions:

1. New safety application fails to monitor user's location: user checks whether GPS location services are enabled for both the mobile device and the application; checks to see if the application is still running in the background.

Priority: Essential, must be implemented

When available: First increment

Frequency of use: Regular daily use

Channel to actor: Primarily via phone's push notification interface

4.4 Wireframe Diagrams

To help with the design of the screen, a wireframe diagram was first drawn. A wireframe is a two-dimensional sketch of the intended screens that can be used to define the information hierarchy of the application. A wireframe diagram may also be used to provide a description of how the user will interact with the interface.

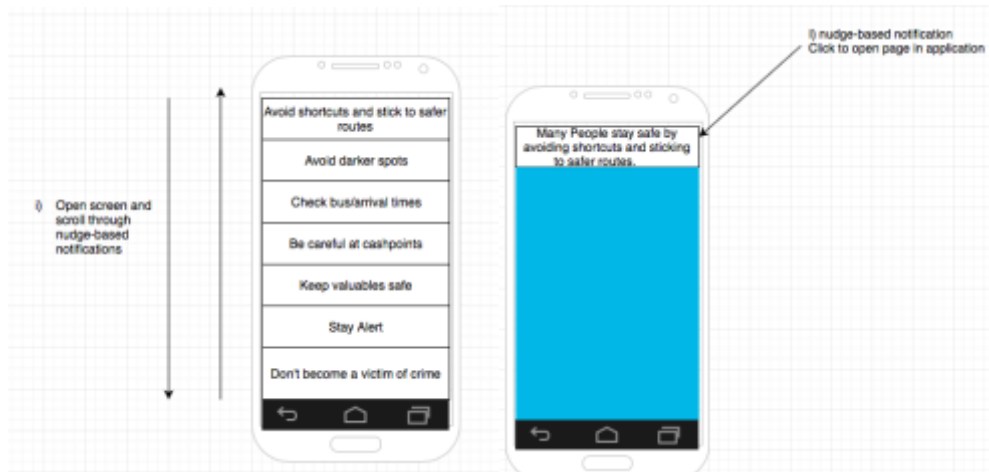


Figure 4.3

Figure 4.4

Upon opening the new safety application, the user has the choice of viewing each nudge-based message by either scrolling through the list of messages or by triggering a predefined geofence

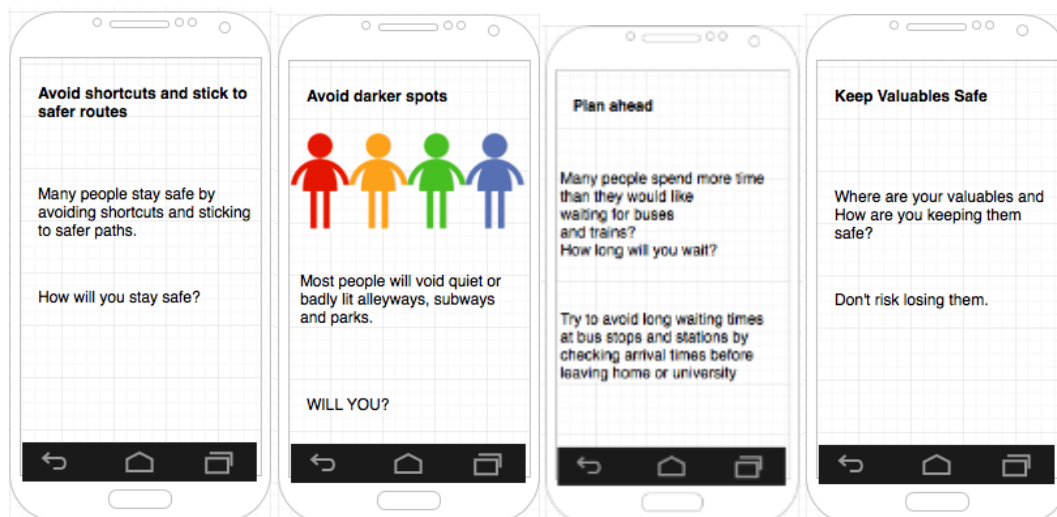


Figure 4.5

Figure 4.6

Figure 4.7

Figure 4.8

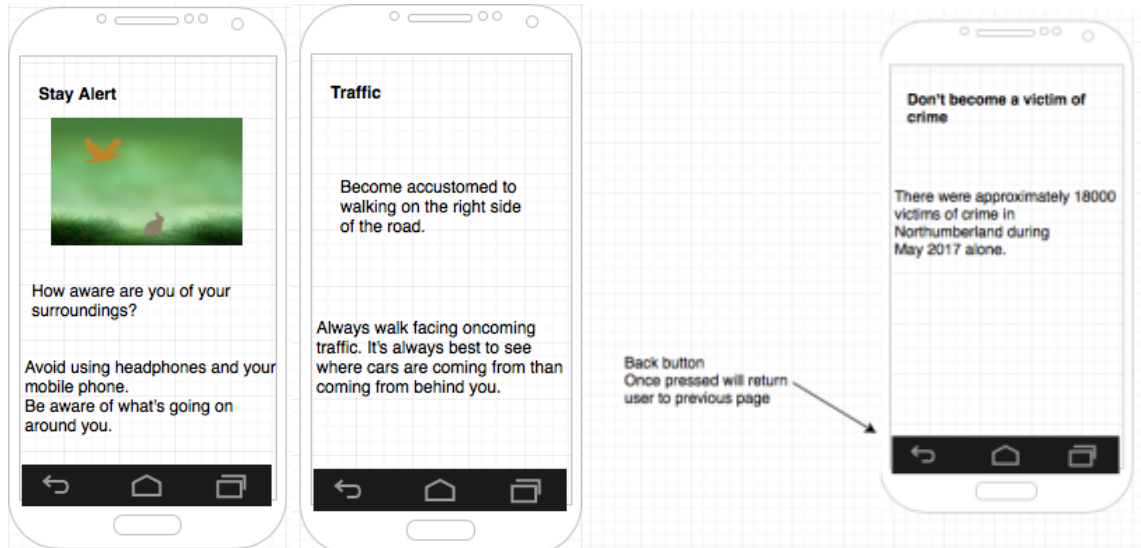


Figure 4.9

Figure 4.10

Figure 4.11

Once a user has clicked on an item from the scroll down list or the nudge-based notification itself, then the user will be navigated directly to a fresh page with the full message displayed.

4.5 Designing the Nudge-based Notifications

In order to offer effective nudge-based notifications to users of the new safety application, it was first necessary to research reliable safety advice. After considering the various options, it was decided that the primary source of this safety advice would be sourced from the Northumbria Police website [28] and the University of Newcastle Upon Tyne website respectively [38].

Initially, 12 suggestions for improving safety in public were collated and then further examined to see which ones could be adapted to create nudge-based notifications. In the end, seven suggestions were selected:

- **Avoid shortcuts and stick to safer routes**

This message, as seen in figure 4.3, has been adapted with the use of the availability heuristic, mere-measurement effect, and priming. By suggesting that most people avoid shortcuts in favour of safer routes, it is hoped that any exaggerated sense of shortcuts being a positive thing are reconsidered by the user. Accordingly, the user should then modify their behaviour in such a way that they would consider their safety first before pursuing any potential shortcut. Furthermore, in order to accentuate the effect of the availability heuristic in this example, the user is then asked how they will stay safe. Asking the user to measure how they plan to carry out an action makes them more likely to carry out that said action.

- **Avoid darker spots**

Figure 4.4 is also an example of how the availability heuristic might be used to nudge users into behaving more safely when out walking. By informing the user that most people will try to avoid quieter, dimly lit streets serves to remind the user of examples of how people might go out of their way to avoid these dimly lit areas for the sake of safety. This, it is hoped, will nudge the user into behaving in a similar fashion. Including an image of several people lined up next to each other aims to reinforce the nudge. Asking the user whether they will also stay safe and avoid dimly lit areas is an example of using priming to help the user make better decisions about their own behaviour.

- **Plan ahead**

Figure 4.5 is an example of using a combination of the availability heuristic, priming and framing to influence user behaviour. First, the user is presented with a statement that claims many people spend too much time waiting for buses and trains. This serves to anchor the user's expectations of how long they might expect to wait for their own bus. When asked how long they might have to wait for a bus, it is hoped that the user will also expect to wait for similarly long periods of time and then go out of their way to mitigate their waiting time by effectively planning ahead.

- **Keep valuables safe**

Figure 4.6 attempts to use priming and risk aversion techniques to nudge users into taking better care of their personal valuables and thus reducing the chances of becoming a victim of crime. The user is first asked about their valuables location and how they are going to keep these items safe. By prompting users to think about how they might keep these items safe, users are more likely to act and make their valuables safer. The user is then explicitly told not to risk losing them. This takes advantage of people's natural tendency to avoid losing things that are already in their possession.

- **Stay Alert**

Figure 4.7 also makes use of priming to bring attention to the user's surroundings. It is hoped that the user's attention will then turn to the safety of his or her environment. The use of a picture with a rabbit and a bird of prey that flies directly over the oblivious rabbit's head is used to accentuate the need to be aware of one's surroundings and safety.

- **Walk in the direction of oncoming traffic**

Figure 4.8 is an example of utilizing the status quo bias. It is an attempt to change the user's behaviour so that they choose to instinctively walk down the right-hand side of a road. This would ensure that users will walk toward traffic and will as a result be able to see where cars are coming from.

- **Don't become a victim of crime**

Figure 4.9 is an attempt at applying the availability heuristic and the unrealistic optimism principle.

Unrealistic optimism is a trait that pervades the life of all humans. This can lead to people taking unnecessary risks in their daily lives. It is hoped that by expressing a true statistic that it will remind users of the incidents where people have become a victim of crime and lead to changes in their behaviour that will increase their own safety.

4.6 Summary of the System Design

This chapter begun by specifying agile as the choice of development methodology for the new project. Sequence, user case, and wireframe diagrams were then designed to guide the development process. It was also in this chapter that the nudge-based notifications were designed along with descriptions of the nudges and techniques used. Having carried out requirements gathering and analysis along with functional and non-functional requirements, risk analysis and management also, the project is ready to move to the next stage of development; implementation.

5.0 System Implementation

The implementation of the solution began with a large crime data set [11] in the form of a CSV file containing the details of crimes committed in Northumbria during May 2017. The kind of crimes recorded included violence and sexual offences, anti-social behaviour, theft and public order offences just to name a few. It was the intention of the project to utilise this data to create geofences so that users would be informed via a nudge-based notification if they had entered an area with higher densities of crime. However, altogether, the CSV file contained a total of 17,995 recorded crimes. Due to the sheer size of data, any kind of direct manipulation was impractical. It became immediately evident that the first challenge would be to process and sort this data before it could be used in any meaningful way. However, at the same time, it would be essential to retain the essential character of the data. The second part of the implementation involved using the cluster algorithm results to build the android application.

5.1 Orange3- Data Mining Package

Further research into data mining and machine learning led to an investigation of various algorithms used to group and sort data. Finally, it was decided that the K-means algorithm could be used to sort the data into different clusters based on longitude and latitude pairs. Due to the size of the data, however, this processing could not be implemented on an android device and so it was necessary to find a suitable environment to carry out the necessary data processing. Orange3 was a perfect candidate as it provided free open source machine learning, data visualization and data analysis functionality. Furthermore, Orange3 allowed for the running of a K-means algorithm over large data sets.

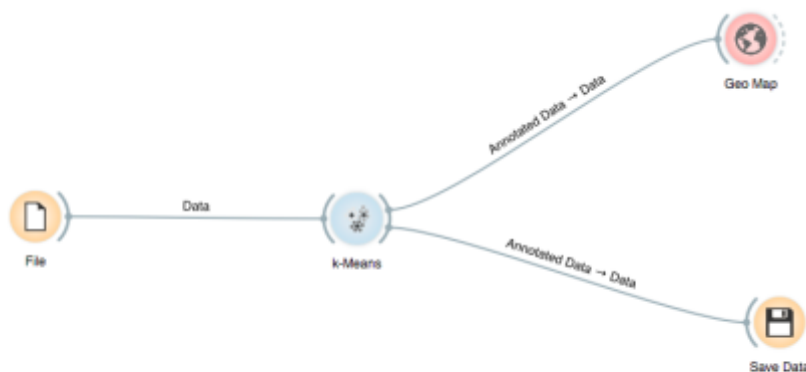
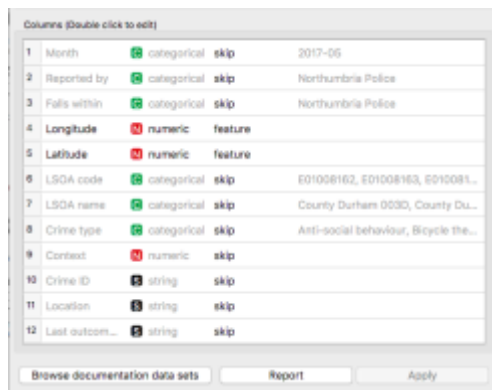


Figure 5.1 orang3 setup for the processing of the project's data

Figure 5.1 shows the process that was set up in order to run the K-means algorithm on Orange3.

After gaining an understanding of how the K-means algorithm works and some experience of using the Orange3 software package, it was possible to set up a process in which data could be uploaded to Orange3 as represented by the image of the file in Figure 5.2. The K-means algorithm could then be run on the data and passed to Geo Map where the results of the K-means algorithm could be visualised in real time on a map of Northumbria. The final output of the data produced by the K-means algorithm could then be saved to a computer as represented in the diagram by a floppy disk.

Uploading data to Orange3



Columns (double click to edit)			
1	Month	categorical	skip
2	Reported by	categorical	skip
3	Falls within	categorical	skip
4	Longitude	numeric	feature
5	Latitude	numeric	feature
6	LSOA code	categorical	skip
7	LSOA name	categorical	skip
8	Crime type	categorical	skip
9	Context	numeric	skip
10	Crime ID	string	skip
11	Location	string	skip
12	Last outcome...	string	skip

Figure 5.2 uploading data to orange interface

Figure 5.2 shows the setup of the file once the data had been uploaded to Orange3. The list numbered from 1-12 shows the various columns present in the original data set. At this stage, the data was filtered manually so that only location data would be processed. This was done by only selecting longitude and latitude values to be passed to the k-means algorithm.

5.2 The K-means Algorithm

The K-means algorithm is a popular data mining and machine learning algorithm that can be used to organise large groups of data points into a smaller number of [14]. The K-means algorithm is an excellent method to achieve data reduction and by using clustering on the large crime dataset for this project, it was possible to reduce the complexity and size of the dataset. The advantages of clustered data include the need for significantly less storage space and the easier and faster manipulation of data compared to the original dataset [14]. This meant that using clustering techniques such as the K-means algorithm was perfect for processing data for the use in the new android safety application.

The K-means algorithm is a clustering algorithm and is often referred to as unsupervised learning as the data is unlabelled. The algorithm starts with two inputs: the number of clusters and the data set. The algorithm then randomly selects points in space in which to place the cluster centroids. Once the centroids have been placed, the distance between each data point and all centroids are calculated before assigning each data point to the nearest cluster centre.

The cluster centre is then recalculated for each cluster [27]. This is an iterative algorithm in that each process is repeated until convergence is achieved.

The K-means algorithm may be best represented in mathematical notation as:

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

Figure 5.3 K-means algorithm

Where J is the objective function, k is the predefined number of clusters, n is the number of cases, and x_i is case i and j is c_j the centroid for cluster j [14].

To summarise the algorithm processes:

1. Data clustered into k groups.
2. K points chosen at random as cluster centres.
3. Data objects then assigned to nearest cluster centre.
4. Calculate the arithmetic mean of all data objects in each cluster.
5. Repeat until convergence

Setting up the K-means algorithm

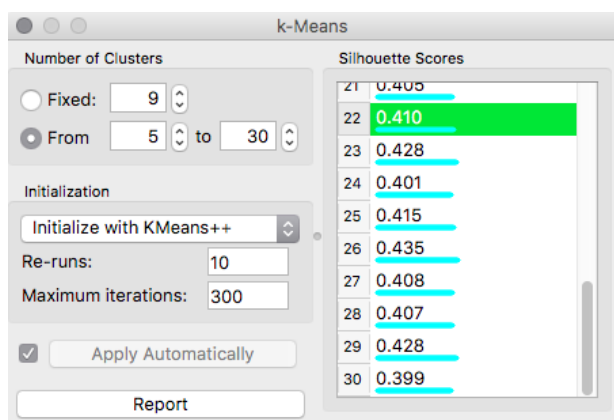


Figure 5.4 configuring k-means

Figure 5.4 shows the K-means widget used for setting the conditions of the cluster algorithm. The initialization option allowed the choice of whether to have the first centroid placed randomly and then successive centroids placed based on the probabilities associated with the squared distance from the closest centre or to just have all the centroids placed randomly and then updated with further iterations. The former choice was chosen as it offered greater efficiency.

The re-runs refer to the number of times that the algorithm will be run. For this project, with approximately 18,000 pairs of data to be processed, it was decided that ten times would be sufficient for this sized dataset. Maximum iterations refer to the maximum number of iterations per algorithm. Although the number was set at the default 300, it was expected that the total number of iterations required per algorithm would be significantly less.

One of the disadvantages of using the K-means algorithm is that the user must manually declare the total number of clusters required. This is problematic as most methods used to estimate the required number of clusters are usually subjective. As seen in Figure 5.4, 22 clusters were chosen for this project. The reason for this decision was that too few clusters resulted in each city or town being marked as a potential cluster. The problem with this is that too much detail would have been lost about street level and area crime within the city. To resolve this problem, the developer carefully increased the number of clusters while checking the newly created clusters on the Geo Map shown in in Figure 5.5. As the clusters were slowly increased, boundaries produced within individual cities and towns became clearly defined.

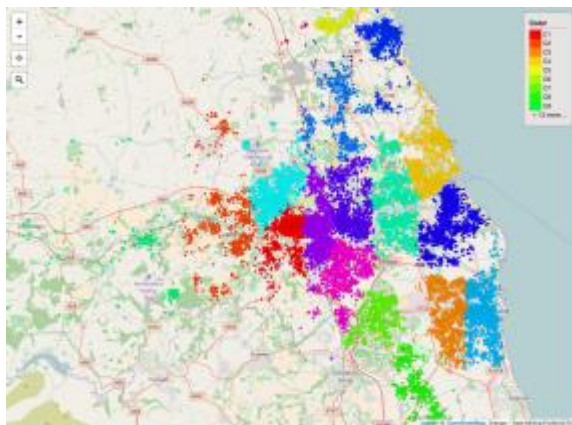


Figure 5.5 visualised representation of the processed K-means output when 22 cluster centres were chosen.

Figure 5.5 shows a screenshot from Geo Map on Orange3. It shows the results of the clusters once the K-means algorithm has finished running. Each colour dot represents a crime data point which has been grouped into one of 22 categories ranging from C1 to C22.

Having successfully run the K-means cluster algorithm over the crime dataset and choosing an appropriate number of cluster centres with the help of Geo Map, the new dataset was then saved to the computer.

	A	B	C
1	Longitude	Latitude	Cluster
2	-1.82201	54.904934	C10
3	-1.82201	54.904934	C10
4	-1.560279	54.875639	C7
5	-1.560279	54.875639	C7
6	-1.508855	54.848348	C8
7	-1.50731	54.846059	C8
8	-1.508855	54.848348	C8
9	-1.508855	54.848348	C8

Figure 5.6 preview of processed data

Figure 5.6 shows a preview of data once it had been processed and saved to a CSV file on the computer. Longitude and Latitude data remain unchanged with the total number of longitude and latitude pairs still at 17,995. Column C, however, has been added to the data. This column associates each longitude and latitude pair with a cluster index from C1 to C22.

5.3 Calculating New Centroid Values

Although Orange3's software package successfully run a K-means algorithm over the crime dataset to categorise the data into clusters, the software does not provide the new longitude and latitude values of each cluster centre/centroid location. To create the new geofences it is essential that the centroid's longitude and latitude is known. For this reason, the next challenge was to write a program to further process the new data to find each cluster's centre. Fortunately, due to the proximity between data points it was possible to calculate the centroid's centre of gravity by merely calculating the arithmetic mean of all the longitude and latitude data points that fell into each cluster category.

Each new longitude and latitude value was calculated using the arithmetic mean using:

$$\begin{aligned}
 & \frac{1}{n} \sum_{i=1}^n a_i, \frac{1}{n} \sum_{i=1}^n b_i \\
 & = \frac{1}{n} (a_1 + a_2 + a_3 + \dots + a_n), \frac{1}{n} (b_1 + b_2 + b_3 + \dots + b_n)
 \end{aligned}$$

Figure 5.7 formula used to calculate the arithmetic mean.

where n is the total number of data points associated with a given cluster, i is the starting point, a is the longitude, and b the latitude.

The first step in calculating the new centroid values was to upload the newly clustered data to a new database. This allowed for the relevant data to be extracted from the datasets with relative ease. The average longitude and latitude pairs could be calculated using the following SQL statement where data is the name of the table:

```
SELECT AVG(Longitude), AVG(Latitude) FROM data WHERE Cluster = 'C1';
```

However, due to the data having been organised into 22 clusters for this project, it was more efficient to write a short piece of code that was able to iterate through all clusters to provide the average longitude and latitude values. The advantages of automating this process include being able to easily scale up the new safety application by adding new crime data from other nearby regions of the country. However, if the distances between data points increased significantly the arithmetic mean would no longer be sufficient in determining the centroids' location and another method would have to be used.

PreparedStatement objects may be used to store precompiled SQL statements. By using PreparedStatement it was possible to use the object to execute the query many times, improving speed and efficiency as it only needs to be compiled once [32].

After connecting to the database, two variables were created and set to two separate queries where countCluster was used to count the total number of unique centroids in the cluster column:

```
String sql = "SELECT AVG(Longitude), AVG(Latitude) FROM data WHERE Cluster = ?";
```

```
String countCluster = "SELECT COUNT(Distinct Cluster) FROM data";
```

A variable could then be easily created and inserted at the point of the question mark in the sql statement:

```
String select = "c" + i;  
preparedStatement.setString(1, select);
```

The final step in processing the data was to print the final results to a text file. This was carried out using the PrintWriter class.

```
54.9670207676473,-1.6531737745097979  
54.97809254908487,-1.7388580798668873  
54.90495062674655,-1.4310478083832325  
55.025177613346386,-1.4567108155053978  
55.16872628395051,-1.5692410695847354  
55.32574225000026,-1.5953984117647066  
54.900518182300864,-1.5300750991150451  
54.8414810153256,-1.472690404214559  
55.004228553488446,-2.188129897674416  
54.955800611940305,-1.8679910796019905  
54.99176307075072,-1.5076698248490024  
55.00033905137961,-1.672317118934346  
54.90333333967906,-1.3855969071980998  
55.06348013541658,-1.5862041020833335  
55.11988987313433,-1.522427169154225  
54.97712964710421,-1.4324236818532836  
54.985813453416306,-1.568767832298135  
54.97100691842296,-1.6122596662763558  
55.424954767123296,-1.722146253424658  
54.93604305951231,-1.5761276360975582  
55.72063595364238,-2.0162177350993393  
55.167463016666645,-1.6825981944444433
```

Figure 5.8 centroid data

Figure 5.8 shows the final output from the java program and lists the 22 centroid longitude and latitude pairs that were used in the android application to create unique geofences.

5.4 Building the Android Application

Once the data had been processed and the cluster centres / centroids calculated, it was then necessary to use this data to build the android application.

5.4.1 Implementing the Location Tracking Requirement

This requirement was met through the configuring of the manifest and gradle.build files and allows the user's to be tracked while walking outside.

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

Figure 5.9 Location tracking permissions

This permission is configured in the manifests file and allows access to GPS.

```
compile 'com.google.android.gms:play-services-location:11.0.0'
```

Figure 5.10 play-services-location dependency

The dependency is then added to the build.gradle file.

5.4.2 Implementing the Notification and the Nudge-based Notification Requirement

Once a user enters a geofence boundary, a notification is sent to the user. Furthermore, these are designed to be nudge-based notifications that will offer valuable safety information that will aim to change the user's behaviour so that they can better avoid becoming a victim of crime.

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
```

Figure 5.11 notification builder

A notification builder is first created so that notification properties can be set :


```
builder.setSmallIcon( com.google.android.gms.location.sample.geofencing.R.drawable.common_full_open_on_phone)
    .setLargeIcon(BitmapFactory.decodeResource(getResources(),
        com.google.android.gms.location.sample.geofencing.R.drawable.common_google_signin_btn_text_light_focused))
    .setColor(Color.BLUE)
    .setContentTitle(notificationDetails)
    .setContentView("Click notification to view full message")
    .setContentIntent(notificationPendingIntent);
```

The final part of issuing the notification is to call:

```
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, builder.build());
```

Figure 5.12 notification issued

The nudge-based notifications were designed (see chapter 4.4) and the files copied into the resource drawable file in the new android project application folder:

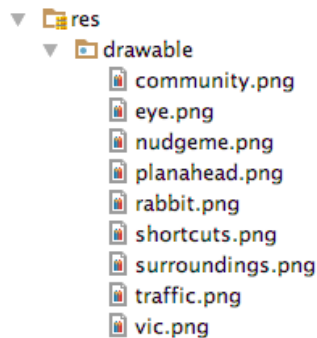


Figure 5.13 list of images in drawable

These could then be accessed using the getMessage() method which can be seen in 5.4.4.

5.4.3 Implementing the Random Notification Requirement

When a user enters a geofence, a notification will be triggered. However, initially, one message was assigned to each geofence. This meant that someone who travelled the same routes would always receive the same messages. This presented a new challenge as to how to make the messages random regardless of which geofences were triggered.

```

public int getRandomNumber(){
    Random rand = new Random();
    n= rand.nextInt(7)+1;
    return n;
}

```

Figure 5.14 getRandom() method

First, a getRandomNumber() method was declared in the Notification Activity class. This method's purpose is to generate a random number between 1 and 7.

```

private String getMessage(int transitionType) {
    if(transitionType==Geofence.GEOFENCE_TRANSITION_ENTER) {
        getRandomNumber();
        if (n == 1) {
            return "Avoid Shortcuts and stick to safer routes";
        } else if(n==2){
            return "Avoid Darker Spots";
        }else if(n==3){
            return "Plan ahead";
        }else if(n==4){
            return "Keep Valuables safe";
        }else if(n==5){
            return "Stay Alert";
        }else if(n==6){
            return "Walk in the direction of traffic";
        }else if(n==7){
            return "Do not become a Victim of Crime";
        }
    }
    return null;
}

```

Figure 5.15 getMessage() method

Next, condition statements were added to the getMessage() method. Once a geofence had been triggered, a call to the getRandomNumber() method was made. Based on the random output, the condition statements decide which message should be included in the notification.

5.4.4 Implementing the Welcome Screen Requirement

The welcome screen requirement acts as the portal to the application. Once the application icon has been clicked, the splashscreen will be activated for 3 seconds before starting the menu screen.

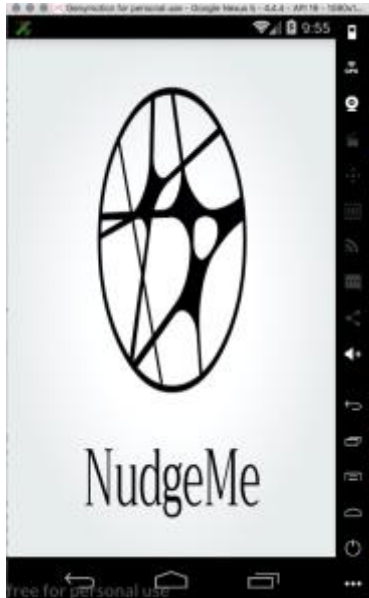


Figure 5.16 splash screen

This was achieved with the use of a small multi-threaded concurrent program:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView( com.google.android.gms.location.sample.geofencing.R.layout.activity_splash_screen);  
    Thread myThread = run() -> {  
        try {  
            sleep(3000);  
            Intent intent = new Intent(getApplicationContext(), com.google.android.gms.location.sample.geofencing.MainActivity.class);  
            startActivity(intent);  
            finish();  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
    };  
    myThread.start();  
}
```

Figure 5.17 multi-threaded program for splashscreen

A second activity was created that was set to display the `activity_splash_screen`. A new thread was then run for 3 seconds before opening to the `MainActivity` with the menu screen.

5.4.5 Implementing the Notification Scroll Requirement

On reaching the main menu screen the user may scroll through a list of safety notifications. Once an option is selected, it can be opened to reveal the full nudge-based message.

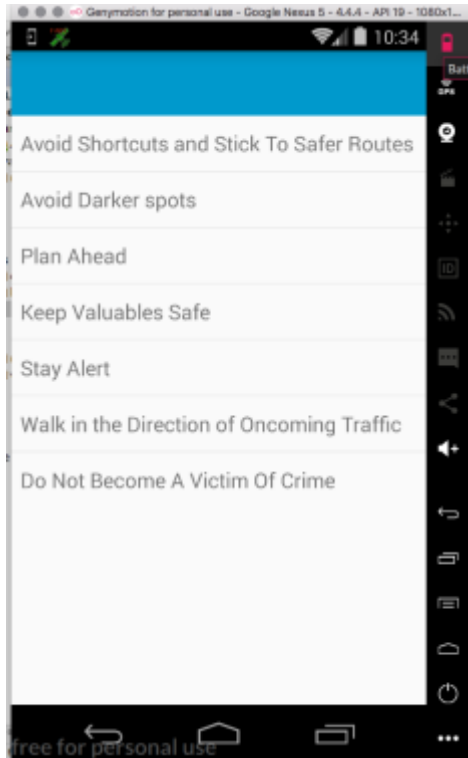


Figure 5.18 menu scroll screen

A new Activity class was created called Main2Activity. The toolbar's assigned value was then checked against the messages stored in strings.xml in the value file:

```
<string-array name="nudges">
    <item>Avoid Shortcuts and Stick To Safer Routes</item>
    <item>Avoid Darker spots</item>
    <item>Plan Ahead</item>
    <item>Keep Valuables Safe</item>
    <item>Stay Alert</item>
    <item>Walk in the Direction of Oncoming Traffic</item>
    <item>Do Not Become A Victim Of Crime</item>
</string-array>
```

Figure 5.19 string file

```

Bundle bundle = getIntent().getExtras();
if(bundle != null){
    mToolbar.setTitle(bundle.getString("nudges"));

    if(mToolbar.getTitle().toString().equalsIgnoreCase("avoid shortcuts and stick to safer routes")){
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, R.drawable.shortcuts));
    } else if(mToolbar.getTitle().toString().equalsIgnoreCase("avoid darker spots")){
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, com.google.android.gms.location.sample.geofencing.R.drawable.community));
    } else if(mToolbar.getTitle().toString().equalsIgnoreCase("plan ahead")){
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, com.google.android.gms.location.sample.geofencing.R.drawable.planahead));
    } else if(mToolbar.getTitle().toString().equalsIgnoreCase("keep valuables safe")){
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, com.google.android.gms.location.sample.geofencing.R.drawable.eyeball));
    } else if(mToolbar.getTitle().toString().equalsIgnoreCase("stay alert")){
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, com.google.android.gms.location.sample.geofencing.R.drawable.surroundings));
    } else if(mToolbar.getTitle().toString().equalsIgnoreCase("traffic")){
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, com.google.android.gms.location.sample.geofencing.R.drawable.traffic));
    } else if(mToolbar.getTitle().toString().equalsIgnoreCase("Do not become a victim of crime")) {
        picture.setImageDrawable(ContextCompat.getDrawable(Main2Activity.this, com.google.android.gms.location.sample.geofencing.R.drawable.vic));
    }
}
}

```

Figure 5.20 setting up scroll screen

Using condition statements, a picture variable was assigned to an image from the drawable file depending on the value of the toolbar variable.

5.5 Summary of the Implementation

This chapter was split into two main sections. The first section described the challenge of processing large datasets into manageable information that could be used to create geofences. To do this it was first necessary to make use of a data mining and machine learning package called Orange3. This software package allowed for the data to be uploaded where a K-means algorithm could be used to process the data into clusters. Using the Geo Map module on Orange3 to select a suitable number of clusters, the dataset could then be sufficiently categorised. The new output file now contained the same sized dataset, only now it had been categorized into 22 categories. However, even at this stage, the data was still too large to work with so the developer wrote a simple java program using SQL to find each cluster's centre of gravity. This finally reduced the original dataset from 18000 to 22. The 22 new centroids could now be used to create geofences. The second stage involved using the newly processed data and integrating it into an android application that could create the geofences and issue nudge-based notifications. It was in this section that some of the solutions to the requirements were described.

6.0 Testing

Once the implementation of the software application had been successfully completed, it was then necessary to ensure that the finished program had achieved what it had set out to do. The application was therefore tested to ensure that it met requirements and that any elusive defects might be identified. There are numerous approaches to software testing that have both their advantages and disadvantages. This chapter will describe black and white box testing, system integration testing, and acceptance testing respectively. The chapter provides details regarding the assessment of the functional and non-functional requirements and how the geofences were tested.

6.1 Black and White Box Testing

Black box testing is an effective form of software testing that takes little consideration of the internal logical workings of the software and code and is instead concerned with the testing of the software's functions. It is an external view to software testing with a focus on a system's functional requirements [30]. An advantage of adopting the black box approach to testing is that the person doing the testing is somebody different from the person who carried out the development of the project. This can ensure that testing remains unbiased. Other advantages include not needing specialist knowledge of the code to carry out testing, testing can be carried out by many people and can cover many test cases. Also, the test cases can be produced as soon as the functional requirements have been stated. White box testing, however, is directly concerned with the working structure of the software to be tested. For this reason, it is necessary that someone carrying out white box testing has an intimate knowledge of the code and software. White box testing allows for the creation of test cases that help to ensure comprehensive testing of the software's logic. However, it is often difficult to check every single path for errors and therefore means that oftentimes errors can go undetected [30]. Black box testing was the main approach adopted for this project.

6.2 System Integration Testing

System testing also falls within the purview of black box testing. Under System integration testing, all the software's components and system elements are brought together and tested as a whole [30]. During the testing of the new software application, an attempt was made to adopt black box testing with the use of a powerful model-based testing technique known as state transition testing. State transition testing may be applied to systems which can only be in a finite number of different states at any given time. Such systems are often referred to as finite state machines [43].

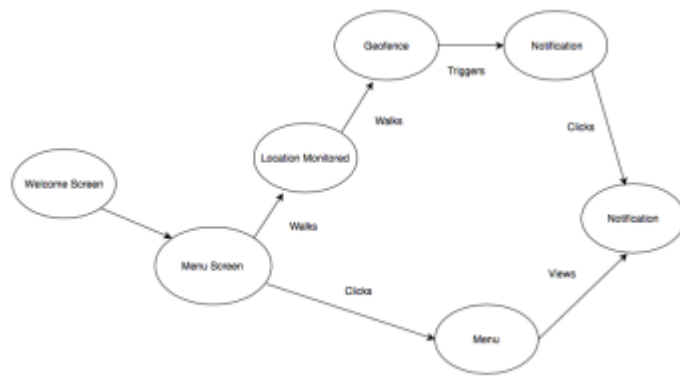


Figure 6.1 State Transition Diagram of the new safety application

The state transition diagram as shown in Figure 6.1 shows each test case as represented by a path, and the states represented as circles. The state transition diagram helped to derive test cases so that the functional requirements could be effectively tested.

6.3 Functional Requirement Testing

Test	Expected Result	Actual Result	Test Result: Pass/Fail
User's location is monitored: User walks.	User's locations should be monitored.	While the user is walking in public, their locations is monitored.	Pass
User triggers geofence(i): The user walks into an area defined by a virtual geofence.	A geofence event should be triggered.	After walking into a predefined geofence, a geofence event is triggered.	Pass
User triggers geofence(ii):	A notification should be sent to the phone.	Once a geofence has been triggered, a notification is sent to the user's phone	Pass

Push-notification sent:	The notification should be a nudge-based notification that avoids increasing fear of crime.	The notification that is sent to the phone is a nudge-based attempt at changing behaviour without increasing fear of crime.	Pass
User should be able to disable location tracking:	The user should be able to disable location tracking once the applications has been opened.	The user does not have an option to turn off location tracking once the application has been turned on.	Fail
User remains in area defined by virtual geofence:	The user should not be spammed with safety notifications.	The user enters and remains in an area predefined by virtual geofences and does not receive excessive safety notifications.	Pass
User clicks notification:	The user should be able to view the nudge-message in full-screen.	When the user clicks the notification, they are able to see the full message in full-screen.	Pass
Nudge-based Notification: User opens the application:	The user should be able to view a list of all the safety suggestions, scroll through them and click on each one for more details.	Upon opening the application, the user can view all safety notifications, scroll through them and click on each one for more details.	Pass

Welcome Screen: User opens the application	The user should be greeted by a welcome screen with a logo and the name of the app upon opening the application.	Upon opening the application, the user is greeted with a welcome screen along with the application's name and logo.	Pass
Random Notifications:	Once a notification is sent, all notifications will be random.	Each time a notification is sent, it is a randomly selected message.	Pass

Table 6.1 results of the functional requirements testing

6.4 Acceptance Testing

Acceptance testing is the final stage of the testing process and it is at this point that a user or customer can test the application as opposed to it being tested with simulated data [37]. As part of the acceptance testing for this project, participants were chosen to test the application and then asked to attend a short focus group to gauge their feelings and opinions about how well the system achieved its goals. The findings from the acceptance testing are further discussed in the evaluation section of the next chapter.

6.5 Testing the Geofences

To test whether all 22 geofences were functioning correctly it was necessary to make use of an android emulator. Once the emulator had been setup correctly, it was then possible to start the new safety application through android studio and then have it run on the emulator on the computer. This was not only very useful for helping to debug code during the development process but it also allowed for the testing of the geofences. Testing the geofences involved making use of an application that is able to provide a fake GPS location to mimic movement around a chosen area.

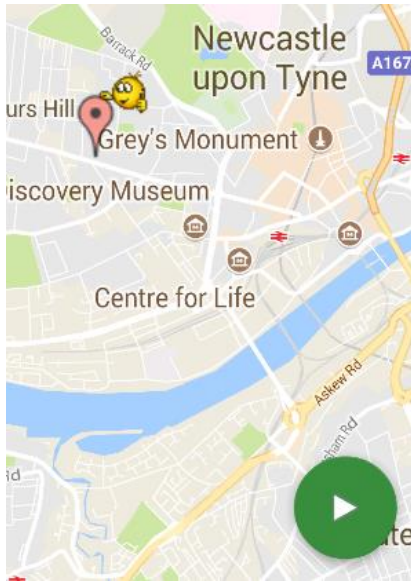


Figure 6.2 setting mock location

Figure 6.2 shows how a mock GPS location was setup. By first running the new safety application in the background and then selecting a new location and hitting the green play button as seen above, it was possible to simulate a mock location.



Figure 6.3 receiving notification

Figure 6.3 shows a notification being received once the mock location had been set within a virtual geofence boundary.

6.6 Summary of the Testing

This chapter described the processes used to test the completed system. A system integration testing approach was chosen with the use of a state transition diagram approach used to derive test cases. Following this, both functional and non-functional requirements were tested. Acceptance testing was also carried along with geofence testing with the aid of GPS mock location software. The next chapter will describe the results of the testing in more detail and offer an evaluation based on these results.

7.0 Evaluation and Results

Thanks to an agile approach to software development a working piece of software was successfully delivered by the deadline with the most essential requirements having been met. Had an alternative approach to software development been adopted then it is possible that only partially finished software would have been delivered but with little working functionality.

This chapter will first explore how time management was facilitated with the use of a Gantt chart. It will also explore some of the advantages and drawbacks of such a strategy. Next, a more in-depth account of meeting the functional and non-functional requirements are given followed by an explanation of why a focus group strategy was adopted for the system evaluation. This is then closely followed by the results of the focus group system evaluation.

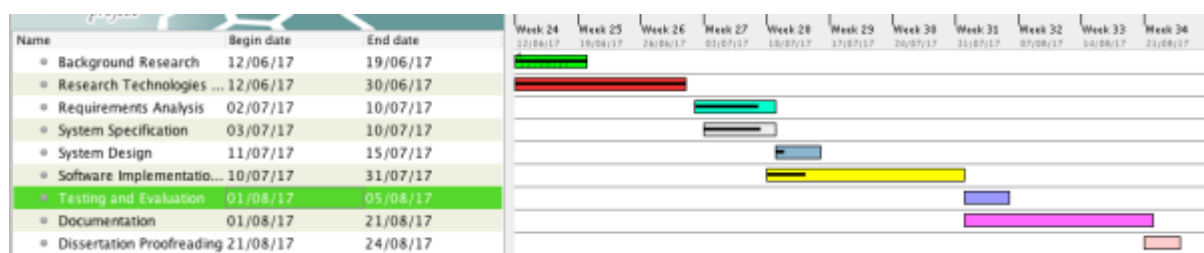


Figure 7.1 Gantt Chart

Figure 7.1 shows a Gantt chart of the planned schedule for the software project.

Time management of the project was largely planned with the use of a Gantt chart as shown in figure 7.1. The Gantt chart is a very effective tool for planning and scheduling a software engineering project allowing a team of developers to understand the tasks involved, responsibility for tasks, and how long each task should take. By utilising a Gantt chart in the early stages of the project, it was possible for the developer to clearly see the tasks that needed to be completed for the successful completion of the project. Despite the obvious advantages in transparency that the use of the Gantt chart may bring, it was found that the chart often became tedious due to having to be regularly updated. Although, it was clearly beneficial for this project, it is debatable whether Gantt charts can be truly effective in agile development environments with short development cycles where members of a team often move between tasks depending on changing requirements and priorities. For this particular project, it was found that the schedule outlined by the original Gantt chart as show in Figure 7.1 was somewhat optimistic. Although some parts of the project went according to the time scales as outlined above, other tasks went beyond by a large margin which had a detrimental impact on the software schedule as a whole. The schedule for the implementation was massively underestimated which resulted in a reorganisation of the Gantt chart.

The use of a focus group to elicit requirements was hugely beneficial in producing accurate requirements that reflected user needs. Regularly working with users and customers is a core tenet of the agile development process and greatly aided in achieving user satisfaction of the final iteration of the software product. The use of a risk plan allowed the developer to identify some of the possible risks that might endanger the punctual delivery of the final software product and thus make the necessary provisions that would avoid such an event. Examples of actions taken to mitigate serious setbacks include regularly backing up work, staying in regular

contact with the supervisor, and adopting the agile approach to development in order to be as flexible as possible to accommodate changing requirements.

The planning and design phase involved creating a number of diagrams that helped with the implementation stage of the project. The sequence diagram allowed the developer to clearly visualise the way in which events switch between objects while the design of the use case diagram further aided in the precise elicitation of the requirements.

7.1 Meeting Functional Requirements

Every effort was made to meet all the requirements, however, before the end of the project, it became evident that it would not be possible to meet all the requirements outlined in the functional requirements list. However, by combining the agile development approach to software engineering and the use of risk management techniques, it was possible to adopt an approach that meant that requirements were worked on in order of importance. As a result, by the final delivery date, it was possible to deliver a working piece of software.

- The software application successfully monitors a user's location as they ambulate across the region of Northumbria. This function works in the background and does not issue feedback based on changes regarding the user's location. However, this functional requirement was essential in being able to achieve the next requirement.
- Geofences are successfully triggered as the user's location is monitored while they move between different areas in the Northumberland area.
- The software application successfully sends a notification to the user's device once a geofence has been triggered.
- Notifications were adapted using nudge-based theories in behavioural economics to change a user's behaviour without inducing unnecessary fear of crime.
- Users are unable to turn off location tracking for the application. Although attempts were made to implement this function. Time constraints meant that requirements had to be prioritised as high, medium and low respectively. To ensure that a working software application could be delivered by the deadline it was necessary to focus attention on first completing those functional requirements higher on the functional requirement list.
- The application does not send excessive numbers of notifications to its users. The application has been designed so that only one message will be sent to the user once they enter a given geofence.
- The application does allow the user to click on the notification to be guided back to the home screen where they are able to view the same message received in the notification. However, due to insufficient understanding of the android studio environment, it was difficult to redirect the user to the precise message displayed in the notification.
- The users can scroll through a list of the notifications once the application has been opened. The user then has the option of clicking an option to view the complete nudge-based message in a full screen
- Upon turning on the application, the user is presented with welcome screen with the application's logo and name.

7.2 Meeting Non-Functional Requirements

During the development process a conscious effort was made to keep in mind the software application's non-functional requirements. As a result, most of the non-functional requirements were successfully met. The android studio environment enforces an MVVC pattern that ensures clear organisation of code so that adjustments or additional functionality may be easily implemented. According to the focus group evaluation, the new safety application was very easy to navigate with notifications that effectively changed user's perceptions of safety while out in public and thus succeeding in changing the user's behaviour to act more safely.

7.3 Focus Group for Program Evaluation

While deciding on a suitable method for measuring the effectiveness of the new software application, several approaches were considered. Initially a quantitative approach was considered with the use of surveys and Likert scales. Such an approach would yield tangible data. However, due to resource and time limitations, the sample size would be too small to make any meaningful judgments about the software application. After further consideration, despite the appealing nature of such quantitative methods, it was decided that a more qualitative approach was needed. The primary purpose of the new software application was to use nudge-based notifications to change a user's behaviour to act more safely while in public without raising fear of crime or causing anxiety. Therefore, in addition to capturing user opinions on usability, it was more important to capture whether the software application was effective at (i) conveying safety messages through nudge-based notifications and (ii) whether the nudge-based notifications led to a change in their behaviour without (iii) raising fear of crime/ anxiety?

Kontio et al. [23] explore the benefits of using focus group-based approaches to obtain user experiences and it is concluded that the use of focus groups to evaluate user experience can add value as part of a software engineering evaluation strategy. The use of focus groups may '...produce candid, sometimes insightful information, and the method is fairly inexpensive and fast to perform [23]. It was finally decided that the best approach for evaluating user experience for this particular project would be through the use of a focus group. This approach also fit conveniently with the agile approach of software development in that it would be time-efficient and continue to include users in the final stages of the software development life cycle. Additionally, the focus study would allow for the gathering of rich information in the form of feelings and thoughts regarding the effectiveness of the new safety application.

Eight students were selected from the computer science department all of which had experience with the new safety application three days prior to the hosting of the focus group. Effort was made to ensure that all participants would attend the meeting. This was done through confirmation telephone calls, social media, and text messaging. Despite all attempts to maximise the number of attending participants, only six attended on the day. Prior to the start of the focus group special effort was made to locate a room that was easily accessible to all participants. The moderator also prepared two packs of biscuits in order to keep the participants relaxed and at ease. During the focus group, the moderator aimed to

guide the conversation with open ended questions with every effort to avoid dichotomous questions. Think back – type questions were also utilised in order to maximise the expression of feelings about using the new safety application. The focus group was fairly short and lasted only 20 minutes and was largely considered a success.

7.4 Results

The results from the focus group indicated that all participants believed the application was simple and easy to navigate. Although it was criticised for being a rather simple and unattractive application, it was also praised for being an application they would happily keep running in the background.

(i) Conveying safety messages through nudge based notifications:

The general consensus regarding the messages is that they were successful in offering comprehensible advice on how to stay safe. Most agreed that the messages were easy enough to understand, however, the message asking people to get used to walking on the right-side of the road was first considered a little odd but they soon realised the value of the message. The two notifications that most people liked were the messages that suggested avoiding darker spots and staying alert. The ‘Stay Alert’ message with a picture of the bird of prey hovering over the rabbit was praised for matching the rhetorical question that followed: ‘How aware are you of your surroundings?’. The message that promoted ‘Avoiding darker spaces’ was appreciated also for its catching image of people joined together accompanied with the message ‘Most people will avoid quiet or badly lit alleyways, subways, and parks’.

(ii) Changes in behaviour

Most participants agreed that in hindsight their behaviour had changed in some respects after receiving the notifications. The most influential messages included both the ‘Stay Alert’ and ‘Avoid darker spots’ notifications. Participants reported increased awareness of potential threats. Interestingly, the message that was most

influential and led to a change in behaviour was the message about ‘avoiding shortcuts and sticking to safer routes’. Without the aid of an image the nudge was able to affect change in participants’ behaviour. While most participants agreed as to the effectiveness of the nudge, only two were able to articulate why. They agreed that the message was rather ‘accusing’ in that it implied that others were doing something to keep themselves safe while the participants were making no effort at all which in turn made them think more carefully about their own safety. The use of the availability heuristic to inflate the perception of how long people wait for buses and trains combined with framing served to inadvertently change one participant’s behaviour as they presumed that because others tend to wait a long

time for buses, he would, himself, have wait long periods of time. As a result, he often checked his phone for bus schedules on certain routes. This, without him realising, also reduced his chances of becoming a victim of crime while waiting for public transport.

The ‘Stay Alert’ nudge was also very effective in affecting changes in behaviour as one participant keenly noted, ‘When I saw the picture of a rabbit being stalked from above my first thought was to look up but after seeing the message I quickly became aware of what was going on to the side and behind me’.

(iii) Fear of Crime / Anxiety

In general, the messages did not increase anxiety or fear of crime in the participants, however, it was noted that the message ‘Avoid becoming a victim of crime’ might provoke anxiety in some people. This was interesting and is worth further investigation to see if the message can be modified. Most participants did, however, agree that as a result of the notifications, they had become more aware of their surroundings and were more likely to consider the safety of themselves and their valuables while in public. An interesting topic that came up was the implicit reference to areas with higher densities of crime. This was considered implicit as each notification, although a suggestion on how to keep safe, were only received when they had entered one of the geofences that signified higher densities of crime. The students who participated were grateful for this information and realised that even with places that that they were most familiar with, safety is something always to keep in mind.

7.5 Results Summary

The results from the focus group indicate that the application of the availability heuristic to nudge-based safety notifications are highly effective in drawing awareness to personal safety and affecting change in behaviour. Just as with the application of the availability heuristic in promoting organ donation, the application of the availability heuristic can serve to inflate perceptions of how the rest of the population act to stay safe. In doing so, this can influence individuals to take similar precautions. The use of the mere-measurement effect had a great impact on the participants. This was often described by the participants as the use of rhetorical questions, however, the true aim was to stimulate the users to think about how they might do something, for example: “How will you stay safe?”, “How aware are you of your surroundings?”, and “WILL YOU?”. By making a user think about how they will do something increases the probability of them doing that thing. This was reflected in the focus group. The study highlighted the effectiveness of using priming as an invaluable tool in the choice architect’s nudge repertoire with participants agreeing to the effectiveness of images that reflected the notification’s written suggestions. Framing also worked to effectively anchor a participant’s expectations of how long they would have to wait for a bus. This was achieved by

first stating that most people are dissatisfied with how long they have to wait for public transport. This served to anchor the participant's expectations of how long the user can expect to wait for transport. The results show how the effectiveness of a single nudge-based safety notification can be maximised through the application of a combination of framing, availability heuristics, priming, anchor and adjustment, and mere-measurement effect techniques.

The success of nudge-based techniques, as the focus study suggests, is also dependent on the simplicity of the message. For example, the traffic nudge was seen as ambiguous and required some thought as to its relevance. As a result, the message, although utilising similar techniques proved unpopular.

Although the participants expressed no explicit reference to increased anxiety and fear of crime, there was a feeling, during the focus group, of treading a fine line between increasing awareness of personal safety and increasing anxiety and fear of crime among the participants.

Participants were generally happy with the new safety application and were impressed by the idea of using nudge-based messages to reduce the chances of its users becoming a victim of crime based on their present location. The application's simplicity was also appreciated, however, it was later suggested that the application could be improved by including a map with the geofences marked clearly for the user's convenience.

8.0 Conclusion

This paper is a detailed description of the processes involved in the building of a software application that aimed to influence users' behaviour so that they would act more safely while in public. It was hoped that this could be achieved with the use of nudge-based notifications based on a user's given location.

First, a broad introduction described the recent proliferation of technologies adopted to fight crime and the politics of the concept of fear of crime. A case was then put forward for more to be done in the context of crime prevention and increasing personal safety while in public.

Second, in-depth research was carried out into behavioural economics and other key technologies. It was at this point that nudges were further explored as a potential strategy for implicitly influencing user behaviour without increasing anxiety or fear of crime. Java was confirmed as the language of choice for the new software application's development along with android as the development environment. Geolocation and geofences were found to be perfect for tracking user location and defining virtual boundaries demarcating areas of high density crime. To discover ways in which large datasets can be processed and categorised in order to extrapolate meaningful information, data mining was explored. It was discovered that cluster algorithms would be a potential solution to sorting and organizing the crime dataset. Background research concluded with the review of a similar safety application, the SafeZone App. After careful consideration, the strengths and weaknesses of the application was considered and it was decided that a new application with city-wide reach and a focus on crime prevention could add value to existing applications. The successful completion of the background research concluded the feasibility study and the project was ready to move onto the next stage.

Before the building of the application could take place, it was first necessary to develop a plan of what the software application would look like and what it would do. This was achieved during the design stage of the software development life cycle. At this point functional and non-functional requirements were formally listed and carefully considered that helped in creating a hierarchy of requirements. Risks were considered and possible solutions were devised to reduce the impact of possible impediments to the new software application's successful completion. As part of the design process, it was decided that a development methodology that promoted flexibility, requirement adaptability, productivity and quick turn-around was required. For these reasons an agile approach to software development was adopted. Before moving onto the implementation of the software application diagrams were designed to further aid as a guide to the implementation of the actual software application.

Implementation was largely successful in that a working product was released before the final delivery date, however, due to time constraints some of the medium to low requirements remained unimplemented. These included the implementation of a switch to disable location tracking for the new safety application and the ability to move from notification directly to the relevant notification. A black box approach was adopted to the testing phase of the life cycle and it was found that the software application's core functional and non-functional requirements had been met.

It was during the evaluation stage that the project's overall performance was considered. Time management through the use of a Gantt chart was evaluated and further considerations into the

extent to which requirements had been met were made. Both quantitative and qualitative approaches to the user evaluation of the program were considered. It was decided that due to the unique objectives of the software application, the best approach would be a qualitative one.

The qualitative approach to user experience evaluation took the form of a focus group in which participants were asked about their experiences of using the software. The focus group took the form of an unstructured interview with open questions. The main objectives of this focus group were to gather feedback on the comprehensibility of the nudge-based notifications, whether the nudge-based notifications led to a change in user behaviour without raising fear of crime/ anxiety? In addition, it was important to learn how the users' felt about using the system.

It was found that the availability heuristic could be used to effectively inflate users' perceptions of the way other people act and subsequently change user's behaviour to make them behave more safely while in public. The effect of the availability heuristic could be magnified by applying the mere-measurement effect. By asking rhetorical-like questions that serve to make the user think about how they will carry out a certain action such as keeping safe, it is possible to further increase the chances of the user carrying out that given action.

Priming with the use of images proved particularly effective in affecting behaviour change. The effects of priming can also be magnified when it is combined with another nudging technique such as the mere-measurement effect. Framing was also a powerful tool used to affect positive change in behaviour. It was found that in order to maximise the effectiveness of nudge-based notifications a combination of two or more nudges could be used to complement each other. However, the effectiveness of a potential nudge may be negatively influenced by its complexity. The more complex a nudge is to understand, the less effective it will be. Trying to influence increased personal safety among users without raising the fear of crime is a constant struggle that requires careful manoeuvre and balance.

Through the adoption of an agile development approach, the developer has been able to successfully implement a system that can track its users and notify them if they enter an area of higher density crime. The new software application sends nudge-based notifications that attempt to implicitly influence users to behave more safely while walking in public. Although some functional requirements have not been met, by categorizing functional requirements in order of priority combined with an agile development approach, the developer has been able to successfully implement a working product with essential core features.

The new safety software application has much room for improvement and growth. Through further cycles and iterations of the agile development process, it is certain that all requirements can be met. However, improvements to the overall system might include using a premium data mining package such as Spark. Similar technologies would allow better seamless integration between the data mining package, databases, and the android application. Furthermore, alternative cluster algorithms might be tested against the K-means algorithm for better efficiency and grouping of data. A potential alternative to the K-means algorithm is the density-based spatial clustering of applications with noise (DBSCAN) algorithm. Further studies might be carried out into how to better design nudges to better influence safer behaviour while avoiding increasing fear of crime users.

References

- [1] I. 29148:2011 and ©, 'INTERNATIONAL STANDARD ISO / IEC / IEEE Systems and software engineering — agile environment', *ISO/IEC/IEEE 26515 First Ed. 2011-12-01; Corrected version 2012-03-15*, vol. 2012, 2011.
- [2] B. Arief, D. Greathead, L. Coventry, and A. van Moorsel, 'Towards the Implementation of an Internet-based Neighbourhood Watch Scheme', *Kent Acad. Repos.*, 2011.
- [4] Y. Baba, M. Austin, 'Neighborhood Environmental Satisfaction, Victimization, and Social Participation as Determinants of Perceived Neighborhood Safety', *Environment and Behavior.*, 1989.
- [5] V. Bajpai and R. P. Gorthi, 'On Non-Functional Requirements : A Survey', pp. 0–3, 2012.
- [6] Behaviour Insight Team, 'Applying behavioural insight to health', *Heal. San Fr.*, vol. 1, no. 21 January 2011, pp. 9–12, 2010.
- [7] B. W. B. Boehm, 'Software risk management: principles and practices', *IEEE Softw.*, vol. 8, no. 1, pp. 32–41, 1991.
- [8] Cabinet Office Behavioural Insights Team, 'Applying Behavioural Insights to Organ Donation: preliminary results from a randomised controlled trial', pp. 1–11, 2013.
- [9] J. Byrne and G. T. Marx, 'Technological innovations in crime prevention and policing. A review of the research on implementation and impact', *J. Police Stud.*, vol. 3, no. 20, pp. 17–40, 2011.
- [10] Cabinet Office Behavioural Insights Team, 'Applying Behavioural Insights to Organ Donation: preliminary results from a randomised controlled trial', pp. 1–11, 2013.
- [12] Department for Business Innovation & Skills, 'Better Choices : Better Deals', *Innovation*, p. 22, 2011.
- [14] V. Faber, 'Clustering and the Continuous k-Means Algorithm', *Los Alamos Sci.*, no. 22, pp. 138–144, 1994.
- [15] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, 'From Data Mining to Knowledge Discovery in Databases', *AI Mag.*, vol. 17, no. 3, p. 37, 1996.
- [16] U. Garg and A. Singhal, 'Software Requirement Prioritization based on Non Functional Requirements', *IEEE Softw.*, vol. 6, pp. 793–797, 2017.
- [17] J. Garofalo, 'the Fear of Crime: Causes and Consequences', *J. Crim. Law Criminol. J. Crim. Law Criminol. J. Crim. LAW Criminol.*, vol. 72234181, no. 2, pp. 839–857, 1973.
- [18] M. Gladwell, 'Blink: The power of thinking without thinking', Back Bay Books, 2011.

- [21] D. Jeske, L. Coventry, and P. Briggs, ‘Nudging Whom How: IT Proficiency, Impulse Control and Secure Behaviour’, *CHI Work. Pers. Behav. Chang. Technol. 2014*, 2013.
- [22] D. Kahneman, ‘Thinking , Fast and Slow (Abstract)’, *Book*, p. 512, 2011.
- [23] J. Kontio, L. Lehtola, and J. Bragge, ‘Using the focus group method in software engineering: obtaining practitioner and user experiences’, *2004 Int. Symp. Empir. Softw. Eng. ISESE 2004*, pp. 271–280, 2004.
- [24] D. Mairiza, D. Zowghi, and N. Nurmuliani, ‘An investigation into the notion of non-functional requirements’, *Proc. 2010 ACM Symp. Appl. Comput. - SAC '10*, vol. 1, no. 1, p. 311, 2010.
- [25] P. K. Manning, ‘Technology’s Ways: Information technology, crime analysis and the rationalizing of policing’, *Crim. Justice*, vol. 1, no. 1, pp. 83–103, 2001.
- [26] A. Moran, *Agile Risk Management*, no. March. Springer, 2014.
- [27] S. Na, L. Xumin, and G. Yong, ‘Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm’, *2010 Third Int. Symp. Intell. Inf. Technol. Secur. Informatics*, pp. 63–67, 2010.
- [30] R. Pressman, Roger S., Maxim, Bruce, *Software Engineering: A practitioner’s Approach*, 8th ed. New York: Mc GrawHill Education, 2015.
- [33] R. Rennekamp and M. Nall, ‘Using focus groups in program development and evaluation’, [http://www. ca. uky. edu/AgPSD/Focus. pdf](http://www.ca.uky.edu/AgPSD/Focus.pdf), 2006.
- [34] P. Rossetti, T. Dinisman, and A. Moroz, ‘INSIGHT REPORT AN EASY TARGET ? rates for violent crime and theft’, *Vict. Support*, no. April, 2016.
- [35] A. Roxin, J. Gaber, and M. Wack, ‘Survey of Wireless Geolocation Techniques’, 2007.
- [36] M. Seidl, M. Scholz, C. Huemer, and G. Kappel, *UML @ Classroom: An Introduction to Object-Oriented Modeling*. 2015.
- [37] I. Sommerville, *Software Engineering*, 9th ed. Boston: Pearson Education, 2010.
- [39] C. O. B. I. Team, ‘Applying behavioural insights to charitable giving’, pp. 1–25, 2013.
- [40] R. H. Thaler and C. R. Sunstein, *Nudge: Improving decisions about health, wealth, and hapiness*. London: Penguin Books, 2008.
- [41] J. Turland, L. Coventry, D. Jeske, P. Briggs, and A. van Moorsel, ‘Nudging Towards security: Developing an Application for Wireless Network Selection for Android Phones’, *Proc. 2015 Br. HCI Conf. - Br. HCI '15*, pp. 193–201, 2015.
- [42] UK Cabinet Office (Behavioral Insights Team), ‘Applying Behavioral Insights to reduce Fraud, Error and Debt’, *Cabinet Off. London*, vol. 185, p. 38, 2012.

- [43] F. Wagner, R. Schmuki, T. Wagner, and P. Wolstenholme, *Modeling Software with Finite State Machines A Practical Approach*. 2006.
- [44] D. Weisburd and J. E. Eck, ‘What Can Police Do to Reduce Crime, Disorder, and Fear?’, *Ann. Am. Acad. Pol. Soc. Sci.*, vol. 593, no. 1, pp. 42–65, 2004.
- [45] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Cambridge: Morgan Kaufman, 2011.
- [46] I. Yevseyeva *et al.*, ‘Consumerisation of IT: Mitigating Risky User Actions and Improving Productivity with Nudging’, *Procedia Technol.*, vol. 16, pp. 508–517, 2014.

Websites

- [3] Budget cuts will radically change policing-Association of Police and Crime Commissioners
http://www.apccs.police.uk/latest_news/budget-cuts-will-radically-changepolicing/?cookie_dismiss=true, June 2017
- [11] Data Downloads-Data.Police.uk
<https://data.police.uk/data/>, July 2017
- [13] Location Strategies- Developer
<https://developer.android.com/guide/topics/location/strategies.html>, July 2017
- [19] Googlesamples / android-play-location
<https://github.com/googlesamples/android-play-location/tree/master/Geofencing>, July 2017
- [20] What is State Transition Testing in Software Testing- ISTQB exam certification
<http://istqbexamcertification.com/what-is-state-transition-testing-in-software-testing/>, July 2017
- [28] Personal Safety- Northumbria Police
https://www.northumbria.police.uk/advice_and_information/crime_prevention/personal_safety/, July 2017
- [29] Crime in England and Wales-Office for National Statistics
<https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/bulletins/crimeinenglandandwales/yearendingdec2016>, June 2017
- [31] What’s new in JDK 8? – Oracle
<http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html>, July 2017

- [32] Interface Prepared Statement- Oracle Documentation
<https://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html>, July 2017
- [38] Personal Safety Information for Students- Student Progress
<http://www.ncl.ac.uk/students/progress/student-resources/community/safety.htm>, July 2017

Appendix A

Unstructured Interview Questions Used in the Focus Group

- What did you like best about the new safety application?
- What did you think about the actual safety messages?
- What were your favourite messages?
- Did you understand all of the messages?
- What did you think about the pictures?
- How did the application make you change your behaviour?
- Think back to how you felt when you received the notifications.

