University of Surrey

Faculty of Engineering and Physical Sciences

Bsc Computer Science (Hons)

COM3001: Final Year Project Report 2019/2020

Virtual Sales Assistant

Dylan Parker

6467850

Dp00405

Supervisor: Mark Manulis

Declaration of Originality

All work, contained within this report, and submitted alongside it is my own. All publishers and authors of any material that is not my own have been credited and clearly identified, regardless of their state of publication. IEEE referencing has been used, as requested within the Student handbook, to identify the owners of any material. I agree to the submission of my work through the Universities plagiarism detection service Turnitin® UK to authenticate my work as legitimate, and not forged. I acknowledge that any work that is deemed illegitimate may incur penalty.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Mark Manulis for all his suggestions and advice given to me throughout this last year, as well as Dr. Brijesh Dongol for monitoring and guiding our progress as the year has passed. I would also like to give thanks to my family and friends for supporting me, and providing advice, through the stressful times I have encountered during my journey at university.

Abstract

Eighty-seven percent of customers that receive bad service do not return to the business. My project focuses on two objectives; communication, and security, allowing for faster transaction times and greater customer service. This is achieved through implementation of an application that allows customers to collect and purchase secured items without the need of a sales colleague. This reduces the load on employees and allows them to focus on creating a better environment for the customers, whilst remaining open if questions do arise.

Within businesses, there are four key regions of security that must be monitored and maintained to ensure minimal occurrence of a malicious attack. These regions are; money, theft, premises, and cyber. My project aims to mitigate the risks placed onto a business by these four regions, and allow them some peace of mind when operating their stores. Two companies, Sainsburys and Tescos, have implemented systems that provide similar features to the application I intend to develop. Tesco's implementation introduces in-store scanners that, through use of a clubcard, can be retrieved and used to scan items around the store before purchasing them at a designated self-service till. Sainsbury's SmartShop application transfers this experience onto the customers mobile phone, utilising the camera to scan barcodes and add them to a virtual basket. This can then be transferred to the till by scanning the barcode found on the top the tills display. Some of the restrictions involved with these applications include; clubcard or nectar-card registration, internet connection, and inability to purchase tagged products without assistance. My application intends to avoid these issues through allowing users without accounts, access to the application's core functionality, no requirement for internet connection, and a remotely unlockable tag through deactivation at the till. payment, online accounts, full business compatibility, and multiple authentication methods all fall outside of the project scope and thus will not be included in the final application.

Within my application, customers can add items to their basket through use of QR or NFC, and transfer their basket to the self-service tills via Bluetooth or QR. This basket is then verified with a database to ensure all the products are legitimate, and if a positive response is received, the items are unlocked. The GUI of the application is based on two main parameters; the reachability of interactive elements, and the expected locations of those interactive elements. Shopping applications frequently put the importance of the expectations of locations over reachability, due to the restriction on screen estate, rather than the disregard of the user's reach.

The waterfall model was used when organising all regions of this project, allowing the optimisation of the time spent, and a guarantee of completion. This was especially important, as during my project preparation, I was able to identify Android Studio and XAMPP as the two main programs I would use, due to my existing understanding of them, and the wide community the programs have. The application performed as expected for all requirements and achieved each objective and success criteria set out. Future work includes; hashing of passwords and digital signatures.

Table of Contents

Declaration of Originality **1**

Acknowledgements **2**

Abstract **3**

Table of Contents **4**

Table of Figures **9**

Table of Tables **11**

Abbreviations **12**

1 Introduction **13**

1.1 Project Motivation 13

1.2 Aims and Objectives 13

1.3 Benefit To Society 14

1.4 Success Criteria 14

1.5 Project Structure 14

2 Literature Review **16**

2.1 Key Technologies 16

2.1.1 NFC 16

2.1.2 QR 16

2.1.3 Bluetooth 17

2.2 Security Threats and Countermeasures 17

2.2.1 Premises 17

2.2.2 Money 17

2.2.3 Theft 18

2.2.4 Cyber 18

2.3 Existing Solutions 18

2.3.1 In Store Handheld 18

2.3.2 Sainsburys SmartShop 20

2.3.3 Tesco Pay+ 21

2.4 Software development life cycle 21

3 Application Analysis **23**
3.1 Existing Solution 23
3.1.1 Limitations and Drawbacks 23
3.1.2 Potential Attacks 23
3.2 Proposed Solution 24
3.3 Requirements 25
3.3.1 Functional Requirements 25
3.3.2 Non-Functional Requirements 29
3.3.3 Project Scope 30
3.4 Specifications 31
3.4.1 Software 31
3.4.2 Hardware 31
4 Application Design **32**
4.1 Use case diagram 32
4.1.1 Phone 32
4.1.2 Till 33
4.2 System architecture 33
4.2.1 Real-World Implementation 33
4.2.2 Small-Scale Implementation 34
4.3 Graphic User Interface 35
4.3.1 Considerations 35
4.3.2 Phone Drafts 37
4.3.3 Phone Draft Reachability 38
4.3.4 Phone Final Result 39
4.3.5 Till Drafts 41
4.3.6 Till Final Result 41
4.4 API Design 42
4.5 Entity relationship diagram 43
5 Application Implementation **45**
5.1 Chosen software methodology 45

5.2 Prototype 45
5.3 Software and Language Justification 45
5.4 Methods 46
5.4.1 Basket 46
5.4.2 Till 47
5.4.3 Scanning a Product 47
5.4.4 Adding a product to the store catalogue 48
5.4.5 Transfering Basket 49
5.4.6 Basket Confirm 50
5.4.7 Toolbar and Bottom Navigation Bar 50
5.4.8 Accounts 51
5.4.9 Website 52
5.4.10 Settings 52
5.4.11 Tutorials 53
5.4.12 Find Nearby Store 53
5.4.13 Bookmarks 53
5.4.14 Contact The Developer 53
6 Testing and Integration **54**
6.1 Requirements Testing 54
6.2 Cross-Device Testing 77
6.3 User Acceptance Testing 77
7 Statement of Ethics **80**
7.1 Real-World Implementation 80
7.1.1 Legal Considerations 80
7.1.2 Ethical Considerations 80
7.1.3 Social Considerations 81
7.1.4 Professional Considerations 81
7.2 Small-Scale Implementation 81
8 Project Conclusion **82**
8.1 Project Evaluation 82

8.2 Improvements and Future Work 83
8.3 Final Statement 84
References **85**
Appendices **88**

Table of Figures

Figure Number	Description	Page(s)
1	Scanning an item with the in-store scanner	18
2	Placing the scanned items into your basket	18
3	Scanning the barcode on the till to transfer your basket	19
4	Sainsburys SmartShop Application	19
5	Transfering your details and paying through use of QR	19
6	Illustration of the Software Development Life Cycle	19
7	UML Diagram of all functions performable on the applications phone-side	31
8	UML Diagram of all functions performable on the applications till-side	32
9	Diagram of the real-world infrastructure implementation	33
10	Diagram of the small-scale infrastructure implementation	33
11	Reachability of a phones screen based on hand(s) used	34
12	Percentage of One-Handed, Two-Handed, and Cradled phone users	35
13	JD Sports application screenshots	35
14	Argos application screenshots	35
15	Currys PC World application screenshots	36
16	Drafts of each page, and pop-up, of the applications phone-side	37
17	Reachability of three key draft pages for the applications phone-side	38
18	Final results of each page, and pop-up, of the applications phone-side	39
19	Drafts of each page, and pop-up, of the applications till-side	40
20	Final results of each page, and pop-up, of the applications till-side	40,41
21	Entity relationship diagram of the Products and Items tables found in the database	42

22	Entity relationship diagram of the Accounts and Bookmarks tables found in the database	43
23	Visual interpretation of the waterfall model software methodology	44
24	Screenshots of the application on screens of different dimensions	76
25	Screenshots of the ‘Adding Item to the Basket’ tutorial	77
26	Screenshots of the ‘Deleting an Item, or Visiting its Webpage’ tutorial	77
27	Screenshots of the ‘Paying for your Basket’ tutorial	78

Table of Tables

Table Number	Description	Page(s)
1	All definitions for abbreviations found within the report.	11
2	A breakdown of each section of the project.	13,14
3	All the desired functions that are needed to be implemented.	25-28
4	Descriptions on further requirements of the application and the regions they are applicable in.	28,29
5	Actions required by the database for specific users and the parameters sent and received.	41,42
6	Titles for each Software and Programming Language used within the project.	44,45
7	Descriptions of each requirement, how they operate and the results achieved.	53-75
8	The Objectives set out at the beginning of the report and the results, whether achieved or failed to achieve.	81

Abbreviations

Table 1 - all definitions for abbreviations found within the report.

Abbreviation	Definition
UK	United Kingdom
NFC	Near-field Communication
QR	Quick Response
RFID	Radio Frequency Identification
OS	Operating System
EAN	European Article Number
ASCII	American Standard Code for Information Interchange
URL	Uniform Resource Locator
UV	Ultra-Violet
CCTV	Closed Circuit Television
GDPR	General Data Protection Regulation
WBS	Work Breakdown Structure
GUI	Graphic User Interface
API	Application Programming Interface
IDE	Integrated Development Environment
MIME	Multipurpose Internet Mail Extensions
VSA	Virtual Sales Assistant

1 Introduction

1.1 Project Motivation

Working in the world of retail has had a huge impact on the direction I have chosen to take my project. Eighty-Seven percent of customers that receive a negative experience do not return to the business. Furthermore, a customer is four times more likely to defect to a competitor if the issue is service related, rather than product or price related [1]. A large amount of customer complaints received within retail revolve around colleague behaviour and long transaction times, this occurs due to a limited number of staff in the store at one time. The project I have designed was primarily created in an attempt to tackle these two key issues whilst having the added benefit of increasing product security.

NFC is commonplace in transaction processes throughout the high street whereas QR codes are not normally seen on products. I intend to use both of these technologies to allow customers to purchase goods and remotely detach tags from the products, through the use of a self-serve till. This will remove the need for assistance from employees during the transaction process and allow customers to obtain and purchase the goods they need by themselves quickly and efficiently.

Retail stores may still desire interaction between customers and their employees. This project does not intend on removing this entirely, as it can still occur at an earlier stage in the shopping journey, preferable when the customer is browsing for their desired products. The project has been designed to assist the staff by reducing workload at the checkout stage.

1.2 Aims and Objectives

The aim of this project is to research and develop a security system that allows consumers to purchase expensive, secured/tagged products without the need of assistance from a sales colleague or other representative, which a classical store structure and security system would require. The fundamental objective of the system is to improve customer satisfaction and reduce transaction time whilst maintaining the standard of security.

The following are the objectives for the project:

- Explore and understand security threats and countermeasures that are used in a retail transaction system.
- Discover and review security threats and countermeasures found within a standard retail store.
- Review literature relevant to NFC, QR, and Bluetooth
- Design, implement and test a program which allows for processing, purchase and removal of product security tags (without the use of conventional physical removal systems).
- Evaluate the usability and quality of the new transaction system and its security.

- Recommend potential improvements that can be made to future proof my project and further improve retail store security.

1.3 Benefit To Society

This project, and the developed application will help streamline the transaction process for customers purchasing in a wide variety of stores. The program gives customers complete control of their journey, whilst maintaining the level of security of the store. This also allows existing colleagues to be more focused on the upkeep of the store, ensuring it is held to a high standard and represented well. In the event of a customer needing assistance with a product they need not look any further than the application. The app can redirect them to the web page of any product they are currently viewing and give them more information. Businesses also benefit from the application; it reduces the strain on their business as customers can swiftly enter, find their product, and pay.

1.4 Success Criteria

The following criteria, if met, will determine whether the project is deemed a success. Upon completion of the systems development these criteria will be used to evaluate the progress and potential success of the project.

- The designed system meets the aim and all relevant objectives listed above.
- The system is easy and simple to use.

1.5 Project Structure

Table 2 - A breakdown of each section of the project.

Section	**Description**
Introduction	An introduction to the project on what I intend to develop, the motivation behind the chosen project, and the technologies I am planning on using. The primary aim of the project, objectives and success criteria are discussed as well.
Literature Review	A review and deep insight into the technologies I am using, and how they are incorporated into store infrastructure already. I also touch upon the current store transaction journey and the security measures found in stores today. The Software Development Cycle is documented here.
Application Analysis	An outline and full engagement of the requirements, both functional and non-functional, of the project. It includes further description of the existing solutions that have been implemented by businesses and their limitations and drawbacks. How the solution I have generated not only recreates this but also how it's expanded upon and resolves these issues is also discussed.
Application Design	Contains all drafts and final developments of design that was

	generated when working on the project. Furthermore, a detailed illustration of the system architecture, database structure, and coding structure.
Application Implementation	The implementation of major/key developments within the application have been documented here. Importantly, the chosen software methodology is also discussed.
Testing and Integration	A comprehensive detailing of the tests undertaken to ensure the application meets the requirements specified within section 3.3.1 and 3.3.2. Each test displays the step-by-step process taken to achieve the desired outcome. Some tests exist only to break the application whilst others show function. User Acceptance testing and Cross-Device testing are also examined here.
Statement of Ethics	The discussion and review of the ethics that the project should adhere to when governing the protection and use of information.
Project Conclusion	The key aims and developments announced by this report. Contains all desired future work and potential improvements for the project, alongside the outcomes of the report as described by the objectives and success criteria.

2 Literature Review

2.1 Key Technologies

2.1.1 NFC

NFC is an inexpensive and embeddable wireless data transfer method that works in close proximity with other devices in order to communicate and has no need for an internet connection. In order to be used an NFC chip requires activation from a second chip, which happens remotely. Upon activation a small amount of information can be sent [2]. Examples of devices that contain these chips are; smart phones, card readers and parking metres [3].

NFC can be used in three forms depending on the situation in which they are featured; card emulation, reader/writer and peer-to-peer. Card emulation revolves around NFC-enabled devices being able to act as multiple smart cards and perform features like payments or ticketing. Reader/writer allows NFC-enabled devices to read information from an NFC tag and follow instructions labelled within the tag. These tags usually only have a few instructions to follow given their small storage restrictions. When multiple tags are encountered an algorithm determines which tag takes precedence. Finally, Peer-to-Peer enables communication and information exchange between two NFC-enabled devices. Before a connection is allowed, the parameters for communication are defined [4].

NFC evolved from RFID technology, which uses radio frequencies to send information. RFID is read only and thus data sent can not be critiqued nor returned in the event of an error. This has been improved upon with NFC, which allows for two way communication [5]. NFC is identifiable to consumers throughout the course of the customer journey, and is utilised typically during the payment phase. Apple Pay, Android Pay and Samsung Pay are all powered by NFC emulation [6].

2.1.2 QR

A QR code is a matrix barcode; it is read through a recording or picture device, for example a phone camera, using a QR code reader. QR code readers are products that are provided with current day mobile OS, whereas an older OS may require it to be installed from their respective stores free of charge. It is constructed of a set of black squares on a square grid. QR codes store up to several hundred times more information and have faster readability than their EAN barcode counterparts. It is also possible to restore up to thirty percent of data that could be lost within a QR code due to dirt, damage or obstruction [7].

QR codes use four standardised encoding modes; numeric, alphanumeric, binary and kanji. Numeric converts a list of given integers into a QR code. Alphanumeric is restricted to specific types of ASCII characters including; uppercase letters, digits, horizontal space and eight punctuation characters. Although it has its limitations, it is still possible to encode a URL within the QR code. The QR code generated using the

binary method stores pure binary and is the least efficient method to store data within a QR code [8].

2.1.3 Bluetooth

Bluetooth is a wireless radio-wave technology that works over seventy nine different channels. Primarily on 2.45 GHz. It allows communication and data transfer between two compatible devices over short distances, typically less than 10 metres. This is achieved through the creation of a short range network and allows up to eight connections at one time [9].

When two devices attempt to communicate, one of the seventy nine channels is chosen at random and used to transfer data. If this channel is occupied, a new channel is chosen at random. This is known as spread-spectrum frequency hopping and can occur thousands of times a second to ensure a clear channel of communication [10]. If more than two devices are present, a piconet is formed; devices can join or leave a piconet at any time. One device, known as the master, has control over the networks. All other devices are known as slaves and follow the instructions of the master.

2.2 Security Threats and Countermeasures

Security measures exist that do not directly affect or influence the sales journey. These are required by retail stores to prevent theft and protect their goods. This includes; automated tag unlocking upon purchase verification, and complete removal of cash requirements in the transaction process; reducing the likelihood of burglaries. It is assumed the other measures are implemented within the store infrastructure already, given that it would be required for the store to function without crime. It is important to discuss all security measures to further grasp the importance of a security infrastructure within a store environment. There are four key areas of security that require protection in order to maintain and limit the damage to the contents of the store or the store itself. These areas are; premises, money, theft, and cyber [11].

2.2.1 Premises

It is important that the retail store remains secure when both open and closed to customers. Usually, criminals interested in stealing large quantities of high value goods will attempt to steal these during the store's closed hours. There are measures in place to prevent theft of the store from happening, these include; alarms, security lighting, and shutters [12]. At a time where the store is open it is not possible to obtain such a large value of goods without coming into contact with a member of the company or a customer. Names on visitor and employee badges, alongside visitors being accompanied at all times reduce the likelihood of theft. This reduces the likelihood of impersonation and allows trusted members to be identified easily.

2.2.2 Money

Money, physical and electronic, is one of the most important assets to a successful company regardless of the industry it resides in. It is also one of the few assets that are handled by the customers before the company obains them. This makes any money that enters the company susceptible to counterfeiting, damage or human error [13]. There

are many ways to deal with each and it is important for a retail company to instruct its employees, given they are handling the money, on how to handle the assets and what to look for to prevent any malicious activity.

The potential for obtaining money that has been counterfeited, damaged, or miscounted, can be reduced through implementation of regulations or technology, which can limit or entirely prevent them from being accepted. Some examples of ways to prevent counterfeiting and damage are; eye inspection, using a UV light emitter, or markers. Human error can allow the incorrect sum of money to be taken or acceptance of a tender that is not valid. Through double counting and training, this can be limited. Regular money counts should also occur on each till to ensure that the value of money accepted is equal to the total value of products sold, through the till [14]. In the event that theft or miscount of money does occur, CCTV can then be utilised to identify the individuals responsible, given it is within the period that the CCTV footage is stored. This usually is 7 days or greater [12].

2.2.3 Theft

Theft can be prevented through storing expensive products within locked cabinets or within a storage space that is not accessible to customers, i.e. a warehouse. Tags and labels also prevent products from being removed from the store without deactivation [15]. In the event an item is stolen it may go unnoticed for an extended period of time; regular audits will allow identification of potential product theft or misplacement.

2.2.4 Cyber

Cyber Security is typically the most underrated and underinvested infrastructure within businesses. Most companies will not go further than what is expected of them by the GDPR regulations when handling customer data, and although this in itself is acceptable, the quality of the implementation is usually lacklustre and overlooked. The most common infrastructure for prevention of security is the use of usernames and passwords, alongside restrictions on the sites you can access using the terminal's web browser. This prevents customers from easily accessing information through direct use of a terminal. Furthermore, servers and the store intranet will include a firewall for sorting of allowed and disallowed access. This is also where you are likely to find end-to-end encryption when attempting to store or read information you would not like accessed by a third party [12].

2.3 Existing Solutions

The following are some existing applications or systems that are implemented in stores across the United Kingdom. Each has gone through a deep analysis to understand their benefits and considerations.

2.3.1 In Store Handheld

All large modern-day supermarkets contain self-serve tills for an easier transaction process for those customers with limited items to purchase [16]. However, some supermarket brands have deployed a handheld scanner which is used to reduce time of purchase. Seen at the end of the shopping journey. They allow customers to routinely

add items to an electronic basket which is then ported to the self-serve tills upon arrival. These types of systems are thought to make consumers' lives easier but are not typically used and require infrastructure implementation. Multiple stores have their own renditions of in-store scanners; Tesco's 'Scan as you shop' is one of the many examples of these systems being implemented.

Tesco's implementation of this type of system, it resolves some of the issues presented by a traditional system. One main restriction of the system is the requirement of owning a Tesco Clubcard, which is to be scanned to collect points and to identify the user, alongside showing the scanner you will be allowed to use. The three steps Tesco illustrates when using these scanners is as follows:

1. Scan - The handheld scanner is used to scan any products being placed into the basket (Figure 1). Products can be removed by accessing the user interface. Any products, excluding some singular or unpackaged fruit and vegetables, that do not have a barcode on them require weighing before a ticket can be printed and scanned.

Figure 1 - Scanning an item with the in-store scanner [17]

2. Pack - All products can be placed into bags, with no need to remove them at the end of the sale. This allows for greater organisation and prioritisation of goods (Figure 2).

Figure 2 - Placing the scanned items into your basket [17]

3. Pay - Scan the barcode at the top of the till and your basket is transferred to the till (Figure 3). Scan your coupons, and then pay. Your first use of the in-store scanners will require an audit to ensure no difficulty when adding or removing items from the virtual basket. Further audits may occur at random intervals.

Figure 3 - Scanning the barcode on the till to transfer your basket [17]

2.3.2 Sainsburys SmartShop

Utilises your phone to both scan and purchase products, without the need for a handheld device in store. Although this option is also available. A Nectar card is required to use any handset that is in store [18]. This method allows you to skip the queues of the current method of payment and instead use seperate, designated tills that are only usable by SmartShop users. Due to the testing nature of this new system, it is only available in a few stores in the United Kingdom.

The process of purchase of the SmartShop application is almost identical to Tesco's in-store scanners, using your phone's camera instead of a barcode scanner. The application also allows for a small amount of personalisation and customisation for the user, for example a shopping basket which can be checked and altered as your shop proceeds. Similarly with Tesco's In Store Scanners, any products that do not have barcodes on them will also have a barcode on the ticket of the product. Furthermore, the application allows for collection of nectar cards and redemption of coupons (Figure 4). One of the limitations of the system is the restriction of certain products; tobacco, lottery tickets, gift cards, postage stamps, mobile phone top-ups or any PayPoint products. These restrictions are in place due to age restrictions on these items. The restrictions can be overturned through a separate purchase or authorisation at the till [18].

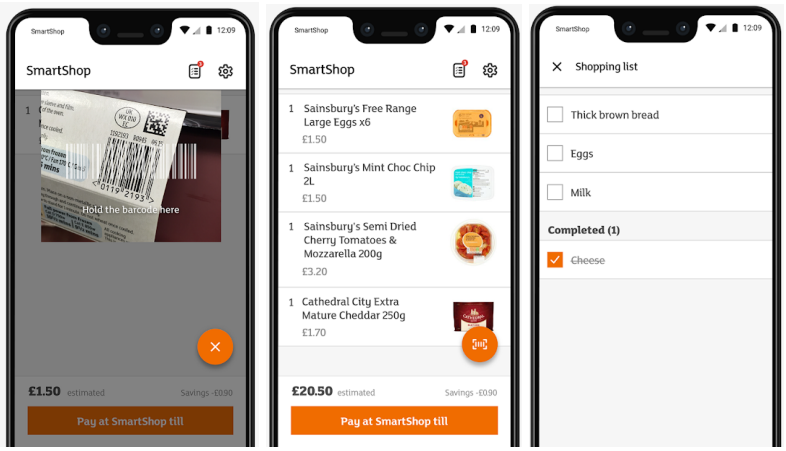


Figure 4 - Sainsburys SmartShop Application [19]

2.3.3 Tesco Pay+

Tesco Pay+ does not have functionality that allows it to process products, however, it does allow for a simpler transaction process. An avid Tesco shopper may consider this application due to its involvement with; tesco clubcard points, spending tracking, and paying from the app - both online and offline [20].

The lack of a requirement for internet access is achieved through use of a QR Code at the till when proceeding with the payment process (Figure 5). The Application will automatically add any clubcard points that you are owed, alongside paying for your basket, all in one motion.

Figure 5 - Transfering your details and paying through use of QR [20]

2.4 Software development life cycle

The software development cycle is a commonly used software development methodology. It is known for producing high quality software at low prices. Similar to other popular methodologies, it involves distinct and direct modules [21]. These modules are tackled in order, which allows for anticipation of potential issues or mistakes that may be made otherwise (Figure 6). This prevents rework and reduces the load on the programmer, or group of programmers working on the application, or software.

Figure 6 - Illustration of the Software Development Life Cycle [22]

A breakdown of each stage and what it involves is listed below [23].

1. Planning - Focuses on the project scope.
2. Analysis - Involves gathering information on the intended users of the application through means of interview or research. Alongside feasibility of the application.
3. Design - A prerequisite to beginning work on the product/software application. Involves creating architecture and briefs to ensure the software can be created efficiently with minimal cost.
4. Implementation - The phase of software development. Completed through sprints (agile) or a single block effort (waterfall).
5. Testing and Integration - Rigorous testing of the application to ensure a strong application has been developed with minimal faults. Integration into the public eye. Usually seamless and invisible as it occurs when the software is ready.
6. Maintenance - Continuous monitoring of the application alongside responses provided by users of the application. This ensures the product remains working as intended.

Common models that are based on the software development life cycle model include; Waterfall Model, Iterative Model, and the Agile Model. These can be chosen based on the requirements of the project being developed. This is important as each of the models, including the ones mentioned, have strengths and weaknesses associated with them. These strengths and weaknesses need to be considered when choosing the desired model.

3 Application Analysis

3.1 Existing Solution

3.1.1 Limitations and Drawbacks

There are limitations and drawbacks for each of the Existing Solutions described in the Literature Review. You will find a list of all identifiable limitations and drawbacks below.

- **Registration Requirement:** Any customer that desires to use the scanners must have access to, and be the account holder to a Clubcard or Nectar Card, depending on their choice of Supermarket; Tesco or Sainsbury's, respectively .
- **Tagged Products:** All products that are tagged must be put aside so an employee can remove the tag. Removal of these tags may cause undue delay, especially if the tagged items are buried beneath other products.
- **Item Scanning:** Every item is required to be scanned, unless it's an item which needs weighing, like some fruit or vegetables. These items instead require a barcode to be printed out and scanned.
- **Service Checks:** A service check will happen at the first shop you have with the handheld scanners and may also happen at any shop you have. This means an employee will need to check some of the shopping items, and potentially rescan the entire basket.
- **Infrastructure:** Tesco's Handheld scanners and Sainsbury's SmartShop require the installation of new tills that allow for connection to the devices held by the customer. These tills also allow for priority purchasing for Nectar or Clubcard members.
- **Damage:** Scanners are susceptible to damage or updates, meaning a physical or wireless connection becomes impossible. Barcode damage or interference will cause a scan to become unrecognisable by the device.
- **Internet Connection:** Sainsbury's SmartShop requires constant internet connection, so poor or no internet connection will cause delays and potentially render the application useless. Sainsbury's provides an in-store wifi to counter this issue, however this requires further registration and delay.
- **Basket Transfer:** There exists only one means of basket transfer for the SmartShop Application and In-Store Scanners; this is done through the use of the Camera to transfer the basket through scanning a barcode on the till.
- **Purchase Restrictions:** Sainsbury's SmartShop application prevents you from purchasing age restricted items. As per the government's instructions.
- **Availability:** The systems discussed are only found within large supermarkets. Retail premises do not currently have access to these systems.

3.1.2 Potential Attacks

With any system there exist weak points that have been overlooked and could be exploited by any malicious party. Through analysis of Tesco's shop as you go, Sainsbury's SmartShop, and Tesco's Pay+ application, the most easily manipulatable and high frequency exploits are discussed below.

- **Damage or alteration to the Till Barcode:** This would prevent customers from purchasing the store's goods. This could also be replaced with a look-a-like to add more products to the device or duplicate the basket onto an alternate till or device. Potentially causing the customer unwanted stress or increased expenditure.
- **Damage or alteration to the Product Barcode:** This could be removed to prevent or delay purchase of goods. It may also be replaced completely or switched to add more expensive goods to the customers basket, which without detailed barcodes, could go unnoticed and lead to stock error and further expenditure.
- **Malicious attack on Wifi:** Sainsburys SmartShop requires an internet connection to operate. In the event the in-store wifi is unable to be accessed the customer may have to resort to using data, or discontinuing the shop altogether. In the event of the basket being stored online rather than on-device the user may lose, or have altered, their basket.
- **Damage, or failure of, in-store scanners:** This would render the entire system useless, and would require customers to purchase their products/goods the traditional way.
- **Theft of Clubcard/Nectar Card:** An individual could impersonate someone when using the system, or deliberately remove the card from their person in order to prevent them from using the system.

Due to the nature of some of the attacks mentioned, namely theft, it is very difficult to produce a resolution to mitigate this type of threat. However the other exploits, including; damage to either the till or product barcode, and malicious attack on the stores wifi, should contain some form of protection, which my application will provide.

3.2 Proposed Solution

Primarily, the intended implementation is directed towards retail stores; there currently exists no means of self-service payment methods. Customers must instead seek attention from an employee in order to purchase an item. This is most likely due to the expensive goods that are at stake throughout a retail store, in comparison to supermarkets which only store high valued products in exceptional circumstances. These products are also usually found at the back, or second floor, of the supermarket and are at low risk of being stolen. The number of items recommended to purchase at a self-serve till is below ten items; the price and necessity of the products on display at a retail store make it unlikely for customers to purchase a large quantity of goods at one time. As a result, an application implementation similar to this is very attractive to these regions of the high street.

For each of the potential attacks mentioned in 3.1.2 I have drafted a way of mitigating or removing the risk of attack through these means. They are described further below.

- **Damage or alteration to the Till Barcode:** Remove use of Till barcode to transfer basket data. Instead implement a wireless method, or methods of transfer which uses the software/hardware accessible on both the phone and till to move the customer's basket.

- **Damage or alteration to the Product Barcode:** By replacing the barcode with a QR code, the customer can still scan the item even with thirty percent of the QR code removed. If the QR code is removed or switched, the user can instead scan through the NFC tag, locked to prevent change and located on the product.
- **Malicious attack on Wifi:** The application will be usable offline.
- **Damage, or failure of, in-store scanners:** There will be no implementation of scanners, the application will instead be installed on the users/customers phone.
- **Theft of Clubcard/Nectar Card:** There will be no implementation of any form of restriction of access to the application, provided the user has a phone and the correct operating system (Android 4.4 or higher). The customer needs no account of any form to make a purchase.

Furthermore, the application I intend to implement further refines the transaction process a customer must go through to purchase an item. No longer do they require to separately scan items at the till, and pay for their goods. At the time of purchase all of these will be completed wirelessly in one motion. The application will contain a wide variety of features found amongst the applications and systems (Section 2.3). This includes deleting and adding items to a virtual basket, bookmarking an item, assisting in store location, and more discussed within the implementation section of this report.

3.3 Requirements

From the models discussed within the Literature Review, it is clear that detailed and achievable goals are a necessity. These goals allow us to redefine the scope of the project, and reduce stress on the developers. These goals outline everything the software application is desired to have. Alongside the Gantt chart and WBS, it allows for time constraints to be met.

The functional (user) requirements specify how the application itself must behave, and the features it must have. They specify results of a system that must be met for the application to meet all user needs. Non-functional (operational) requirements are a set of standards that judge the criteria and operation of the system rather than the specific behaviours. Each of these requirements, with regards to my project, can be found below.

3.3.1 Functional Requirements

Table 3 contains the functional requirements of the software application. The table has been ordered, starting with the functional requirements believed to be the highest priority and ending with the lowest. Each requirement has also been labelled to allow for an indication of priority level changes. The levels are as follows; high, medium, and low. High priority is awarded to core functionality of the application that directly impacts the users primary goal. Medium priority assists the user in achieving that goal, and low priority adds as extra functionality the user may not need to use but can if desired.

Table 3 - All the desired functions that are needed to be implemented.

ID	Name	Description	Evaluation	Priority

1.1	Launch Till	The system must allow admins to launch the Till. The system must allow admins to write to NFC Tags.	Located in the systems menu, and is only accessible to administrators. Admins must have full control over the information they desire to write onto the tag and have clear indication as to the process of doing so.	High
1.2	Add Item to Basket	The system must allow the user to add items to their basket via QR. The system must allow the user to add items to their basket via NFC.	A button, clearly indicated, will provide both the NFC and QR service to customers. As NFC is the preferred method, it will launch unless the device does not support it.	High
1.3	Transfer Basket Data / Pay for Basket	The system must allow users to transfer their basket data via Bluetooth. The system must allow users to transfer their basket data via QR.	A 'pay' button will indicate whether a customer desires to pay via phone (Bluetooth) or pay via till (QR). The rest is done automatically after the till has been indicated.	High
1.4	Confirm Basket data	The system must be able to recognise the legitimacy of the provided data and respond accordingly. The system must allow users' locked items to be unlocked in the event of a positive response.	Through comparison of the provided data with the data stored in the database, it can be confirmed. Provided the database is running and can be connected to from the Till. A confirmation of legitimacy provided to the till will then authorise a call request to the database to acquire the relevant information regarding the tag information. This can then be used to disable the tag attached to the items scanned.	High
1.5	Login to Admin	Admins must be able to view and edit their provided information.	Admins will have access to a profile page which provides them all the information requested upon registration.	Medium

		Admins must be able to use this information to log into their account. Admins must be able to log out of their account.	Account information will be stored within the database. This allows for login attempts to be verified and if so, accepted. Log out attempts can be accomplished through the provided button.	
1.6	Swipe on Basket Data	Users must be able to remove data from their basket. Users must be able to find more information about their item. Through the associated company's website.	Through swiping left or right on the desired item, a specific task will be achieved. This is provided an item exists within the basket. Swiping left will remove the item from the list, whilst swiping right will direct the user to the associated website page.	Low
1.7	Change Pages using the Navigation Bar	The System must allow users to open the Account Page using the bottom navigation bar. The System must allow users to open the Basket Page using the bottom navigation bar. The System must allow users to open the Website Page using the bottom navigation bar. The System must allow users to open the Settings Page using the bottom navigation bar.	Pressing on the basket icon within the navigation bar will launch and transfer the user to the Account page. Pressing on the basket icon within the navigation bar will launch and transfer the user to the Basket page. Pressing on the basket icon within the navigation bar will launch and transfer the user to the Website page. Pressing on the basket icon within the navigation bar will launch and transfer the user to the Settings page.	Low
1.8	Register Customers	The System must allow customers to register an account. Each account must be linked to an email Customers must be able to view and edit their	A form is provided and upon filling the customers account will be created. Each component of the form must be completed in order to register an account. Customers will have access to a profile page which provides them all the	Low

		provided information. Customers must be able to use this information to log into their account. Customers must be able to log out of their account. Customers must be able to delete their account.	information requested upon registration. Alongside permissions to edit this information, excluding the email. Registered account information will be stored within the database. This allows for login attempts to be verified and if so, accepted. Log out attempts can be accomplished through the provided button. Deletion of accounts will be achieved through the settings menu and must prompt an alert dialog confirming their desire to remove the account.	
1.9	Adding a web page to Bookmarks	Customers must be able to add any web page, that is not the front page, to their bookmarks. Customers must be able to view their bookmarks. Customers must be able to remove a bookmark as well as visit the associated page.	A button within the toolbar allows users to add any webpage, excluding the front page, to their bookmarks. Achievable through the settings page and selecting the bookmarks button. Similarly with the basket data, swiping left will remove the item from the list, whilst swiping right will direct the user to the associated website page.	Low
1.10	Locating a nearby store	Customers must be able to locate a nearby store of their choice through Google Maps integration.	Achievable through the settings page and selecting the ‘Find nearby store’ button.	Low
1.11	Contact the Developer	Customers must be able to contact the developer with any issues found within the application.	Achievable through the settings page and selecting the ‘Contact the developer’ button. Outfills the recipient's email address.	Low
1.12	Block uses of	Admins must be able to	Achievable through the	Low

	needed technologies	remove access to NFC. Admins must be able to remove access to Bluetooth. Admins must be able to remove access to the Camera.	settings page. The admin(s) can then select any of the three to block. This will help identify what occurs if a device does not have access to a particular technology.	

3.3.2 Non-Functional Requirements

Similarly, a table has been created to store the non-functional requirements of the software application (Table 4). Each non-functional requirement has been tied to a region that is relevant to the requirement.

Table 4 - Descriptions on further requirements of the application and the regions they are applicable in.

ID	Region	Description
2.1	Availability	The application should be available to use at all hours, seven days a week.
2.2	Availability	The database should be accessible regardless of the number of submissions or the number of items needed to be accounted for during a single submission.
2.3	Performance	Data transferred between the database and application should happen in a timely manner. More accurately, within five seconds. This begins after the devices have connected when paying via phone, or when the QR code is successfully scanned.
2.4	Performance	All fragments should be loaded within two seconds.
2.5	Performance	The camera should launch and be functional within three seconds.
2.6	Security	The tags attached to an item should not unlock or be able to be unlocked without confirmation from the database or payment.
2.7	Security	Profile information held on an account should only be viewable to the account holder.
2.8	Security	Till launching should only be accessible to admins.
2.9	Usability	The user interface should be easy and straightforward to use.
2.10	Usability	Items stored in the basket should be displayed in such a way that all information is clear and readable.
2.11	Usability	The price of all items stored in the basket should be easily understood and available at all times. It should also update in a timely manner upon addition of a new item.

3.3.3 Project Scope

Project limitations are inevitable within a small scale research project. With a real-world scenario. Within a real-world scenario these would be implemented but they do not fall within the predefined outcome of the project so they have been excluded.

- **Payment:** Requirement 1.3 discusses the addition of a payment button. This button is in itself an indication of payment. This is because handling payment information requires security. Alongside this, a connection between the application and the customers associated bank is required beckoning further security of the application. As the payment itself is not part of the intended outcome, and does not represent what users need assistance with, it has been removed.
- **Online accounts:** Requirements 1.5 and 1.8 discuss registering an account, plus the storage of accounts within a local database rather than online. In any launched application this would not be feasible as it would require large expenses. Although, due to the nature of the application, an account is not required to operate the core functionality of the application. The accounts will instead be stored within a local SQL database, which is required to be accessible by the device in order to login or for any account activity to occur.
- **Full Business Compatibility:** Requirement 1.6 talks about the use of the website in assistance with the chosen items. With full implementation and production of the application, full compatibility would be available, including business registration. Customers can then use the application in the associated business to add items to their basket, be directed to the correct webpages, and view promotions. For this project I only intend to use one business to demonstrate functionality; Currys PC World. The application's products will be products available on the Currys PC World website. Alongside this, any website functionality will link directly to their website, rather than one chosen by the user.
- **Multiple Authentication Methods:** As discussed within requirement 1.4, the first authentication method is the data held on the customer's mobile device. An additional feature supermarkets have installed for their self-serve tills is a scale. Due to the vulnerabilities of products and their non-secured nature, the scale acts as a second authentication method. However, due to the nature of this feature, and the difficulty of obtaining dummy products with precise weights, it will not be possible to implement this. Although, the weights of products are recorded in the database to provide near authenticity of coding implementation for the application.

3.4 Specifications

3.4.1 Software

The application is compatible with any android device running on Android 4.4 (KitKat) or later. NFC Tag manipulation was introduced in Android 2.3; Bluetooth and Camera

were integrated before android 4.4. Therefore any device permitted to run the application will be able to utilise the code written. This excludes hardware restrictions.

In the event of a non-compatible device attempting to run sections of the application that require a specific android API level higher than their device, an alert will be shown. This alert will redirect them to another way of completing the action they are attempting to proceed with. One example is the use of NFC to obtain product information. If NFC is unable to be used they will instead be redirected to the camera function in an attempt to use the QR scanning feature.

3.4.2 Hardware

For full functionality of the application. A device that contains NFC, Bluetooth and Camera hardware is required. Similarly, in the event that the user's device does not support a specific function, the user will be redirected to other functions. Allowing the user to continue with their intended action.

The likelihood of a device not having a camera installed on the rear of the device is rare; the first occurrence of a phone containing a camera was the J-PHONE, built in 2000 [24]. A camera has become a necessity for any mobile device and thus is included as without it, the phone would be unlikely to perform well financially. However, with both Bluetooth and NFC, there is the opportunity of a non-compatible device attempting to start a function that requires these instruments.

With regard to NFC, it is estimated that over two billion devices contain and have full use of all NFC functions [25]. Samsung, the company with the largest market share of mobile devices, has integrated NFC into its flagship Galaxy range since the Samsung Galaxy S3. Their share, as of the third quarter of 2019 stood at 21.8 percent [26]. Furthermore, Bluetooth has always remained a large component in electronic devices. In 2018 it was estimated that there were over ten billion bluetooth enabled devices world-wide. A growth of five hundred percent since 2012 [27]. This data indicates that at no point should a customer ever not be able to proceed with a task due to the expansive range of devices that contain the technologies.

4 Application Design

4.1 Use case diagram

The complexities and number of functions the application provides has required the creation of two use case diagrams. One illustrating the functions of a user; a customer, account holder, or admin, is able to achieve in either the Phone or Till process. All users are considered Guests until they login to the application, at which stage they become an Account Holder or Admin. This is the same stage that decides the features they obtain access to. Although a customer is not required to own an account to access the core functions of the application.

4.1.1 Phone

Figure 7 depicts the functions achievable by particular users from their phone or device. Any function that is extended from another function is optional and does not have to be utilised by the user.

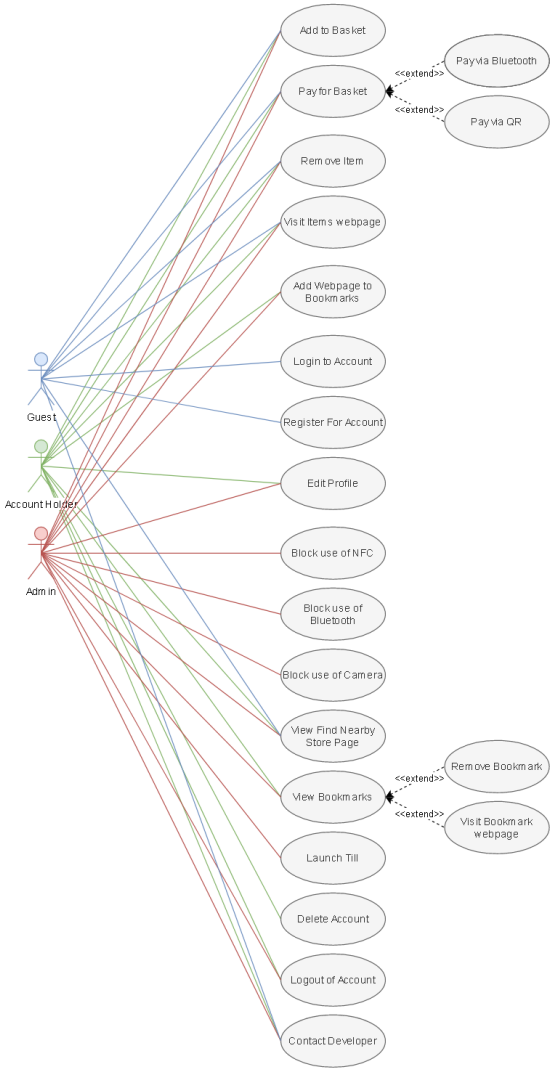


Figure 7 - UML Diagram of all functions performable on the applications phone-side

4.1.2 Till

Figure 8 depicts the functions achievable by particular users from the Tills. All users can write to a NFC tag, but without the correct format the item will be dismissed from the database during its comparison with other product data. This feature is something I intend to restrict to admins in my future works. Any function that is included with another function is a requirement and, in this case, must occur directly after the function is complete.

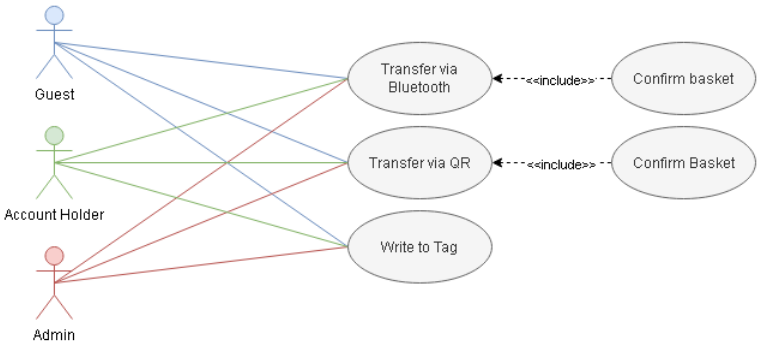


Figure 8 - UML Diagram of all functions performable on the applications till-side

4.2 System architecture

4.2.1 Real-World Implementation

Mobile 1 and Mobile 2 connect the bluetooth hub which redirects the information to/from the Tills selected. These two devices have chosen to pay via their phone. Mobile 3 transfers data one to one through use of a QR Code that Till 2 is able to read, using an in-built camera. Once the basket data is collected, the Till compares it to product data stored in the database to ensure no malicious modification. If products found in the basket are identical to those on the database a variable is returned to the Tills, allowing the users to pay (Figure 9).

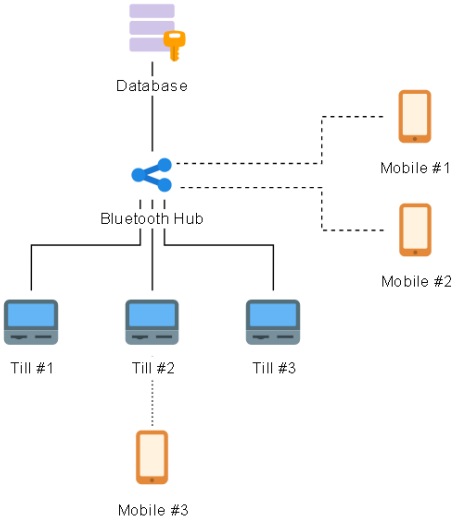


Figure 9 - Diagram of the real-world infrastructure implementation

4.2.2 Small-Scale Implementation

Only one device can be used at a time. This is due to the limited number of devices I have on hand. The devices pair and transfer data directly with one computer. This occurs regardless of chosen method of payment, whether it be through their device or through the provided Till. The devices connect to a computer, containing a PHP file which allows for access to the database. For bluetooth, the information is retrieved and returned to the device and is compared with the stored data. Payment can then occur. For QR, the Till receives the correct information and compares it with the given information. This then indicates whether the payment should be processed and the security tags unlocked (Figure 10).

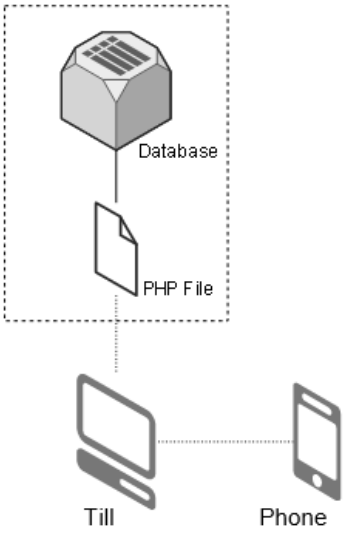


Figure 10 - Diagram of the small-scale infrastructure implementation

4.3 Graphic User Interface

4.3.1 Considerations

When designing the GUI for the application, to ensure a fluid and seamless program, some considerations must take place. These considerations involve the placement of key attributes and interactive features of the GUI; poor placements may cause unwarranted stress or dislike by a customer. These placements must not only exist in places where a customer can expect to find a button but also where they are easily within reach [28]. Below each of these will be discussed.

Reachability of interactive features: Placement of key features and interactive elements should cause minimal damage to physical aspects of the users body. Thus it is important to recognise the potential strain a user may undergo to perform a task or interact with a portion of the user interface. Figure 11 shows the illustrations that demonstrate, on a mid generation iphone, the difficulty of reaching areas of a device. It is also important to note that larger devices, specifically plus sized models may have smaller “natural” reach areas.

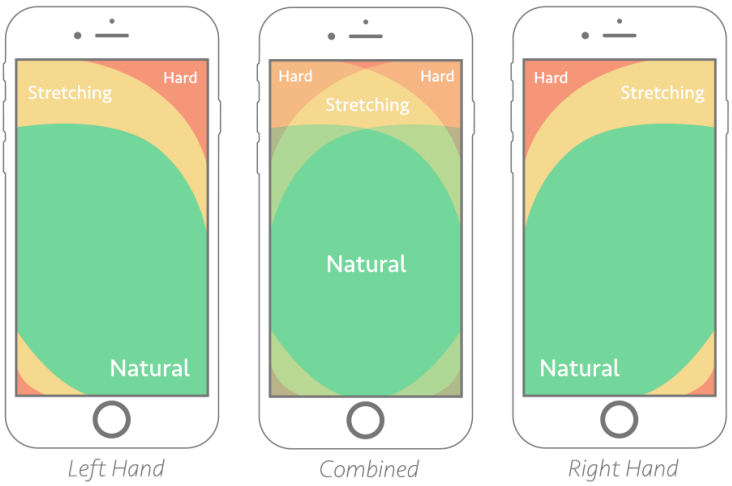


Figure 11 - Reachability of a phones screen based on hand(s) used [29]

This image, when paired with the proportion of users that use phone single and multi-handed, gives us a good insight into the positioning of important features for the interface of the application. Roughly eighty-five percent only use one hand to interact with the screen. This means, depending on their preferred hand, the user may struggle with reaching buttons very close or far from the corner their thumb resides. Whilst only fifteen percent will be able to reach all features in the bottom three quarters of the screen (Figure 12).

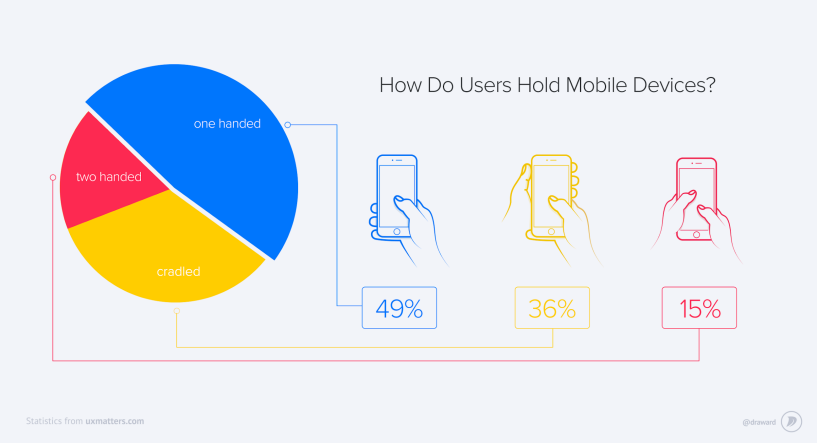


Figure 12 - Percentage of One-Handed, Two-Handed, and Cradled phone users [30]

Following commonality from other major applications: Although new and logical features should be implemented to ensure minimum stress. There is a small amount of consistency expected by software engineers. This allows a minimised learning curve and focuses on the customer's time towards utilising the app for their needs over anything else. As a result most retail applications do not require you to log into the app when making a purchase, instead directing you to their products first. Figure 13, Figure 14, and Figure 15 show screenshots of the applications belonging to high valued companies, with comments on their application design.

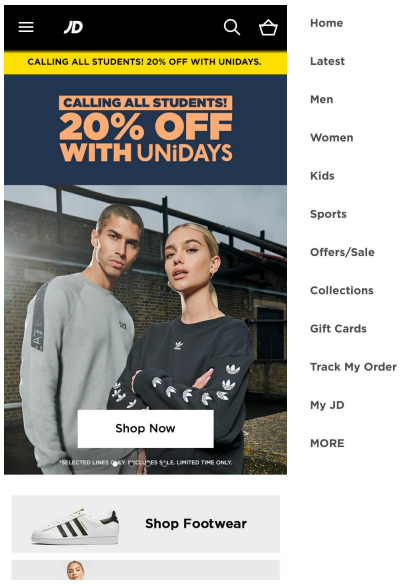


Figure 13 - JD Sports application screenshots

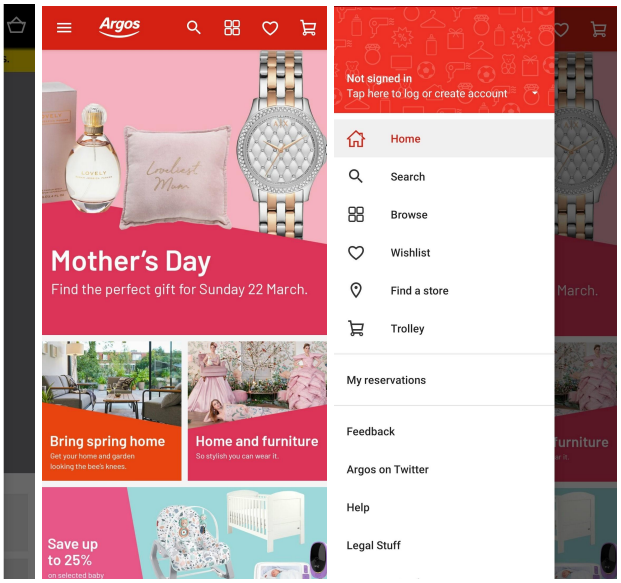


Figure 14 - Argos application screenshots

The above user interfaces come from the JD application (Figure 13) and Argos' application (Figure 14). Their commonalities include; buttons within the toolbar, large buttons, clear text and a side navigation bar. Although, as discussed prior, a large quantity of the buttons fall outside of the "easy to reach" regions. This is common and due to the restrictions of user interface design. As well as this, these places often allow

for smart placement, as well as clear indication of button placement as they are not integrated alongside busy, or clustered regions of the screen.

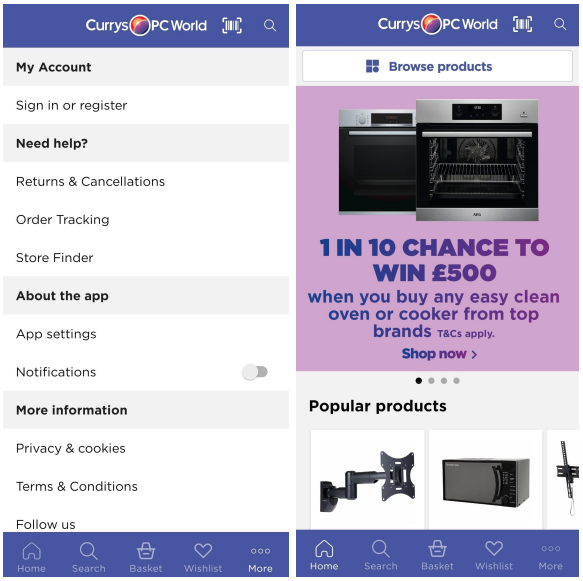


Figure 15 - Currys PC World application screenshots

Similarly with the Currys PC World application, buttons can be found in the toolbar. However the navigation bar is found at the bottom of the display. This is more intuitive as these buttons are more commonly pressed than those that are found in the toolbar. They can also be interacted with more easily, being partly accessible single-handedly and fully accessible multi-handed. Here text and buttons are also clear and large, as above (Figure 15).

With all considerations in place, I developed drafts for each screen I would be required to implement whilst coding the application. These can be found in sections 4.3.2 and 4.3.5 of the report. The most-used fragments have been overlapped with the aforementioned image dictating the reachability of different users. For the phone these include; Login, Basket, and Bluetooth [Pop-Up]. With regards to the till, there is no need to facilitate this “reachability scale” as the user’s free hand will be interacting with the device and thus have free range over all interactive features.

4.3.2 Phone Drafts

Below shows the Phone's user interface drafts. The Phone requires five fragments; Login, Register, Profile, Bluetooth and Basket. For each of the fragments, the bottom segment represents the navigation bar, used to display each fragment. Login and Register fragments contain input boxes, indicated by the white boxed regions. Below each is a button that submits the inputted data. The profile fragment displays the provided data, and is only available to access when logged into the app. The website fragment contains the website of the selected business. Finally, the Basket contains a list that displays the items they have added. Furthermore, the basket contains two additional buttons. One within the toolbar, that allows for users to pay for their

products. The second is located at the bottom of the device, allowing the user to add more items to their basket.

Settings, Bookmarks and Find Nearby Store Pages have also been drafted. Settings contains a ListView of Buttons that enable an indefinite amount of features to be added to the application. Each accessed by selecting the relevant button within the ListView. This is also how you reach the Bookmarks and Find nearby store pages. The Bookmarks Page contains a RecyclerView which allows for dynamic lists, depending on the amount of bookmarks registered to the account that is logged in. Finally, the find nearby store page contains a WebView, allowing the user to gain access to Google Maps within the application. Three Pop-Ups have also been created; 'Bluetooth', 'GenerateQR', and 'NFC'. Bluetooth allows the user to see the bluetooth devices, connect, and transfer their basket data. The QR generator displays a QR code crafted from the items within their basket. While the NFC contains an image illustrating the process of reading from a tag (Figure 16).

The drafts had been coordinated to ensure key features are identifiable, each colour indicating a specific implementation or function. More specifically, Light grey regions indicate buttons, whilst dark grey regions are indications of the toolbar. Finally, boxed white regions are miscellaneous and to be specified when important.

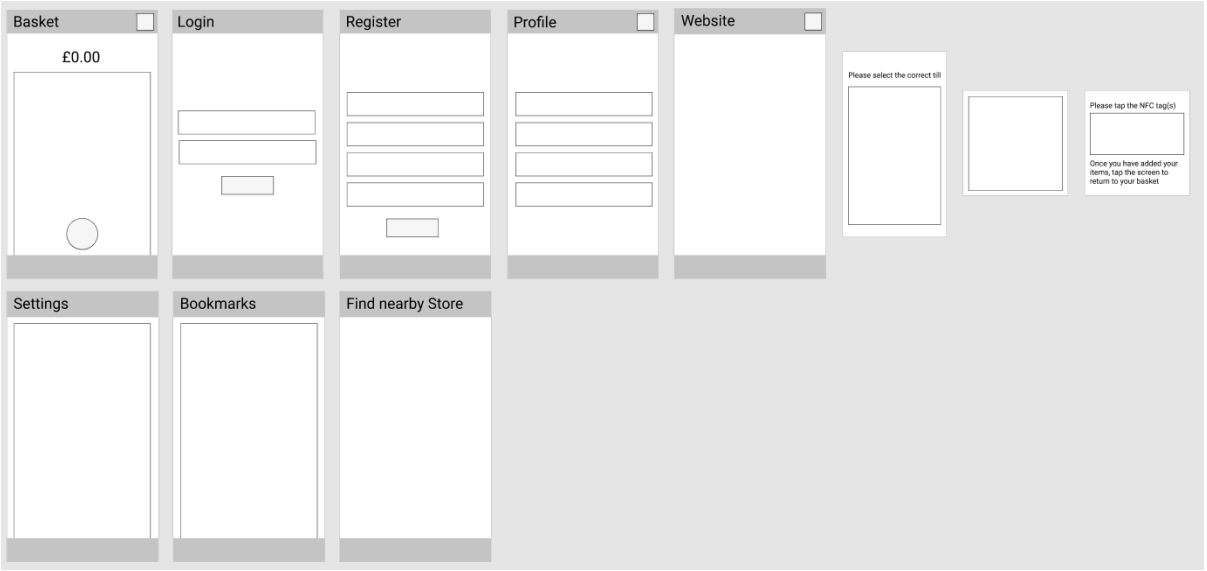


Figure 16 - Drafts of each page, and pop-up, of the applications phone-side

All Alert Dialogs have been excluded from design drafts as they only allow for limited manipulation (adding of TextView, EditText, and others), but near to no graphical influence.

4.3.3 Phone Draft Reachability

I have selected the three most visited pages or pop-ups, to demonstrate the reachability I intend to create within my application. These pages include; Basket, Basket with Till Pop-Up and Login Page (Figure 17). With the exclusion of the buttons located at the top right of the screen it is clear that the designs enable customers to reach vital parts of

the application. Furthermore, the buttons also exist in locations expected by the customer. Where a button falls out of the reachability of the user, it is a button that is not regularly used, or placed out of reach to prevent mispresses.

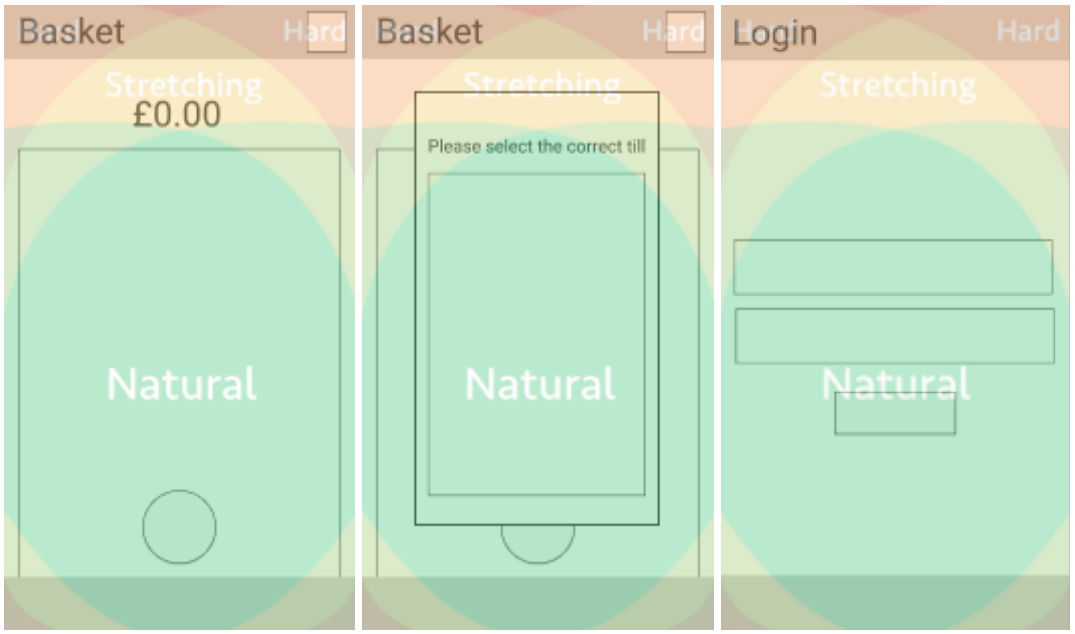


Figure 17 - Reachability of three key draft pages for the applications phone-side

4.3.4 Phone Final Result

The final result achieved for each of the pages is shown below. Although not drafted, AlertDialogs and some Miscellaneous pages/pop-ups have also been provided. The Pages named “Account” are the Login, Register, and Profile Pages, respectively. The Settings Page also has three different variants depending on the type of account, or no account, that is logged in at that time. The Settings Pages are shown in the following order; Guest, Account Holder, Admin. Any buttons not viewable on the Settings Page also indicates a restriction to that function for the account in question. (Figure 18)

The “Are you sure you want to delete your account” pop up appears on press of the “Delete Account” button. The “Are you sure you want to log out” pop up appears on press of the “Logout” button. The “Choose your preferred email application” pop-up appears on press of the “Contact the Developer” button. The “Enter Data Values” pop-up appears over top of the Website Page when the Bookmark button, located in the top right of the screen, is pressed (Figure 18).

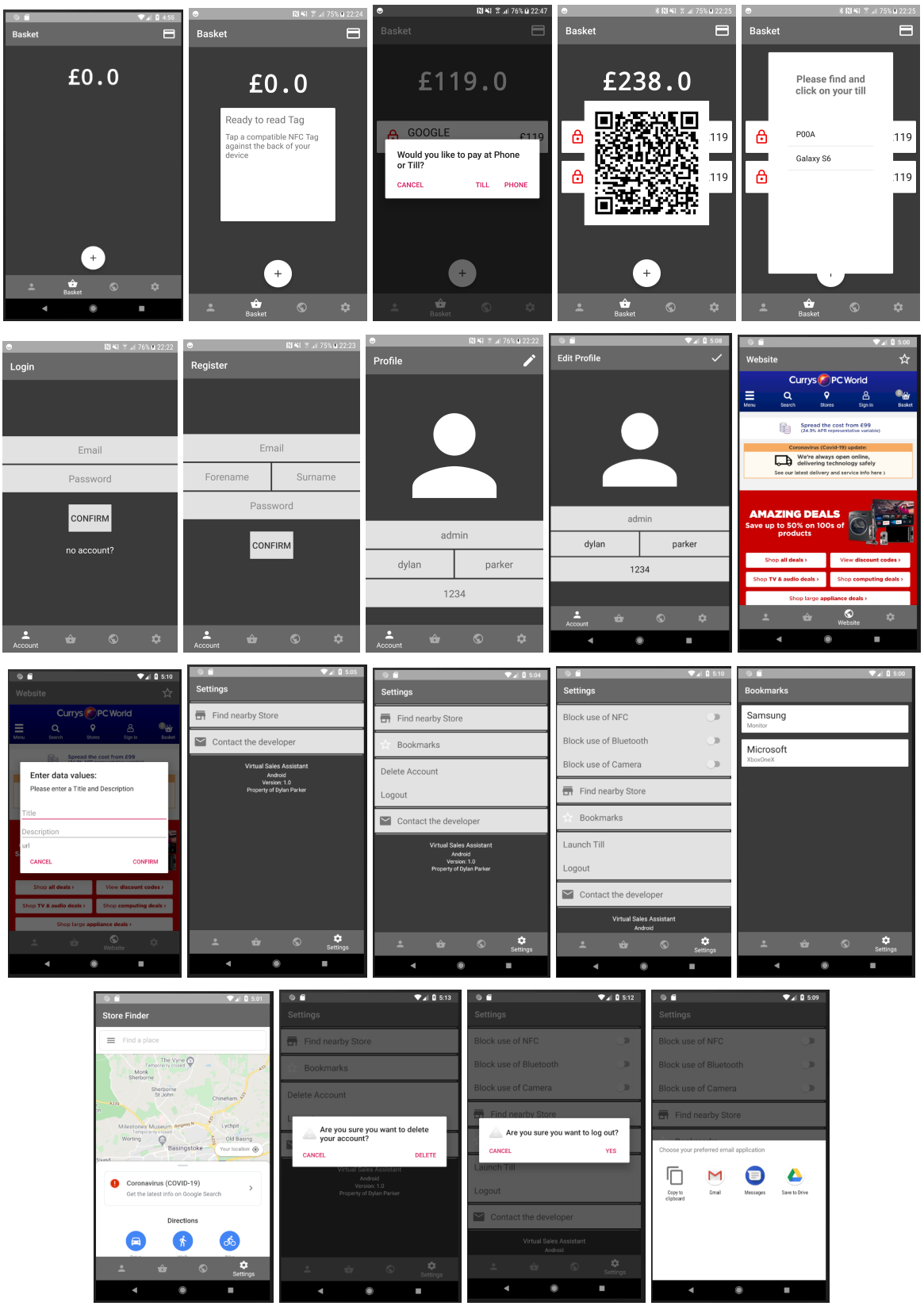


Figure 18 - Final results of each page, and pop-up, of the applications phone-side

4.3.5 Till Drafts

The following images show the Tills user interface drafts. The Till requires two pages; Main Page and Basket Confirmation Page. The Main Page contains two buttons which allow users to identify whether they intend to pay via till or phone. The Basket Confirmation Page shows a replica of the basket on their handset with any issues in the event of error. This means the white box, located on the Basket Page beneath the basket cost, contains the list of items they are purchasing. Finally, two Pop-Ups have been drafted; Write NFC Tag and Bluetooth. These smaller boxes will overlap a full-scale draft and will contain images within the predefined white boxes (Figure 19).

Similarly as before, the drafts had been coordinated to ensure key features are identifiable, each colour indicating a specific implementation or function. More specifically, Light grey regions indicate buttons, whilst dark grey regions are indications of the toolbar. Finally, boxed white regions are miscellaneous and to be specified when important.

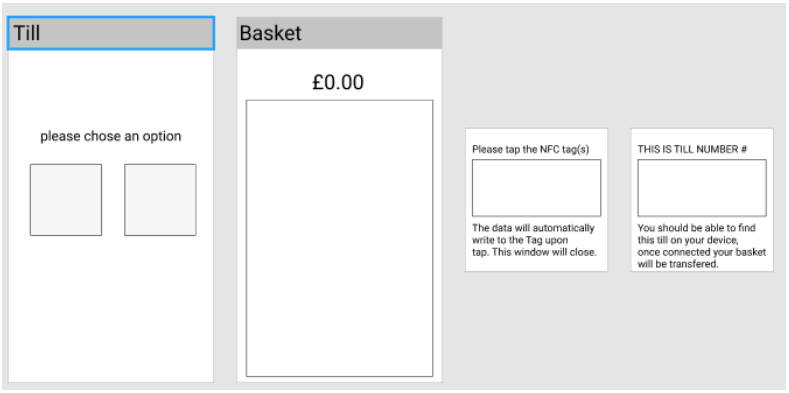
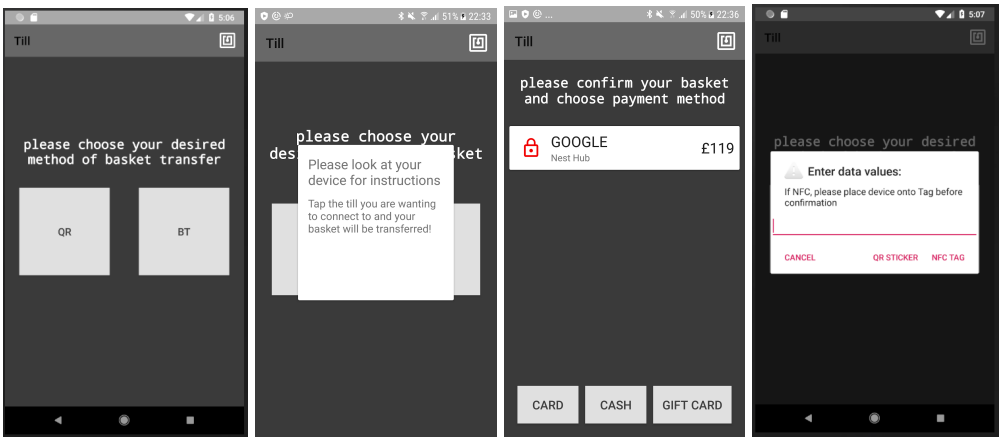


Figure 19 - Drafts of each page, and pop-up, of the applications till-side

4.3.6 Till Final Result

Figure 20 shows the final results for the Till Pages. Within each pop-up I was unable to input the ImageView due to its restrictive nature, this would have prevented smaller screened devices from seeing the text. The user can also choose to “print” a QR sticker through the Write To Tag button, although at this time it displays a QR on the screen as printing one is not feasible within the scope of the project (Figure 20).



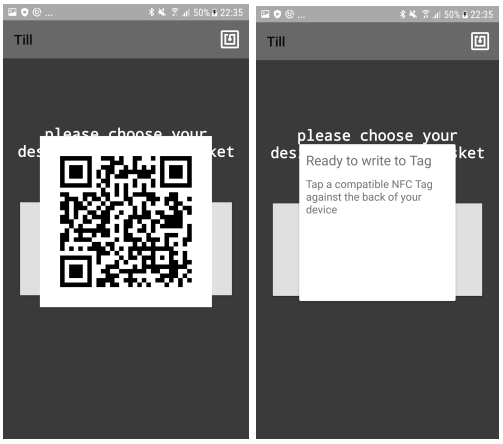


Figure 20 - Final results of each page, and pop-up, of the applications till-side

4.4 API Design

For implementation of the functional requirement noted in 3.3.1, communication between the application and database must be generated. Table 5 contains the required parameters and responses, for and from the database, for particular users that attempt to achieve the desired action mentioned in the functional requirements.

Table 5 - Actions required by the database for specific users and the parameters sent and received.

ID	Action	User	Parameter	Response
1.8	Register	Guest	email, password, forename, surname	null
1.8	Login	Account Holder	email	password, forename, surname
1.5	Login	Admin	email	password, forename, surname
1.8	Update	Account Holder	email, password, forename, surname	null
1.5	Update	Admin	email, password, forename, surname	null
1.8	Delete	Account Holder	Email	null
1.4	Receive [Confirm Basket]	Guest	productid	brandname, modelnumber, price, quantity, weight, description

1.4	Receive [Confirm Basket]	Account Holder	productid	brandname, modelnumber, price, quantity, weight, description
1.4	Receive Item	Guest	productid, tagid	null
1.4	Receive Item	Account Holder	productid, tagid	null
1.9	Set [Bookmark]	Account Holder	email, title, description, link	null
1.9	Get [Bookmark]	Account Holder	email	title, description, link
1.9	Remove [Bookmark]	Account Holder	email, link	null

4.5 Entity relationship diagram

An entity relationship diagram for the implementation of the MySQL Database server is shown in Figure 21. For the tables Products and Items, productid and itemid are each the primary keys respectively. There exists a foreign key within the items table; productid, taken from the Products tables, is needed to ensure there exists a relationship between each table because the table Items contains information about the product. This information is found in Products. For each Product you can have many items, hence the one-to-many relationship illustrated in Figure 21.

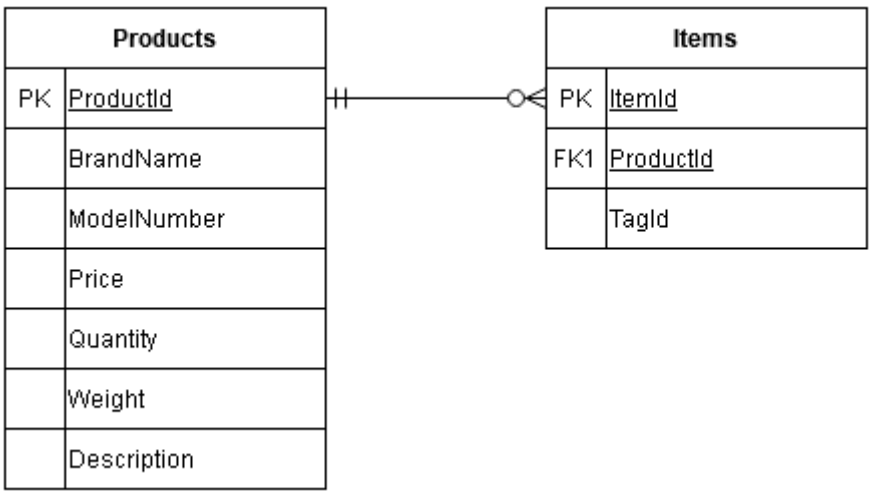


Figure 21 - Entity relationship diagram of the Products and Items tables found in the database

Another entity diagram exists, illustrated in Figure 22. This links the accounts with associated bookmarks the email holder has added to their account. The Primary Key for Account and Bookmarks are AccountID and BookmarkID, respectively. The Accounts database contains a unique email requirement which is used as a Foreign Key within the Bookmarks table. Although this approach is not traditional, and a foreign key should instead be made from AccountID, it was required due to coding restrictions provided by the PHP scripture that was used to access and perform SQL instructions on the database.

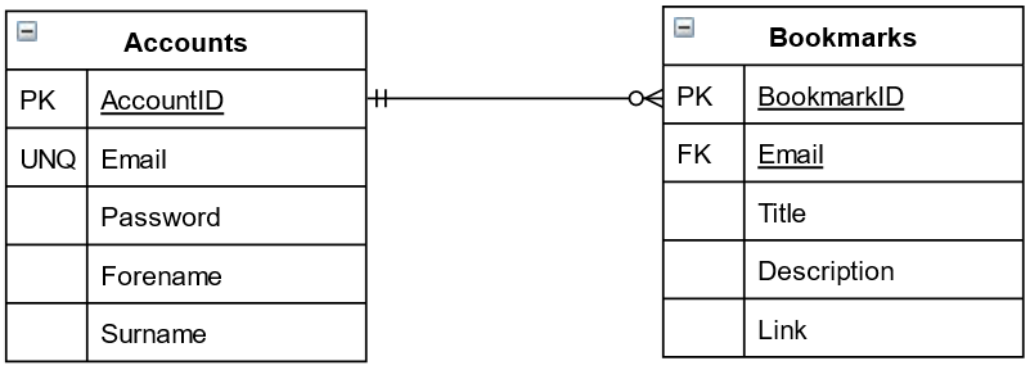


Figure 22 - Entity relationship diagram of the Accounts and Bookmarks tables found in the database

5 Application Implementation

5.1 Chosen software methodology

For the project implementation, I have chosen to follow the guidance provided by the Waterfall Model. Although typically a more rigorous approach to software creation, it offers a good structural approach. This enables me to distinctly plan each component of the application with other priorities in mind. As I am developing this project, alongside other modules, it provides a good gauge of the work I am to contribute to remain on track. At the beginning of the project a Gantt chart, seen under the 'Planning' heading, was produced with this methodology in mind. In the event of delay, procedures are drafted and executed to ensure further time is not wasted. As each component is completed, it can then be dismissed completely as all objectives and implementations for the respective component are finished. The next component can then be started (Figure 23).

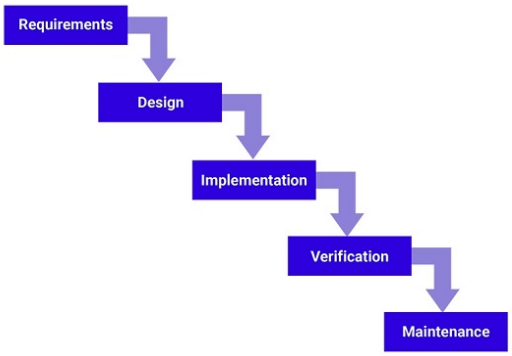


Figure 23 - Visual interpretation of the waterfall model software methodology [31]

5.2 Prototype

To ensure the project was viable, a prototype implementation was developed. This allowed me to confirm that all the technologies I intend to use would be compatible in one application. Although it is seen previously that the final implementation of the application uses bluetooth, to communicate between the phone and till, it was intended that NFC be used instead. However, due to the limitations of NFC I was unable to make the phone impersonate an NFC tag. Thus bluetooth was supplemented into the application and the use of NFC was repurposed to allow the customer an additional method of adding a product to their basket. With this, the prototype was a success. Implementation of the critical functionality had been proven operational.

5.3 Software and Language Justification

Table 6 - Titles for each Software and Programming Language used within the project.

Operating System	Android 4.4 (KitKat)
Integrated	Android Studio

Development Environment (IDE)	
Programming Languages	Java, XML, PHP, SQL
Other Programs	XAMPP (Apache and MySQL)

Android 4.4 was selected as it allows for implementation of NFC, Camera and Bluetooth due to their availability in earlier versions of Android. As well as this, 96.2 percent of all android devices are run on a version of android 4.4 or higher [32]. Therefore a very significant quantity of phones will be able to install and use the application.

My chosen IDE was Android Studio. This was primarily chosen due to my previous use of the IDE and my understanding of the Programming Languages (Java and XML) that it encompasses. Android Studio also allows for builds and debugging of the application without a physical device through the emulator it contains. Furthermore, it has extensive manipulability and extension potential. Therefore, any resources I need to produce the desired result can be found through libraries other people have provided. Similarly with XAMPP, used for the database. Given my experience with both of them it seemed a smart choice to use these programs as they provided a good base, and allowed me to implement the functions specified in section 3.3.1 and 3.3.2. XAMPP is very easy to understand and provides bountiful tools for configuring the SQL database, from both a user interface and coding standpoint.

5.4 Methods

5.4.1 Basket

The basket was a major requirement for the implementation of the application, as it allows for the development of the core functionality of the application, as described within section 3.3.1 of the report. Through inspection of other applications seen within section 2.3, it was clear that a virtual basket allowed for consistency and a reduced chance of confusion when introducing the application to an existing market. Thus I chose to implement a RecyclerView into the application, in order to allow for a dynamic list with variable storage commitments. This is initialised within the onCreate method of the BasketFragment (Appendix 1). Within this class, I utilised the ViewHolder methods to obtain the associated textView, imageView, and relativeLayouts. The onBindViewHolder uses these Views to then display the information provided to the AdapterBasket class for the user to see (Appendix 2). Through summing the price of each product within the prices ArrayList, found in the BasketFragment, I was able to obtain a value that could be displayed as the cost of the basket. With the list implemented, I then had to allow for users to input data into the dynamic list. This involved adding a floating action button into the Basket Page; in its infancy, this would allow users to enter any information into the list to ensure it would function as expected (Appendix 3). It since has been altered to ensure all items added are obtained from either QR codes, or NFC tags.

Since users could add items to their basket, it was important that they could also remove any items they were no longer interested in. This involved implementing the swipe feature specified in the functional requirements of the application; it was achievable through the use of the ItemTouchHelper which can identify when an item within the recyclerView is swiped left, or swiped right. When the item is swiped right, the onChildDraw method would draw and display an image and decide the product to signify the deletion was about to occur (Appendix 4). The BasketFragment would then be called again to re-initialize the virtual basket, to update the recyclerView. The final core functionality to implement, involving the basket, was the pay feature. This required the implementation of the QR and Bluetooth functionality before it would be completed, this in turn, required the implementation of the till.

5.4.2 Till

One of the first decisions I had to make when creating the till functionality was whether I should continue to have a single application and split the functionality between the till and the phone, or have two applications. After attempting to develop two separate applications, I found that these devices would share a large proportion of coding and thus it would be more efficient to develop them as one application, with two distinct regions. I created a user interface for the till to accommodate the features I would be implementing in the next few stages. These are; writing QR code, writing to NFC tags, reading QR codes, and transferring basket data between both the phone and till, as well as the database and till.

5.4.3 Scanning a Product

The first functionality for adding a product to the basket, required the implementation of reading QR codes. This extended the functionality explored within section 5.4.1, where a floating action button was added and allowed input of any data, provided correct format, to be placed into the virtual basket. Through the implementation of both the QR and NFC read functionality this has been restricted to variables obtained through these sources alone. This provides an additional level of security to the application, as without the correct application or reader, obtaining the plaintext of products becomes more difficult and less likely to manipulation.

Through use of the Google Barcode API, I first built a barcode detector and set its format to read QR codes. Subsequently, I built a camera source that utilises this barcode detector to allow for QR codes to be identified when in the view of the device's camera (Appendix 5). One issue this presented was the failure to focus the device, reducing the resolution of the image regardless of the distance the device was from the object. To resolve this, I added an onClick function that recognised when the user had pressed on the screen, obtained the parameters of the source, and focused the camera (Appendix 6). I now was able to detect QR codes and focus the camera. However, there existed no way to retrieve the data the camera could read. This involved me overriding the surfaceChanged method, and implementing a Processor for the barcodeDetector to obtain the information contained within the QR code displayed on the screen (Appendix 7). The surfaceChanged method is called everytime a new surface is detected in the

cameraSource. The surfaceCreated and surfaceDestroyed methods respectively start and stop the cameraSource functionality.

The second stage of functionality regarding the implementation of the basket, and furthermore adding items to the basket involved the use of NFC. After implementing QR, a function that uses the camera, I wanted to implement a technology that would reduce the time and effort required to add an item to the basket. The only restriction is the requirement of NFC hardware on the users device, hence the implementation of the QR reader. First, I wrote the method readFromTag that connects, obtains, and converts the NFC tag data into a readable variable (Appendix 8). This method is called from the overridden onNewIntent method which identifies the nearby tag, and provided the variable declaration indicates a read operation, obtains the message from the readFromTag function. This data is then returned to the activity that created the intent and launched the Activity (Appendix 9).

When the application, or NFCActivity is paused the onPause method is called; this usually occurs when the user presses the home button on their device, or switches to another application. The onPause method dictates what is to occur, when any instance of application pausing occurs, namely the foreground dispatch, which controls the override for NFC detection and manipulation, is disabled. This will allow other functions to manage any NFC operations and be prioritised before the application. The onResume re-enables the foreground dispatch to allow the application authorisation over any other processes that attempt to intercept the NFC activities (Appendix 10).

The NFCActivity java file also contains a displayMetrics method, that allows me to resize the display of the NFCActivity user interface when the Activity is launched. In the final implementation of this method, the user interface takes up seventy percent of the screen (Appendix 11). This method, alongside the creation of a new theme, allows for ‘pop-ups’ within the application, that can be dismissed by pressing the screen outside of the dimensions listed in the displayMetrics method (Appendix 12).

5.4.4 Adding a product to the store catalogue

The next stage in implementation of the core functionality involved allowing the user to ‘print’ QR codes, and write to NFC tags that were to be placed onto their products. As printing a QR code fell outside of the scope of the project, I instead displayed it on the till’s screen so it could be scanned and verify the functionality of both read and write QR. Within a real-world implementation the barcode would be printed and placed on the respective product. Unfortunately, as Android Studio has no support of Writing a QR code I was required to involve a third party implementation, QRGEncoder [33], that allowed me to generate QR codes through providing it a string, the contents type, and desired dimensions (Appendix 13). This QR code is then displayed to the user through the use of an ImageView.

Previous implementation allowed the application to read NFC tags located on products. I was also required to develop the NFC functionality that allowed for tags to be written to. This led to the development of writeToNfc, found in the same Activity, NFCActivity, as the readFromNfc method. This method, similarly to readFromNfc, connects to the

identified NFC tag and creates a MIME, a requirement for NDEF record creation, of the desired data to input into the tag. This data is then written into the tag as a MIME Record (Appendix 14). As this method is contained within the same java file as the readFromNfc method, it can utilise the same overridden functions, onResume, onPause, and onNewIntent to control the use of foreground dispatch and more (Appendix 9, Appendix 10).

Both the write functionalities, QR, and NFC, utilise the displayMetrics method described within section 5.4.3 of the report. This allows both of these implementations to be displayed as pop-ups and take up less screen estate, as well as allow for dismissal when finished, or otherwise.

5.4.5 Transfering Basket

This stage involves transferring the customers basket data to the till for processing. This has been achieved through the use of two technologies; Bluetooth and QR. The first implementation only included Bluetooth as a method of transfer as it bolstered a secure connection of which the basket data can be transferred. However, this process takes a greater number of steps than transferring via QR. Hence QR was introduced for customers that desire speed over security. QR is also only acceptable to use on a basket of very low items, any more than four and the QR becomes too large and difficult for the camera to interpret.

The implementation of QR was achieved through the reuse of the QRAcivity.java file within the till. On the other hand, the implementation of Bluetooth was much more complex and involved identifying devices that were paired to the device and placing them into an ArrayAdapter, for them to be displayed to the user (Appendix 15). The BluetoothActivity is used by the phone and till when attempting to forge a connection, and transfer data. It was important each could be defined and provided the correct classes. An if statement within the onCreate method allows the separation of these two devices into the client, and server, threads, respectively. The Client Thread opens a socket for communication with a searching device, which is then occupied by the Server Thread. A unique UUID ensures no other device will attempt to connect to the same socket of the Client. Whilst no socket is detected, the Server routinely waits for one to become available, before initiating the TransferData Class (Appendix 16). Upon connection, the handler message is updated and the TransferData class is run, and passes the customers basket information (Appendix 17 , Appendix 18). The TransferData class creates input and output streams for sending, and receiving data, respectively. These streams are obtained through the socket identified within both the ServerThread, and ClientThread, classes. The ClientThread class runs the write method and the OutputStream is occupied with the basket data. The ServerThread executes the run method, and the InputStream obtains the information being transmitted. The handler message is updated and the contents returned to the till's previous activity (Appendix 18, Appendix 19).

The Bluetooth technology utilises the displayMetrics method described within section 5.4.3 of the report. This allows the list of paired devices to be displayed as a pop-up and

take up less screen estate, as well as allow for dismissal when finished, or otherwise (Appendix 11).

5.4.6 Basket Confirm

The final stage of the core functionality of the application involved finding and obtaining the products within the customers basket, from the database, and comparing them for any errors or illegitimacy. This is achieved through calling the Database AsyncTask class, this allows the application to be continued to use, without freezing, while the operations occur. As the operations only take a fraction of a second, this type of class is optimal provided the context and requirements. The first called function is the doInBackground method; this obtains the provided string and identifies the required procedure though looking at the final variable in the string. This variable acts as the identifier for the PHP file that needs to be accessed and run to obtain the desired result of the program (Appendix 20). Regardless of the procedure required, the first steps require opening and connecting to the database, and producing an OutputStream to the database. This allows the application to pass information, and requests to the database. A BufferedWriter is created using this OutputStream. Firstly, the contents variable is stripped and separated into a new string file containing only the product ids found within the customers basket. This is then encoded alongside a declaration to identify the data as such. The BufferedWriter passes this data to the specified php file and the BufferedWriter is then cleared and closed, alongside the OutputStream. An InputStream is then opened to allow any requested information to be provided to the application. The returned information is read from a BufferedReader and passed returned to the doInBackground method (Appendix 21). The database function is called again, following the same process, to allow for the run of the receiveitem.php file. This code is needed to update all items and identify them as sold within the database (Appendix 20, Appendix 21).

The onPostExecute method is run after the completion of the doInBackground method and allows for manipulation or extraction of the data obtained within the doInBackground method. The data returned by the database contains a lot of unwanted characters. These are stripped through the use of the replaceAll function, alongside a regex. Each returned variable is then compared and a true, or false, is added to an array depending on the legitimacy of the product. This array is returned to the ConfirmFragment and the final results displayed (Appendix 22). Appendix 23, and Appendix 24, contain the PHP code used to extract and update, respectively, the database code.

5.4.7 Toolbar and Bottom Navigation Bar

At this point in the implementation I wanted to reorganise and improve the user interface of the application to closely resemble a polished and company made program. This involved moving a high majority of the buttons, including the Pay button, into the toolbar to reduce the cluttered feel of the Basket Page. This was achieved through defining a bToolbar variable when replacing fragments (Appendix 25). When creating the toolbar, an overridden method onCreateOptionsMenu, bToolbar, and a set of if statements, allowed the fragment to be identified and thus an associated button to be placed in the toolbar. These buttons included; payment, bookmark, edit profile, and

confirm edit profile (Appendix 26). When the button is pressed, the onOptionsItemSelected is called and manages what each button was required to execute (Appendix 27).

The next implementation was the use of a Navigation Bar, that allowed the user to move between pages of the application. As the core functionality had been implemented and I found myself with spare time, I wanted to make the application more closely resemble a finished product and hence decided to implement it into the program. This was intuitive as it was possible through adding some xml code, alongside an onNavigationItemSelected function to specify the operations of each button when pressed (Appendix 28).

SharedPreferences was also introduced at this stage of the implementation to allow the items within the list and the images associated with them, stored as a string and an arraylist, respectively, could be stored when the application was closed or paused. This usually occurs when the user presses the home button, or switches applications. Appendix 29 contains the onPause method, which implements the saving of the basket using the provided SharedPreferences.Editor. This editor associates the values saved with a name so they can be identified when obtaining them from the memory. This restoration of variables occurs within the onResume method of the NavigationActivity (Appendix 30).

5.4.8 Accounts

The first functionality implemented that utilised the navigation bar, as well as allowing the application to feel more like a developed shopping application was the use of accounts. These allowed restrictions to be placed on unregistered users, as well as the introduction of admins which provided a much more improved debugging mode through the ability to disable each of the primary technologies within the application.

The addition of the profile page, found on the navigation bar, allowed unregistered users a location to register, and registered users a place to find and edit the information they had provided. Appendix 26 shows the buttons found in the toolbar on the Profile and Edit Profile pages. Once a profile edit is completed, the database is passed the altered information and updated (Appendix 31).

Appendix 32, 33, 34, and 35 contain PHP files called by the DatabaseClass when the user wants to delete, update, register, or login using an account.The DatabaseClass calls these files through identifying the operation, and encoding between one and four of the following pieces of information; email, forename, surname, and password. When logging in to the application, the database returns all data relevant to the email provided through use of the BufferedReader created within the Class (Appendix 21). The method onPostExecute further handles this account_login information by comparing it with the entered information, if each piece of information is identical, the account information is passed back through use of the intent. In the event of failure, the user remains logged out and on the login page (Appendix 22).

The accounts data was also added to the onPause and onResume methods within the NavigationActivity, to keep the user logged into the application when left or closed

(Appendix 29, Appendix 30). This means the user is only able to logout of the application through manually visiting the Settings page and selecting the 'Logout' button.

5.4.9 Website

Appendix 36 shows the implementation of the website page, allowing the user to navigate the store's website when needing more information on a product, or exploring their catalogue.

Section 5.4.1 shows the ItemTouchHelper allowed for another direction of swipe, I decided to dedicate it towards providing the customer with more information about the report. When a customer swipes left, it displays a coloured rectangle and icon to inform the customer they are navigating to the website. They are moved to the Website Page and the product web page loaded (Appendix 4).

5.4.10 Settings

To allow for debugging, a finished implementation of core features, and a location to place extended features, I implemented a Settings Page. This provided users with the ability to logout, delete their account, and block the NFC, Camera, and Bluetooth provided they are an admin. These features finalized everything worked on thus far, and met the objectives of a significant proportion of the functional requirements for the program. In order to prevent certain features being available for specific users, I made sure each function displayed in the Settings Page was associated with a relativeLayout. This layout's visibility could be set to View.GONE in order to remove it from the page and prevent specific users from using the functionality (Appendix 37).

When one of the relativeLayouts was pressed, provided it did not contain a Switch, it would replace the fragment with the relevant page associated with the layout. The only other exceptions were the relativeDelete, and relativeLogout layers. These allowed the user to delete, or logout of their account and when pressed would display an Alert Dialog that confirms the action. If confirmed the users SharedPreferences and database is updated, or the users local account information is removed, respectively. The Settings Fragment is then switched with the Basket Fragment (Appendix 38).

The Setting Switch button data was also added to the onPause and onResume methods within the NavigationActivity; this allows an admin to alter the access to functions on a specific device, and it remains unless the admin accesses the device again and alters its state (Appendix 29, Appendix 30). This is very beneficial when debugging and checking the functions within the application.

5.4.11 Tutorials

Appendix 39 shows three public functions, each display the relevant tutorials when called. When the Positive Button is pressed the tutorial value is changed from False to True, preventing it from showing in future uses of the application.

The tutorials data was also added to the onPause and onResume methods within the NavigationActivity; this prevents the tutorials from showing on reopen, or revisiting, of the application (Appendix 29, Appendix 30). This means the user will not be hassled with

tutorials, and in the event of forgetting any procedures must instead visit the application within their OS settings and reset cache.

5.4.12 Find Nearby Store

Similarly to the website page, the find nearby store function was implemented using a web client with javascript enabled and a provided url (Appendix 40). When opened the Find Nearby Store webView redirects you to the maps.google.com web page. Allowing you to freely roam and locate a store close to your location.

5.4.13 Bookmarks

Similar to the Basket, a dynamic list was created using a RecyclerView, that was passed multiple arraylists (Appendix 41). This RecyclerView was initialised within the onCreate method of the BookmarkFragment; a ItemTouchHelper was created to allow for each bookmark within the list to be deleted or viewed through the website (Appendix 42, Appendix 43). A bookmark is required to be linked to an account and cannot exist without one.

Appendix 44, 45, 46, and 47 contain PHP files called by the DatabaseClass when the user wants to update, create, delete, or obtain the bookmarks. The DatabaseClass calls these files through identifying the operation, and encoding two or four of the following pieces of information; email, title, description, and link. Only obtaining the bookmarks (bookmarks_get.php) requires the database to return any information using the BufferedReader created within the Class (Appendix 21). The method onPostExecute further handles this bookmarks_get information by adding some characters that were removed when applying a regex to them. This is achieved through a single for loop and identifying the locations of a https or http string. The bookmarks are then added to an arraylist and returned to the intended intent (Appendix 22).

5.4.14 Contact The Developer

Appendix 48 shows the implementation of the contact the developer feature. This allows users to email the developer of this application. After selecting their email application an email is composed and the sender and recipients email address auto-filled.