KING'S College LONDON

University of London

Loyalty Card App

6CCS3PRJ Final Project Report

Author: Faizan Ullah Joya
Student ID: 1326551
Supervisor: Prof. Dr. Luca Viganò

April 2015

Abstract

In 21st century where retailers incentivise customers with loyalty points or discounts in exchange to collect data of our spending habits, we possess and carry numerous loyalty cards. All of these cards may not fit in one wallet physically nor we wish to carry extra weight, hence we often do not have access to them when required and thus miss the opportunity to collect or use points when required at time of purchases. We aim to solve this real world problem with additional security elements with an implementation of a loyalty card application for smartphone.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary.

I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service.

I confirm this report does not exceed 25,000 words.

Faizan Ullah Joya

April 2015

Acknowledgements

I would like to give special thanks to my helpful supervisor, Prof. Dr. Luca Viganò. The supervision and support that he provided truly helped with the smooth progression of this project. The assistance is much appreciated. Thanks to Department of Informatics staff and IT services at King's College London.

Also thanks to my family for the encouragement and moral support throughout the project.

Contents

1. Introduction 9

1.1 Project Motivations 10

1.2 Scope 10

1.3 Why target Android? 11

1.4 Objectives 13

2. Background 15

2.1 Smartphone 15

2.2 Loyalty Cards 15

2.3 Barcode 17

2.4 Existing work 18

2.5 Security 19

3. Requirement & Specification 20

3.1 Requirements 20

3.1.1 Content requirements 20

3.1.2 Security requirements 21

3.2 Specifications 22

3.2.1 Content specific 23

3.2.2 Security specific 25

3.2.3 Limitations 27

4. Design 28

4.1 Use cases 28

4.2 System Architecture 29

4.3 Data Flow 31

4.4 State Machine 32

4.5 Graphical User Interface 33

4.5.1 Pin Setup 33

4.5.2 Pin Entry 33

4.5.3 Pin Reset 34

4.5.4 Cards Activity 34

4.5.5 Add Card 35

4.5.6 View Card 36

4.5.7 Edit Card 36

4.5.8 Scan Barcode 37

4.6 Third-Party Content 37

4.6.1 ZXing Library 37

4.6.2 Icons 38

5. Implementation and Testing 39

5.1 Development Approach 39

5.2 System Functionalities 42

5.3 Implementation issues 47

5.3.1 Resuming application user authentication 48
5.3.2 Update Card reload 49
5.3.3 Avoid SQL injection 50
5.3.4 Maintaining Lollipop theme for older versions 51
5.3.5 Actionbar back buttons 52
5.4 Changes and Additions from design 52
5.4.1 Enter password twice 52
5.4.2 Hiding Keyboard 53
5.4.3 Showing barcode in other activities 54
5.4.4 Screen rotation settings 54
5.4.5 About page 55
5.5 Testing 55
5.5.1 Unit Testing 56
5.5.2 Non-functional Testing 56
5.5.3 Requirement Based testing 58
6. Professional Issues 64
7. Evaluation 65
7.1 Application Evaluation 65
7.1.1 Limitations 65
7.1.2 Security 67
7.1.3 Overall 67

7.2 Project Evaluation 68

8. Conclusion & Future Work 71

8.1 Conclusion 71

8.2 Future Work 72

9. Definitions 77

10. References 78

11. Appendix 80

Appendix A: Background Information 80

A1: Activity Diagram - Beep'n Go 80

A2: Activity Shortcut Diagram - Beep'n Go 82

A3: Resuming Activity Diagram - Beep'n Go 85

Appendix B: Project Information 86

B1: State machine diagram 86

B2: Graphical User Interface Design 87

B3: Class diagram 95

B4: Change in Graphical User Interface 102

B5: Diffefrent Keyboards for different input types 103

B5: Memory Allocation Management 105

B6: Implemented system 106

B7: Database schema 116

Appendix C: User Guide 117

Appendix D: Program Listing 126

Activities: 131

Barcode: 166

Crypto: 175

DB: 176

Helpers: 182

Util: 186

XML layout: 188

1. Introduction

Demand for saving has increased in shopping market recently and with introduction of Loyalty cards with the concept of collecting points to save and gain rewards on shopping. Due to increase in smartphones, there has also been increase in demand for virtually carrying loyalty cards on phones. Loyalty cards applications like StoCard, KeyRing, and Beep'n Go have come to rise with all of them with over 1 million downloads in Google App-store. Beep'n Go sets the record for most downloaded loyalty card applications with over 5 million downloads[1]. It is easy to spot that Beep'n Go offers some security feature (pin protection) for loyalty cards over others which can be a contributing factor to its popularity, however as we shall see later that it is easily bypass able. This reveals that there is market for a secure loyalty card mobile application.

Current implementations of loyalty card applications are clunky with too many unnecessary features and without security of their content. Some of them are rather difficult to use with inefficient structure and flow of the applications.

Customers collecting loyalty points using Loyalty Cards can redeem them at point of purchase. Hence the question about security arises to protect the Loyalty cards from being used by unauthorised person. We show a great deal of interest in security elements of this project's applications to find a possible solution where we have securely stored loyalty cards in smartphones. The implementation will aim to be used for wide variety of cards and not specific to a type of loyalty card.

[1] https://play.google.com/store/apps/details?id=com.mobeam.beepngo

1.1 Project Motivations

Two topics I am most passionate about are Android development and Security. This project supports and allows me to work on both of the topics while solving a real world problem. In the process I will learn a lot that otherwise I would not have without undertaking this project. This project will also prove helpful for research and gaining knowledge on the topics (mentioned above). This project will enable me to further build my knowledge and develop skills in fields of security and mobile development.

Inspired by ApplePay [2] and banking applications on smartphones we introduce a solution to carrying numerous loyalty cards in smartphone. In similar fashion to storing bank cards on the phone and pay using smartphone, we aim to store loyalty cards in the phone without carrying the physical cards. This would save the extra weight of the cards in wallet and offer access to the cards when required on smartphone. Solution to make a mobile application is based on the fact that smartphone owners tend to carry our smartphones with them. Based on market research data collected by Ofcom, over 60% of adults in UK have a smartphone (1).

1.2 Scope

This project has two main areas to tackle loyalty cards and security. Storing loyalty cards, adding editing and deleting should be rather straightforward as present examples exist. However security elements of these features are a huge concern whilst developing a secure system. Security elements have not been fully

[2] https://www.apple.com/apple-pay/

implemented in the current systems. We will look at in more detail in the next chapter.

Another challenging area will be to insure it is not allowed to bypass the system and gain access to loyalty card without correct user authentication.

1.3 Why target Android?

The project should include a mobile application for smartphones mobile devices. The aim is to produce a native android application to the device as opposed to the web application.

We aim to focus on Android market of the smartphones due to many reasons. Android users are likely to live in cheaper regions of Britain and have less money to spare compared to Apple users.[3] (2). Hence they are more likely to use Loyalty Cards and save money and look to collect points. According to the annual global phone operating system Infographic [4] (3), the following Figure 1.1 and Figure 1.2 represent the market share of phone operating systems.

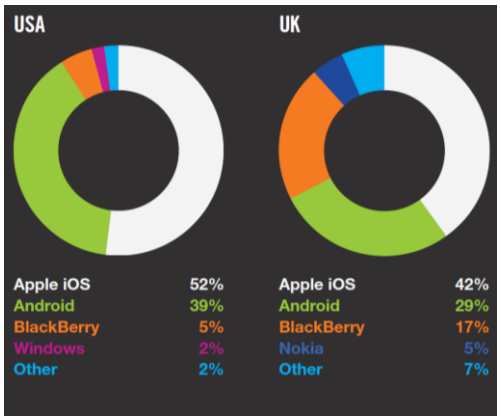

Fig 1.1 Infographic representing mobile OS for USA and UK

[3] https://yougov.co.uk/profiler#/Android/

[4] http://connect.icrossing.co.uk/infographic-android-pulls-in-twice-as-many-users-as-apples-ios_11372

Figure 1.1 shows android compromises of about 30% of mobile phones in United Kingdom. However global market share of android mobile operating system is increasing, as demonstrated in Fig 1.2, and android is the smartphone market leader with over 40% of phones running it. This is followed by iOS, Apples mobile operating system, at 22% market share.

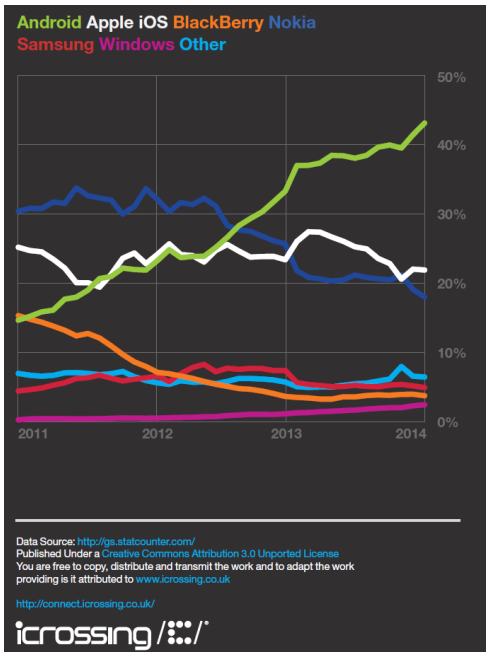

Fig 1.2 Mobile OS Market share over time since 2011

Figure 1.2 reveals that android users are on the rise. Align with developer's skill set and limited time to complete the project; we aim to focus on Android development for this project.

Another great aspect of android development is that it uses Java programming language. Because Java offers automatic garbage collection feature, developer need not worry about memory management and focus on the more important features of the project. Android has its own memory management

system which clears the memory once a threshold is reached without disrupting operations.

1.4 Objectives

Solution should be relatively easy to use. It should have self-explanatory graphical user interface within the current theme of operating system. It should follow the layout constraints of official guidelines of Android as much as possible. Application should securely store the content and not allow other apps to access its data in such way that can be used or allowed for reproduction of the content, stored card details devices.

We aim to develop program application for mobile users to securely save loyalty cards. Requirements for this project will be as followed:

1. To have a program that will store loyalty cards
2. The application should only allow authorised person to view contents using some sort of verification of user
3. The structure and flow of application should be simple and straightforward so it is easy to use
4. It should not be possible for other applications to access its contents and data on a non-rooted phone

In the next chapter we will discuss the background content for this project and looking into: security; barcode; loyalty cards; and rooted versus non-rooted phones. In the design section we will discuss the design philosophy and system architecture of the system. Followed by the implementation and testing section which will

provide in depth view of implementation of all the application features and security elements. This will also reveal how we met the requirements and fulfilled deliverables of this project. Lastly, we will discuss conclusion of the project and suggested improvements in future work section.

2. Background

2.1 Smartphone

A mobile device with the capability to perform functions typically found in a computer is called a smartphone. A smartphone usually from a manufacturer is in non-root access mode. A rooted device grants administrative permissions to the user and can perform functions that may be unsafe to the operating system of the device. Most applications on an app-store are aimed at non-rooted devices. We also focus on non-rooted devices as it provides protection from applications accessing another application's private data. Database accessing and editing is strictly permitted only to the application that owns it by default saved in internal memory. (4)

Similar security feature exist in the current Banking Apps (Such as Santander, Barclays) for android where they do not allow the user to use the app on the rooted phone reasons being: other applications can access their data which can result in security breach and potentially theft of data, money in banking applications case and loyalty points in loyalty cards applications case.

A smartphones usually has a rear facing camera. This will be useful in our application for scanning loyalty cards.

2.2 Loyalty Cards

A typical loyalty card consists of a card shaped object similar or same shape as a credit card or business card. A loyalty card of a retailer would consist of the retailer

logo a name of the card on one side whilst providing a unique barcode and card number to the user. During purchase the customer may present their loyalty card at which the loyalty card is scanned by the barcode reader machine to collect points or equivalent monetary value. The customer may wish to use their collected loyalty card points, usually once they have reached a specific limit. They make collect points over long periods of time such as years and months. One of the most popular loyalty cards in Britain is the Nectar card.[5] Each retailer would have their own unique loyalty cards which cannot be used in other retailers unless of same brand. A customer may carry numerous loyalty cards, one for each retailer.

The collected points in loyalty cards can have monetary value as they can be exchanged for products or to gain discounts on purchases at the retailer. Hence loyalty cards have sum value to them and should not be allowed to be used by unauthorised person. Similar to money and bank cards, one would not allow an unauthorised person to use them. For example: Nectar cards points equate to spending money which can be redeemed in specified stores such as Sainsbury's. 500 nectar points equate to at least £2.50 to spend at participating retailers (5). The problem is that secure loyalty card applications do not exist that protect such money that user may have accumulated in form of points. We aim to provide a solution to this problem with security features in loyalty card application.

If user's wallet is stolen with loyalty cards among it, then an unauthorised person can use the loyalty card points without user's knowledge nor authorisation. Currently supermarkets and retailers do not verify if a loyalty card or a gift card belongs to the person using them during the transaction when in person. There

[5] http://www.mirror.co.uk/lifestyle/family/top-10-loyalty-cards-help-4020713

may be additional checks performed during online transactions for using the loyalty cards and gift cards however it is not the same in store. We aim to discuss and solve this security problem.

2.3 Barcode

Barcode is a representation of encoded data typically divided into two types: linear barcodes and multi-dimensional barcodes (6). Linear one dimensional barcode consists of a series of bars and white spaces representing a unique set of characters. A multidimensional barcode consists of a two-dimensional way to present data. Similar to one dimensional, linear barcode, this represents data however can have more data per unit area. We aim to focus on the linear barcodes as they are found on loyalty cards. Loyalty cards are relatively unknown to use multidimensional barcodes.

Barcode scanners scan the barcode and decode the data. Different types of scanners include: laser; Linear Imager; and Area Imager (7). Laser and Linear imager scanners can only scan 1D barcodes. Either of these are found at the supermarket tills. Area Imager scanners have the advantage of being able to also scan 2D barcodes in addition to 1D.

Barcodes found on loyalty cards are 1D which is in line with the barcode scanners retailers have on tills. These barcode decode to only numerical data and cards have unique numbers. In our system we aim to use smartphone camera to scan the barcode.

2.4 Existing work

Existing loyalty card applications for mobile devices include the features of adding deleting editing loyalty cards. However a secure storage of cards and only allowing authorised user to access them is not guaranteed. Google Play store is the largest android app-store (8).

Loyalty cards applications StoCard and Beep'n Go: offer storing cards but do not offer ample security or any at all for them. StoCard offers no security whatsoever but it is more focused on providing additional deals and promotions to the user.

For and android application it is possible to have shortcuts on an android device to a specific activity of an application installed on the phone. Beep'n Go offers a pin protection, however it is completely redundant as one can bypass the system by producing a shortcut straight to the viewing cards activity. Activity diagram see (appendix A1) shows normal flow of application, and activity shortcut diagram (appendix A2) shows steps for bypassing the Beep'n Go pin entry requirement to access cards. Furthermore, if the application is paused and resumed it does not ask to re-enter pin but opens straight away (see appendix A3). This is a massive security flaw as if the user forgets that the application is open in the background then an unauthorised person can gain access when resumes the application.

What will differentiate this application project to all the other loyalty card applications in the market is that this will include properly implemented elements of security in addition to usual features, storing viewing and editing loyalty cards. We aim to offer correct pin protection feature to open cards, which cannot be

bypassed and will ask for user verification even when the application is being resumed from background paused state.

2.5 Security

Pin or password should not be stored as plain text as this can allow a potential hacker to obtain them if system security is compromised. A recent example is of Sony PlayStation in 2011, when Sony allegedly failed to encrypt the passwords and stored them in as plain text. Sony admitted hackers got access to 77 million usernames passwords and other confidential details. (9) One way to avoid this from happening is to encrypt the passwords using a one way function. Cryptographic hash function is a one way function that takes in a variable length password and generates a fixed length digest using a salt value. A salt is constitutes of random values that is used to generate the hashed password or digest. Same function can also be applied to generate a digest for pin. Common hash functions are MD5 and SHA256.

A pin consists of numbers 0-9 and usually four digits long. Reasons for that are:

- Users can easily remember them, bank cards also require 4 digit pin
- Four digits mean that number of possible combinations are 10^4 = 10,000. This can take the attacker a long time to try using brute force.

3. Requirement & Specification

The purpose of this project is to create a program that the user can use to store loyalty cards in the android smartphone securely.

3.1 Requirements

The smartphone will require a camera on the phone to scan barcodes on the loyalty cards. If there is no camera on the smartphone then user can simply manually type the card number. If the barcode is not scanned then there will also be no barcode data stored in the system to be able to generate the barcode.

3.1.1 Content requirements

- C1 – The application should allowing storing Loyalty cards.
- C2 – Application should display all the stored cards in alphabetical order.
- C3 – User should be able to select a card form the list to view individually with its corresponding card details.
- C4 – A card can be edited and saved with updated details.
- C5 – A stored card can be deleted from the application.
- C6 – User should be able to store card details without scanning barcode: if barcode is not available; or it does not exist.
- C7 – User should be able to scan barcode of the card and store it

- C8 – Application should be able to generate barcode for card if barcode data exists for stored card.
- C9 – User should be able to search for cards using their names as search query.
- C10 - User should get some feedback in form of error dialogs or toasts for invalid inputs
- C11 – Keypad for input should be relative to the input field. Numeric keypad for pin and card number input fields; Password keypad for password entry; and general keyboard with predictions should be shown for names and other fields.
- C12 – Application should ask for conformation for: exiting without saving; discarding adding card details; deleting card.
- C13 – Graphical user interface of the application should complement the system interface as defined by the operating system.

3.1.2 Security requirements

- S1 – A pin should be used to authenticate the user. Four digit pin should be set upon first time opening the application or if the pin has not been setup. This should also ask to setup a backup password.
- S2 – Pin and password should be stored as digest (hash) in database.
- S3 – Pin reset activity should ask for password used whilst setting up pin during pin setup. If the password matches then only allow to reset the pin.

- S4 – User should be asked to enter Pin to authenticate user upon opening or resuming the application from the background.
- S5 – One should not be able to create shortcuts to a specific activity within this application other than the pin activity. Should not be possible to bypass the pin authentication and access cards.
- S6 – No more than five pin entry tries should be allowed at a time. When the user resumes application after it can be given one more pin entry try. To reset such pin entry attempts user should exit the application fully (close) and then open again.
- S7 – No entry should be made to android logs of secure content. This will ensure other applications cannot read the logs of this application that could allow jeopardise the system.
- S8 – Backup password to reset should be at least 4 characters long.
- S9 – Typing fields should be checked for invalid characters.

These security requirements are important to make sure the application produced is secure and does not allow an unauthorised person to use the loyalty cards. Some requirements will be more challenging than others.

3.2 Specifications

Importance of the requirements is shown next to them. A high priority requirement must be implemented for the project to be considered complete. Without low priority requirements, useful but not critical, the application should offer basic

functionality. High priority requirements should be implemented first and then tackle other lower priority requirements.

3.2.1 Content specific

Requirement Code	Requirement Details	Specification	Importance
C1	The application should allowing storing Loyalty cards.	Uses SQLite database in android and custom Card objects. Compulsory fields: Name; and Number of card. Non-compulsory: ID; Barcode number; barcode format. Non-compulsory attributes do not require user to manually input them but may obtain data from the database and barcode scanned data.	High
C2	Application should display all the stored cards in alphabetical order.	Order the cards by their name using a custom list adapter as Card will be a custom object. Will implement Comparable for Card object. A custom item view for individual card in the list.	Medium
C3	User should be able to select a card form the list to view individually with its corresponding card details.	Listview with card objects. Allow user to click on individual items in the list to open the view card activity.	High
C4	A card can be edited and saved with updated details.	Updating Card in database and update the cards list	High
C5	A stored card can be deleted from the application.	Delete Card in database and update the cards list	High
C6	User should be able to store card details without scanning barcode: if barcode is not available; or it does	Add card using Name and Number fields. Name and Number will be compulsory fields.	High

	not exist.		
C7	User should be able to scan barcode of the card and store it	Scanning barcode will require external library. Store the barcode number and barcode format in addition to the name and number of the card. In this case barcode number and card number can be same but are not always as seen in section 2.3 barcode.	Medium
C8	Application should be able to generate barcode for card if barcode data exists for stored card.	While viewing card generate the barcode	Medium
C9	User should be able to search for cards using their names as search query.	Search bar in Actionbar and use text changed listeners while using the input text as query to adapt the cards list to cards with the names matching the search query	Low
C10	User should get some feedback in form of error dialogs or toasts for invalid inputs	Error dialogs would show errors for each text field if invalid entry made. A toast can be shown when a card is successfully added.	Low
C11	Keypad for input should be relative to the input field.	• Numeric keypad for pin • Numeric keypad for Card number • Password keypad for password entry • General keyboard for other	Low
C12	Application should ask for conformation for: exiting without saving; discarding adding card details; deleting card.	Show dialogs to confirm user choice. Dialogs will require a cancel choice and a positive choice where the decision is final to continue user request such as deleting the card.	Low
C13	Graphical user	Using the android	Medium

	interface of the application should complement the system interface as defined by the operating system	developer guidelines offered by Google Android.[6]	

Table 3.1: Content Specific System Specification

3.2.2 Security specific

Requirement Code	Requirement Details	Specification	Importance
S1	A pin should be used to authenticate the user. Four digit pin should be set upon first time opening the application or if the pin has not been setup. This should also ask to setup a backup password.	Checks when running the application if the pin has been set. Will require to access database to check if user exists. If user does exist then ask for pin otherwise open the pin setup activity.	High
S2	Pin and password should be stored as digest(hash) in database.	Using SHA256 algorithm generate hash from pin/pass + salt.	High
S3	Pin reset activity should ask for password used whilst setting up pin during pin setup. If the password matches then only allow to reset the pin.	Hash of Password entered matching hash of password stored in database determine if allowed to change the pin. New pin there store as hash.	High
S4	User should be asked to enter Pin to authenticate user upon opening or resuming the application from the background.	Open pin activity on opening or resuming the application. Set pin activity to be default to load. Default activity would load every time application is opened.	High
S5	One should not be able to create shortcuts to a specific activity within this application other than the pin activity. Should not be possible to bypass the pin authentication and	Use Activity names in the Android Manifest. Do not name the activity with usual format of com.<project name>.<activity name>	High

[6] https://developer.android.com/design/index.html

	access cards.		
S6	No more than five pin entry tries should be allowed at a time. When the user resumes application after it can be given one more pin entry try. To reset such pin entry attempts user should exit the application fully (close) and then open again.	Keep log of pin entry tries left. If exceed the tries left then show some error that no more tries are left. When user resumes decrement the tries counter by 1. When user opens the application the counter would be reset.	High
S7	No entry should be made to android logs of secure content. This will ensure other applications cannot read the logs of this application that could allow jeopardise the system.	Just do make sure that the Log used during development are not part of compiling code. Check logs when un app so see if confidential data is being logged.	High
S8	Backup password to reset should be at least 4 characters long.	Do length checks for the input fields. If invalid then show error dialogs.	High
S9	Typing fields should be checked for invalid characters.	• Check the entered character do not include special characters that may break the database or system. • Specification of requirement C11 will partly help achieve this: only allow certain type of characters to be typed. • Use input validator for numbers and password	High

Table 3.2: Security Specific System Specification

3.2.3 Limitations

Limitations for the system:

a. A card can only have one barcode that can be scanned and stored the data for. So if a card has more than one number on it then the user can store the card number in addition to scanning and saving the barcode if available.

b. Scanning ability for different types of barcode will be limited by the library used. In this case Zxing library which allows scanning of linear, 1D barcodes.

c. Some barcodes may not scan so user is still able to store the card number of the card. Unfortunately without scanning the barcode it will not be possible to know what type of barcode it is to generate it from just user input. So to generate barcode, the user must save the card by scanning the barcode rather than manually typing the details.

d. The database of cards will not be accessible by other application if the phone is not rooted or the applications on the phone do not have root access. This point will be mentioned in the user guide.

4. Design

The design chapter is to give detailed steps with suitable figures revealing how the program would work at an abstract level. This system overview gives us basic understanding of the system to help us implement it. We will discuss implementation of our system in the next chapter.

4.1 Use cases

Use cases identify the interactions between the actor and the system. User is modelled as an actor in our use case diagram. It defines how program provides functionality and defines what is required of the program.

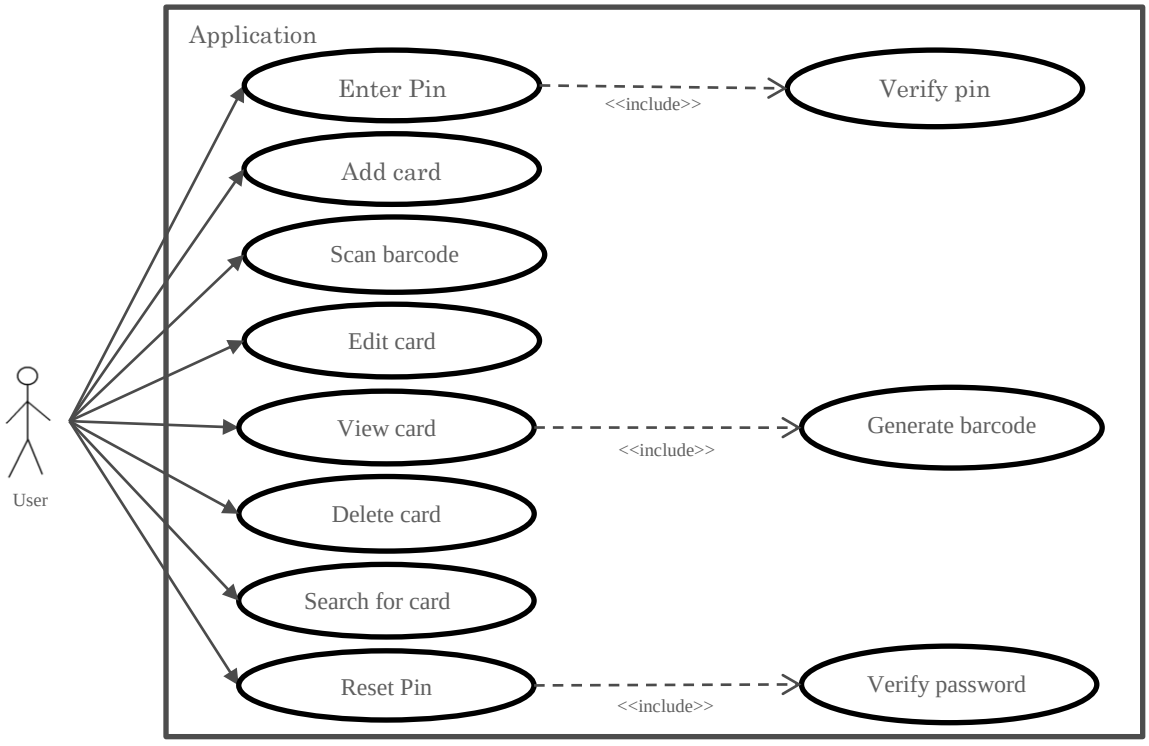

Figure 4.1: Loyalty Card Application Use-case Diagram

Trigger for the application will be opening the application on the device. The user can enter pin which is verified by the system. However if the pin is not set then it will ask the user to set the pin and provide a backup password. Pin setup is not a feature of application that user will interact with more than once hence is not covered in the use case diagram. Pin reset is a feature that the user can interact with more than once in the system hence is present in the use case diagram. To reset pin the password entered by the user is then matched in the system for the backup password.

User can add a card and view it. A card in the system can be edited and viewed. The user can scan the barcode of the card for storing. Viewing the card will also generate the barcode given barcode was scanned and stored in the system prior to viewing it. User can also search for the existing cards.

4.2 System Architecture

System architecture provides a visual overview of the system by dividing it in to subsystems. Dependency of a subsystem is represented by arrow. A three tier architecture diagram followed (Fig4.2.) reveals the presentation layer, logic layer and data management layer for our system.

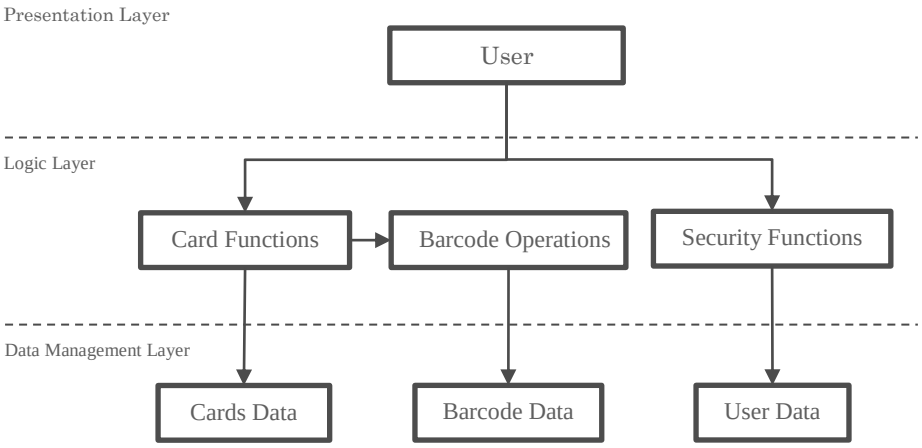

Figure 4.2: Loyalty Card Application System Architecture Diagram

Architecture diagram (Figure 4.1) reveals there is one user hence one graphical user interface, three system logic layer functions (Cards functions, Barcode operations, and Security functions), and three data repositories (Cards data, Barcode data, and User data).

User operations apart from login consist of adding cards, viewing cards, editing cards, deleting cards, and searching for cards.

The Logic layer offers the core of algorithms and functions involved in the system logic. Card functions make the necessary calls on the cards data repository and barcode operations to scan the card and generate the barcodes. The barcode operations involve of accessing barcode data, such as type of barcodes, in from the barcode data repository. Security functions provide the security functions as mentioned in the specifications under section 3.2.2. Security functions involve using data from the user data repository.

Data management layer holds the data repositories which the logic layer functions access.

4.3 Data Flow

Data flow diagram is important for our project. They show how data flows in the system.

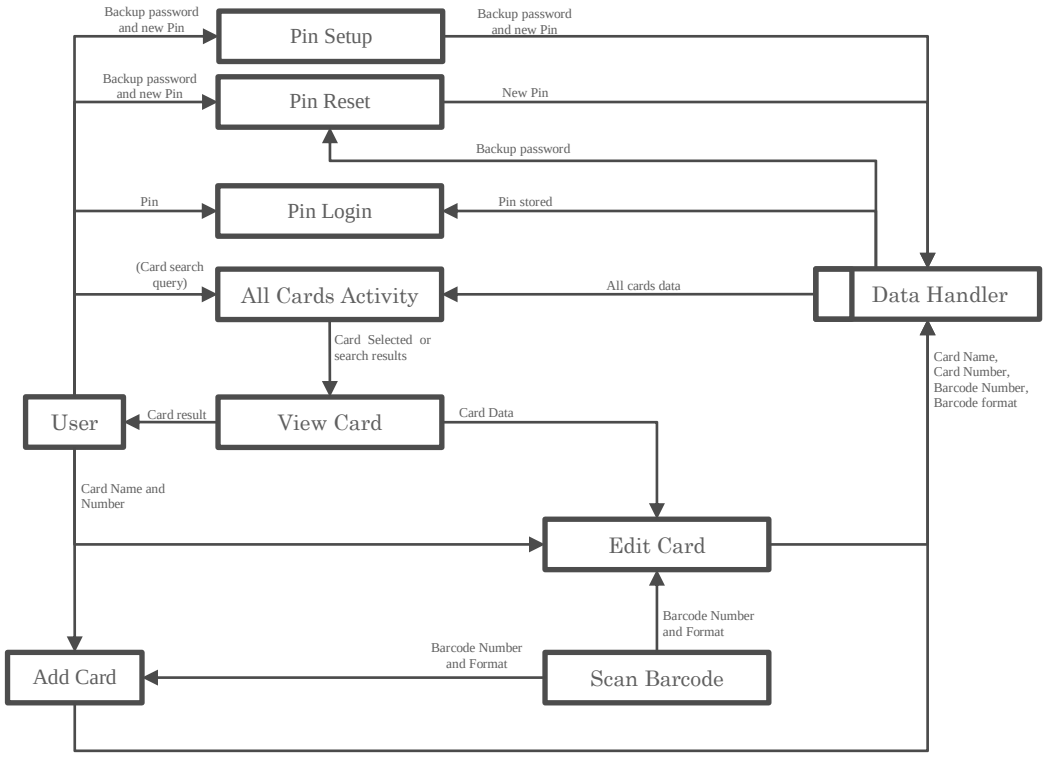

Figure 4.3: Loyalty Card Application Data Flow Diagram

Diagram (Fig 4.3) states user first inputs new pin and backup password in the pin setup activity, this is only done on the first run of the application or when the pin is not set for the system. The inputs are then handed to data handler which may do some further processing on the inputs, such as generating hash for pin and password. When pin is set in the system, the user can enter the pin at the pin login stage to enter the application. Upon login stage, a hash of the pin entered is obtained and compared with the existing pin hash in the system provided by the data handler. User then always has the option to reset the pin where backup

password input will be matched to the backup password stored in the system provided by the data handler. Pin reset activity also takes in the new pin that is handed to the data handler which may do some further processing on it.

All cards activity obtains all the cards data from the data handler. User can select a card from all cards with the option to search for individual card(s). The card selected is sends the search results to the view card activity which will reveal the card result to the user. If the user decides to edit that card then edit card activity fetches the card data from the last activity. The user can input the card name and number to edit the card. Moreover, if the user wish to scan the card then barcode number and format are sent to the edit card activity from the scan barcode. Add card is similar to the edit card with the only exception that it does not receive any existing card data from view card. New or updated card details are sent to the data handler.

4.4 State Machine

State machines model the changing states of objects and events that cause these state changes in a system. Models consist of states and transitions. Transitions (arrows) represent how states change and events are noted across the transition arrows which trigger the state changes. For state machine diagram of loyalty card application please refer to appendix B1.

The state machine diagram of system shows that it covers all the requirements points as found in section 3.1 of this document. It shows the different states in our system and the events that trigger them.

4.5 Graphical User Interface

A user interface for a system allows the user to interact with the program. It is vital that the interface of the system is simple and does not include too many options for the user as this may deter the user from using the system. However, the interface must provide the full functionality of the application.

Because our system consists of smartphone application, we only focus on the graphical user interface of the application. For graphical user interface designs of loyalty card application please refer to appendix B2.

4.5.1 Pin Setup

Upon starting the application for the first time or if the pin is not set in the system, the user will be required to setup the pin. This activity will ask the user to setup a new 4 digit pin for the system which will be used to authenticate the user. User will be required to enter the new pin twice as they will be used for verification and it is important to make sure they are correct. User will also be asked for a backup password which will be used to reset or change the pin. It is vital that the pin is stored securely as it will be used for authentication purposes.

4.5.2 Pin Entry

To authenticate the user, pin entry will be required to enter the application. User can type in the pin and click on the 'proceed' button to continue. If the pin is correct the user will be shown all cards activity, more details in section 4.5.4. Pin entry will require some matching algorithm to the user pin stored in the system. As the pin will be stored as a hash, the hash for pin entered by the user will be required to

be generated each time user clicks the proceed button and checked if it matches to user pin set in the system.

User will also be provided with the option to reset the pin by selecting the 'reset pin' button which will take the user to the Pin Reset activity.

4.5.3 Pin Reset

In this activity the user will be able to reset the pin for the system using the backup password that was set during the pin setup. User will be required to enter the new pin twice as it will be used for verification and it is important to make sure it is correct. Given the password provided by the user is valid and the new pin meets the pin requirements, upon selecting the 'save' button the user will be taken to the pin entry activity. The new hash for the new pin will be required to be stored in the system, so it is important has functions used throughout the system are the same. Otherwise the pin stored would not match the pin entered and will lock the user out of the system and making the whole application pointless.

4.5.4 Cards Activity

Here the user will be able to view all cards in the system. This will also be a main activity once the user has been authenticated. Main activity acts as a central activity. Other activities in the application are navigated from here: Add Card, View Card. As a main activity, this will make sure that when the back button is clicked in this activity it will not go back to the pin activity and pause the application instead.

User will be able to add cards here using the add button. Keeping with the android theme, the add button is shape of a 'lollipop' placed at the bottom right of the screen.

A list off all the cards in the system will be provided here. A custom list view will be used. Selecting a card will open the View Card activity. View card activity will be sent the card data to be shown on screen. From a security perspective, data sent within the application activities should not be accessible by other applications on the device.

User will have the option to search for cards on the top, using the search in action bar. Search results will be shown underneath in the cards list.

Also in the action bar, the user will have the option to reset the pin. This will take the user to the pin reset activity.

4.5.5 Add Card

A self-explanatory title, here the user will be able to add the card. User will have option to scan the card barcode by clicking on the 'scan barcode' button. This will take the user to scan barcode activity as seen in section 4.5.8 of this document. If the card is successfully scanned, card number field will be automatically filled with the card number obtained from scan card activity. User will then only need to manually type the card name in the card name field. User will be able to manually type and/or edit the card number in the card number field. User then can click on the 'save' button to save the card and the user will be navigated to the Cards activity as seen in section 4.5.4 of this document. Alternatively, if the user clicks back button a dialog box will appear asking to confirm discarding changes.

Confirming choice will take user to the Cards activity, else selecting to cancel will remain in this activity.

4.5.6 View Card

In this activity the user will be shown the card. A large barcode will be shown if the barcode data is available. Card number and card name will also be shown. User will have the option to 'edit card' which will take to the Edit Card activity. Upon clicking back button Cards activity will be shown.

4.5.7 Edit Card

This interface is very similar to the Add Card activity but with an extra option to delete the card.

Card name and card name will be prefilled fields with the card data which can be edited. User will have option to (re)scan the card barcode by clicking on the 'scan barcode' button. This will take the user to Scan Barcode activity. If the card is successfully scanned, card number field will be automatically overwritten with the card number obtained from Scan Barcode activity.

User then can click on the 'save' button to save the card and the user will be navigated to the Cards activity. Alternatively, if the user clicks back button a dialog box will appear asking to confirm discarding changes.

If user chooses to delete the card, 'delete' button, a dialog box will appear asking user to conform their choice. By confirming it, the user will be navigated to the Cards activity. Upon cancelling the option on dialog box the user will remain in this activity.

4.5.8 Scan Barcode

This activity uses the phones main camera, rear camera, to scan the barcode. User is also shown guidelines and a red laser bar on the screen to place the barcode under to scan it. No click of buttons will required, the application should auto-focus and scan the barcode automatically. When barcode is successfully scanned or if the back button is clicked the user will be taken to the preceding activity. Preceding activity will be either Add Card or Edit Card.

4.6 Third-Party Content

This project uses some 3rd party content which allows to meet the requirements and specifications. Such content improves productivity in project. Also saves development time and shortens it.

4.6.1 ZXing Library

The project uses 'ZXing library' (10). It is an open-source barcode image processing library. One of the main benefits of this library is that it is implemented in Java. In the code, libraries may be referenced as "ZXing" or "com.google.zxing.*". The project has been designed with latest version of ZXing 3.2 which was released on 15th February 2015. Features used from this library in this project:

1. Barcode scanning functionality, and

2. Generating barcode images.

From a security perspective, any library used should avoid using internet and external resources of the web to use the data in the application. ZXing library is local library and does not require internet access to offer the features that are used

in the project. This allows the card data to remain within the application environment.

4.6.2 Icons

Some icons used in this project are from an open source directory. They are open to be used in private and public projects. (11) Icons used for in project:

1. Application icon
2. Default card icon in Card view, Add Card, Edit Card when no barcode image present.

5. Implementation and Testing

This chapter reveals how the project was developed and implemented using our design plans as seen in Design, chapter 3. This chapter also details or any changes from design, specification and requirements. First system development approach and system functionalities will be discussed which will be followed by issues in implementation and any changes due to it in the system and specification. Results of testing conclude this chapter.

During development and testing process security was kept in mind. This allowed production of secure application environment and better integration of the system work packages, as discussed under section 5.1.

5.1 Development Approach

Please refer to appendix section B3 for class diagram of the implemented system and appendix section D for source code.

Class diagram was not included in the design of the system as iterative development approach was taken. It was possible that system may not fulfil some of the low priority requirements in time which would mean altering the system design. System specification and features were divided into smaller work packages and each sprint in development meant successfully completing the work package. Order of developing work packages was determined using the requirements and specification as defined in the chapter 3 of this document. Dependency of work packages on each other and general security structure of application also partly

defined the order. Work packages for this project and the order they were developed are as follows with requirement codes in the bracket next to them:

1. Cards stacks to show list of cards: Cards Activity. (C1,C2, C3, S7)
2. Framework of application, store and preview cards: Add Card Activity; View Card Activity. (C3, C4, C5,C6, S7)
3. Database schema implementation for storing User and Card data. (C1,C4, C6, S1, S2, S7,)
4. Pin Protection implementation: Pin Setup Activity; Pin Activity. (S1, S2, S4, S5, S6, S7, S8, S9)
5. To be able to reset Pin: Pin Reset Activity. (S3, S7, S9)
6. To be able to edit or delete card: Edit Card Activity. (C4, C5, S7, S9)
7. Application resumes asking for pin. (S4, S7)
8. Barcode Scanning and storing details for cards: Barcode Activity. (C7, S7)
9. Barcode Generation for viewing Cards. (C8, S7)
10. Card searching capability. (C9, S7)
11. Improve interface and make it more user friendly: android lollipop theme; introduce back button for activities in Actionbar. (C13)
12. Show error messages for invalid input entry. (C10)
13. Show confirmation dialogs box: discarding changes to Editing Card and Adding Card activities; deleting card. (C12)

14. Keyboard functionality: only show specific keyboard for use such as only show number pad for entering pin; when click on whitespace or blank space hide keyboard. (C11)

15. About Page for application: About Activity – extra feature not in requirements.

Work packages above numbered from 1 to 7, 8 to 9 and 10 to 14 tackle high priority, medium priority, and low priority requirements respectively. Although work packages fulfilling the low priority requirements were implemented in order defined above, but did not require being in any specific order as they are not dependent on each other. Work packages Design diagrams (Fig 4.1 Use-case, Fig 4.2 System Architecture, Fig 4.3 Data flow, State Machine – appendix B1) and graphical user interface designs (appendix B2) were used in implementation for the system.

Security requirements were adapted from an early stage in development as demonstrated by the work packages. This allowed the system to have secure transfer and manipulation of data within the application from the beginning. This was important as at every stage of development, part systems being developed were assessed if they could be breached. If a part of the system was breakable then fix were found and implemented before moving on to the next work package. Such implementation meant that the system was more robust although such development did provide challenging as developer required to access their code and work from a functional and security point of view.

5.2 System Functionalities

In android development, each activity inherits java classes and activities can override the methods for that activity in android application to work. These methods inherited from the Android Activity class and part of the android lifecycle: onCreate(), onStart(), onResume(), onPause(), onStop(), finish(), onDestroy(), and onRestart(). (12) In the application development most of these methods were overridden. Android activities for this project and their java classes:

1. PinActivity:

   Upon starting the application for the first time, PinActivity is started and is set as default activity. It uses the DataHander class to obtain user pin, if it exists. DataHandler class has functions to create the database and tables which are run upon installation of application. For implemented database schema please refer to appendix B7.

   Pin entry field in GUI layout XML has input type set to "numberPassword" so the entered characters appear as dots. A function isUserPinSet() returns Boolean if the pin is set for the user. If the pin is not set then PinSetupActivity is opened using intent. These checks are done in the background. However if the pin is set then the hash of pin entered is obtained using the generateSHA265() method in Cipher class under Crypto package. Cipher class consults resource (13) for framework of the generateSHA265() method.

   When 'proceed' button is clicked: obtained pin hash is compared with the one stored in database and if match then CardActivity is opened using intents with flags.

Flags make sure that the CardActivity is the main activity to which other task would return to. So when back button is clicked in the CardActivity the application is closed and not return to the PinActivity. Flags added to intent:

- FLAG_ACTIVITY_NEW_TASK – new activity (CardActivity) becomes on the top of history stack
- FLAG_ACTIVITY_CLEAR_TASK – current activity (PinActivity) is cleared

Clicking on the pin reset button opens the PinResetActivity using intents.

2. PinSetupActivity:

   This activity will ask the user to setup a new 4 digit pin for the system which will be used to authenticate the user. Hash is generated using the Cipher class for the backup password and pin entered. User will be required to enter the new pin twice, so two input text fields were implemented for pin. For both pin and password InputValidator class is used to return Boolean values for valid the input parameters. Part of the Helpers package, InputValidator class has methods: isValidName(), isValidNumber(), and isValidPassword(). Password and pin are checked using this before saving the user data. DataHandler class is used to store the user data in database. Once the data is successfully stored the PinActivity is opened.

   Password and pin entry fields in GUI layout XML have input type set to "textPassword" and "numberPassword" respectively so entered characters appear as dots.

There was design change for PinSetupActivity, we will discuss it in section 5.4 of this document under changes.

3. PinResetActivity:

   To reset the pin for the system, backup password is asked. Password and pin entry fields in GUI layout XML have input type set to “textPassword” and “numberPassword” respectively so entered characters appear as dots. See appendix section B5 for different keyboards specific to inputs type of text fields.

   DataHandler class is used to obtain the password of system and is matched with the hash calculated of the password user entered using the Cipher class. Hash is compared because the passwords and pin are not stored in plain text format but in hash format. Given the password provided by the user is valid and the new pin meets the pin requirements using the InputValidator class from Helper package, upon selecting the ‘save’ button the data is updated in the database and PinActivity is opened using intents.

4. CardActivity:

   A cards List view is embedded in the GUI and uses custom item view to show cards data (name and number with generic card logo). Cards objects from the database using the DataHandler class initialises the list view using Cards class and CustomListAdapter class. We had to filter the custom objects, cards, so required a custom filter for our filtering constraints. This is used for searching cards and showing cards in alphabetical order in the list. Actionbar menu items include search, pin reset and about page options. Search item was set always visible and other items are set to hide in the

GUI. When a card is clicked, ViewCard activity is opened using intents passing the values of card. Add button with plus sign placed on bottom right of the screen opens the AddCard activity using intent.

5. AddCard:

   As card scanning is offered in this activity, upon clicking on the scan button SimpleScannerActivity is opened using intent and barcode activity returns the results for barcode number and barcode format. Card number field is filled if the barcode data received with a card number. Card name field still requires to be filled manually and InputValidator class is used to ensure the name of card is valid. DataHandler class is used to store the cards data: card name, number, barcode number, barcode format. Card can be stored with only the card name and number, without the barcode data, to ensure this barcode data is not required to continue to save the card in saving cards checks.

   Upon clicking back so it shows the dialog box to confirm discarding the changes, the button was overridden. Upon user confirming the choice finish() method is called on AddCard activity and previous activity from the activity stack is resumed which in this case is the CardActivity.

6. ViewCard:

   This activity obtains the card data from the CardActivity. Bundles are used in conjunction to receive the data. Card name and number is displayed in the text view. If barcode data is available then barcode is generated using the GenerateBarcode class from the Helper package. GenerateBarcode uses ZXing library (10), takes barcode number and the barcode format as

parameters and returns a bitmap barcode image. This image is then viewed in the predefined image viewer in the ViewCard activity.

Upon clicking back the CardActivity is resumed from the activity stack. When 'edit card' button is clicked the EditCard activity is opened and card data is sent to it using intents.

7. EditCard:

   This activity obtains the card data from the ViewCard activity. Bundles are used in conjunction to receive the data. Card name and number fields are set to the data obtained. These edit fields use InputValidator to ensure the card name and number are valid before updating the details. EditCard uses DataHandler class to update the data in the database.

   Upon clicking on the scan button SimpleScannerActivity is opened using intent and barcode activity returns the results for barcode number and barcode format. Card number field is set to new scanned barcode number if the barcode scan was successful and data was received.

   Back button is overridden: so it shows the dialog box to confirm discarding the changes, upon user confirming the choice finish() method is called on EditCard activity and previous activity from the activity stack is resumed which in this case is the ViewCard.

8. SimpleScannerActivity:

   This activity is part of the Barcode package and uses ZXing library. (10) A third party 'barcodescanner' module is available on github (12). Logistics and framework of this module are used in Barcode package.

SimpleScannerActivity returns results for barcode number and barcode format to the preceding activity. To make sure that the user is not stuck in the scanning barcode mode, checks are performed in the background: once the card is successfully scanned and barcode data is collected, the preceding activity is resumed. Preceding activity is either AddCard or EditCard.

9. AboutActivity:

   This was added as an extra feature for the user. It requires no entry of data but is an activity to just present project in the application. It reveals the developer details and acknowledgements for the sources used in application. Contains a listview with a onClick listener that allows the user to click on the sources from the list which opens the webpage for the source where further information resides. As the application is paused when clicked on source, we had to ensure that upon resuming the application PinActivity is opened.

Above are all the activities of the project outlined with classes and methods they rely on.

5.3 Implementation issues

During the development of any software, some issues tend to rise. So the developer has to work on those resolving the issues which can lead to changes in design of system. During issues arising in the project development, security took primary concern while trying to solve and overcome the problem. This made sure that the issues were solved with security elements and did not drift into a non-secure feature fix for the problem.

5.3.1 Resuming application user authentication

When application was opened not from background, PinActivity appeared because it is set as default activity, however it was not enough. When the application was paused or in background before, it would resume to the activity that was paused. To ensure that upon resuming the application the PinActivity is opened to authenticate the user we had to look into android lifecycle. Android lifecycle defines properties of activities in Android:

- Call onCreate when created and onResume when the activity is resumed
- When another activity is opened, current activity calls onPause and goes behind the new activity on the activity stack.
- When the application is paused or in background, activity application was in calls onPause before going into background, and calls onResume when comes back from the background.

These points revealed that onPause and onResume methods required overriding for each activity in the application to ensure that user was asked to enter pin upon resuming the application. A temporary log had to be kept for when user paused the application so upon resuming the application PinActivity would be opened. To achieve this:

1. Define private Boolean in activity class and initialise to false:

```
private boolean cardPin = false, openSomeActivity = false;
```

2. When open another activity within the application set the:

```
openSomeActivity = true;
```

3. In onPause method set cardPin to true if leaving the application otherwise if opening another activity within application set to false:

```
@Override
protected void onPause() {
        super.onPause();
        //so when user resumes the app it asks for pin again.
        // if opening another activity then false
        if (openSomeActivity) {
            cardPin = false;
        } else {
            cardPin = true;
        }
}
```

4. Upon resuming the activity check if it requires to ask pin, if so open PinActivity using method openPinActivity().

```
@Override
protected void onResume() {
        super.onResume();
        if (cardPin) {
            openPinActivity();
        }
        openSomeActivity = false;
 }
```

These four steps of code had to be implemented in every activity of the application except the PinActivity itself. Such solution was relatively easy to replicate across all activities in the application however required extreme logical reasoning and thinking with aim to build secure system in mind. Solving the problem required reasoning for how the user would interact with the system and what the system would do logically behind the scenes. This allowed identifying and implementing the four steps in system as shown above and achieving the secure solution to the problem.

5.3.2 Update Card reload

When card was updated in the EditCard activity and saved, the user was navigated to the CardActivity instead of the ViewCard. However selecting the back button

would pause the CardActivity and take the user back to the EditCard activity. To avoid this we used the same flags with intents as used when opening CardActivity from the PinActivity. Flags added to intent are: FLAG_ACTIVITY_NEW_TASK, and FLAG_ACTIVITY_CLEAR_TASK. (14)

Implementing a solution to this required researching into the solution too if it would jeopardise the security of the system. It was only implemented as solution because it did maintain the system security. Solution to use flags just resets the activities stack of application. This is done by android system however it is possible to re-define it as we have here.

5.3.3 Avoid SQL injection

To avoid invalid data entry we used regular expression in the InputValidator class. Furthermore:

- Only numeric keypad shown for numeric data entering. Set text type of text field to “number” in XML of text field in activity. So even if a user tried to break using a custom keypad, all other characters than numeric will not be allowed.

- Generic keypad shown for other data entry. Set text type of text field to “textPersonName” in XML of text field in activity. So even if a user tried to break the system using a custom keypad, all other characters than a name would have will be denied.

- Set error messages on the text field if invalid characters were tried to be saved

All these precaution were taken to avoid SQL injection in the database and to have a sealed system. Avoiding SQL injections is a huge security requirement in the system. This would make sure the integrity of the database and the correct storage and accessing of the database. Implementing this was challenging as there were multiple steps taken to avoid SQL injection: in input validator through code and in the graphical user interface of the system by defining specific input and keyboard types.

5.3.4 Maintaining Lollipop theme for older versions

Android Lollipop is version 5.0 API level 21. Google introduced as new features among them is material design. To maintain a similar theme of application for android devices running pre lollipop down to android Jelly Bean, version 4.1 API level 16, we had to add system dependency to compile with 'com.android.support:appcompat-v7:21.0.+' and some code was required for app theme. For version pre-lollipop devices custom layouts were required in addition to custom toolbar for application.

In CardActivity, to allow the add button to work on the older versions of android, pre API level of 21 – Lollipop, a custom layout was required and button was created using a clickable image view which holds the add button sign. This meant that CardActivity required two different XML layout files one for lollipop and one for other version of android. Other than the additional add button image view, rest of the layout is same for both layout versions.

Actionbar item compatibility across different android versions is also discussed in next section, see 5.3.5.

5.3.5 Actionbar back buttons

Adding the back buttons in the Actionbar of activities and for secure application meant that parent activity of the activity had to be manually defined. In Android manifest file, to add back button for an activity we had to take couple of steps:

- Define parent activity in <activity name> using following code format:

```
android:parentActivityName="<activity name>"
```

  for example in ViewCard activity:

```
android:parentActivityName="com.project.lockcard.CardActivity">
```

- To have same functionality support in pre lollipop we had to add extra code with meta tags, using ViewCard as an example again:

```
<meta-data
       android:name="android.support.PARENT_ACTIVITY"
       android:value="com.project.lockcard.CardActivity" />
```

Back button in the Actionbar for EditCard activity could not be implemented. This is because parent activity ViewCard requires card data sent to it as parameters and because parent activity is reset when card is updated and saved.

Adding back buttons required to defining the parent activities and so did not pose a threat to the system. In this case functionality compatibility was considered for other versions of Android in addition to security elements.

5.4 Changes and Additions from design

5.4.1 Enter password twice

This change refers to requirement code S8; please refer to section 3.2.2 Table 3.2 of this document for more details. PinSetup activity requires user to enter password

twice instead of once as in design. As the password is used to reset the reset pin and therefore is as important as the pin. Backup password will be used for verification and it is important to make sure it is correct. Similar to the pin it is entered twice by the user in PinSetup activity. Both entries of passwords are checked if they match among other pin checks before proceeding to save the user authentication data which is vital to security of system.

Additional checks are also performed to check if pin and password entered are the same, if so error is shown that password and pin cannot be the same. Backup password must be at least 4 characters long and contain 1 numerical number. Implementing such checks required some work on text listeners and great deal of concentration was required to make sure correct error messages were shown and there were no false positives.

5.4.2 Hiding Keyboard

Keyboard appears when click on the text field by default. However an extra feature was implemented: keyboard hides when clicked on whitespace or blank space away from text field. (15) This aims to improve user interaction and experience with the application system and make it more users friendly. If this was not implemented then to hide the keyboard the user would have to click on back button, which is still an option to use if wish to. This feature is implemented in: AddCard; EditCard; PinActivity; PinResetActivity; and PinSetupActivity activities.

This feature was relatively easy to replicate across all the activities required in. It did not pose any security risks to the system as did not manipulate the data in any way. Development of this problem had to make sure that upon re-focusing on the text field the correct keyboard was shown and did not provide with a

different keyboard then last time. Text type was defined in the GUI and made sure keyboard shown was consistent. Changing the keyboard would allow non-allowed characters to be typed into the text fields; however a second tier security option to validate the inputs and to only allow specific text type was applied.

5.4.3 Showing barcode in other activities

According to the requirement and specification barcode would only be generated in ViewCard activity when the user would view the card. To improve user experience the barcode is also generated and shown in AddCard when user scans the barcode and in EditCard activity. In EditCard activity if existing barcode data exists for the card editing then barcode is generated and show to the user, if user scans a different barcode then barcode data is updated and new barcode is shown. This helps user to know that barcode scanning was successful and provides the proof of it during the adding or editing the card. ViewCard does generate and show barcode as requirements state.

Such an extension to currently used module, barcode generating, did not pose any security risks to the system. As every work package during development required security evaluation, and barcode generating was pre-checked so was allowed to be used for other activities.

5.4.4 Screen rotation settings

Aim of screen rotation is to allow user to allow use the full length of screen on the device and provide more immersive wide interface experience. However, for loyalty card application it is more suitable to hold the phone in normal default orientation of the device and disallow screen rotation. Another reason to do this was when the

user is trying to scan the barcode from the device; the user will try to rotate the device in multiple directions to allow the scanner to scan the barcode. This would not be possible and decrease the chances of scanning when the phone automatically rotates the application screen according to user's movement of the phone. Exception to this screen rotation is while scanning the card as it can help to scan the barcode more successfully as it allows the screen to show the camera image in portrait mode. The barcode scanning guidelines are also larger so a larger barcode can be scanned this way.

5.4.5 About page

This was added as an extra feature to improve user experience. It is an activity to present project information in the application to the user revealing developer details and acknowledgements for sources used in application. This additional feature was implemented after fulfilling all the requirements and specification. It does not affect other requirements and specification. Security features had to be taken in this activity too: opening pin activity when resuming application from background. Please refer to appendix B6.e5 for screenshot of implemented and tested about page.

5.5 Testing

To ensure system works it is vital to test it. System must meet all the requirements defined in section 3.1 of this document. We will discuss the various forms of testing performed on the system. Security of the system was a main concern and parts of system were tested based on security as well as functionality that they provided as a feature in the application.

5.5.1 Unit Testing

Unit testing was mainly carried out during the development of the system. It was performed using the required inputs for each function and method in the system. Reasons for these tests were:

1. To ensure every function worked properly as supposed to.
2. Parts of system had to work together due to dependencies. This ensured every function worked properly with other features. Without unit testing if would have been impossible to test full system.
3. Each part of system did not pose any security risks. This made sure data only flowed within the application and was not manipulated in any way by external applications. Ensured correct functioning of the system according to the security and other requirements of the system.

At end of each sprint of development, work package completed was testing before moving on to the next sprint. This made sure every part of system was tested before integration. For example, there were some issues with application not resuming to ask for pin. Had this not been discovered early during unit testing could have led to a major security flaw in the system. It would have been difficult to find where the issue lied later in the development stages.

5.5.2 Non-functional Testing

As loyalty card application is aimed to be a consumer product, it requires to meet some quality attributes. Following are the main system quality attributes and reveals if the application fulfils them:

1. Efficiency: application performs the functions as supposed to without wasting system resources. Android memory management system helps with it. (16) Application follows the android lifecycle rules and guidelines. System RAM allocation is stable and does not exceed 20MB. Please refer to appendix B5 for results to memory allocation.

2. Modifiability: the application is modifiable using the code and updates can be released for the application. Within the application cards stored can be edited and the pin can be changed.

3. Portability: application produced is portable as it can be installed on different android devices. Only constraint is that Android version of device requires to be Jelly Bean (version 4.1) or newer.

4. Reliability: the application performs as it is supposed to. It scans and stores loyalty cards while requiring user authentication to view them. Application is complete each module performs and has been developed fully such as scanning and generating barcode modules perform as supposed. Accurate details of the card are shown: card name, number and barcode. Pin and password must be accurate otherwise system shows an error message. Application does not crash or get stuck unexpectedly.

5. Security: the application requires pin authentication and stored the loyalty cards securely. Unauthorised person cannot or is highly unlikely (10,000 possibilities for pin, and over 60 million possibilities for password) to enter the application and view cards. More details and how system meets security requirements are discussed in section 5.5.3.

6. Understandability: the application is easy to use and clearly shows the headings and options to the user. It follows the android system GUI so it is easy for the user to use it as they are familiar with android devices interfaces and applications provided by Google such as Google Chrome and calendar. Buttons and the options in the application are self-explanatory, for example an add sign button means to add something, in this case a loyalty card.

5.5.3 Requirement Based testing

In this subsection of testing, project is compared to the list of requirements as stated in the requirements and specification, chapter 3. Justification of system meeting the requirements is shown in appendix B6 which contains the screenshots of the application. Not all requirements have justification proof screenshots such as requirement "S2 – Pin and password should be stored as digest (hash) in database."

Content requirements

- C1 – The application should allowing storing Loyalty cards: Clicking on the add button opens the AddCard activity. A card can be scanned using scan button and card data can be entered to be saved in the system.

- C2 – Application should display all the stored cards in alphabetical order: application displays all the stored cards in an alphabetical ordered list.

- C3 – User should be able to select a card form the list to view individually with its corresponding card details: user can select the card from the alphabetical order list to view it.

- C4 – A card can be edited and saved with updated details: While viewing the cars, clicking on the 'edit card' button loads EditCard activity to edit the card. Updated card details are saved when user selects 'save' button.

- C5 – A stored card can be deleted from the application: while in editing card state user can select to delete the card. User is asked to confirm dialogue box card is deleted from the application.

- C6 – User should be able to store card details without scanning barcode: can add card details manually without scanning the card and save the card details. Scanning is not compulsory.

- C7 – User should be able to scan barcode of the card and store it: user can to scan barcode of the card and store it. While adding card scan button can be clicked to scan the barcode.

- C8 – Application should be able to generate barcode for card if barcode data exists for stored card: upon viewing the card if the barcode data exists barcode is generated.

- C9 – User should be able to search for cards using their names as search query: user can search for a card using its name in search baron the top. Upon searching for the card the card list underneath is

updated automatically showing the card results. The card can be selected to be viewed.

- C10 - User should get some feedback in form of error dialogs or toasts for invalid inputs: entering invalid entry such as invalid pin or password, card number or name, error messages are shown. When card is successfully saved a toast appears to reveal that the card has been saved.

- C11 – Keypad for input should be relative to the input field. Numeric keypad for pin and card number input fields; Password keypad for password entry; and general keyboard with predictions are shown for names and other fields. See appendix B5 for more details, some screenshots are also provided under section B6.c11.

- C12 – Application should ask for conformation: discarding card details or deleting a card requires user confirmation. Dialogue box appears to confirm user options. If a confirmed by clicking on the dialogue box the action is taken. Otherwise if selected to cancel on dialogue box, it disappears and stays in the current activity.

- C13 – Graphical user interface of the application should complement the system interface as defined by the operating system: user interface in application is not customised to look different than an android system interface would look like. See appendix for details how it compares to existing android Google application such as Google Calendar.

Security requirements

- S1 – A pin should be used to authenticate the user: four digit pin is required to be set upon first time opening the application, or if the pin has not been setup. Backup password is also required to be setup.

- S2 – Pin and password should be stored as digest (hash) in database: they are stored as digest.

- S3 – Pin reset activity should ask for password used whilst setting up pin during pin setup: it does ask for password and if it matches then only it is allowed to reset the pin otherwise error message shown.

- S4 – User should be asked to enter Pin to authenticate user upon opening or resuming the application: PinActivity is always shown upon opening or resuming the application.

- S5 – One should not be able to create shortcuts to a specific activity within this application other than the pin activity: creating shortcuts to activity within the loyalty card application is not possible. A shortcut can only be made to the pin activity in the application which requires user to input the pin. Appendix A2 details how activity shortcuts are made: at step 4 of this process for LockCard, loyalty card application only pin activity is shown and other activities are not so a shortcut cannot be made to bypass the pin.

- S6 – No more than five pin entry tries should be allowed at a time: after the fifth incorrect pin entry system shows error message and does not allow user to enter pin again. When the user leaves

application and resumes it, one more pin entry try is given to successfully proceed in the application. To reset such pin entry attempts user should exit the application fully (close) and then open again.

- S7 – No entry should be made to android logs of secure content: Logs are not used in code so there should not be logs created for application content. This ensures other applications cannot read the logs of this application that could allow jeopardise the system. No screenshot are available for this.

- S8 – Backup password to reset should be at least 4 characters long.: backup password is required to be at least 4 characters long otherwise error message is revealed.

- S9 – Typing fields should be checked for invalid characters: input fields are checked for invalid characters and error messages are shown if so. Unless an error is fixed the user cannot continue to save or update the details in application. This requirement overlaps with content requirement C10. Hence please refer to appendix B6.c10 for screenshots for this. This is to avoid repetition.

Extra features

Extra features implemented as explained in section 5.4 were tested during the development. Below is the list of extra features and they refer to other appendix sections for justification and implementation.

- E1 –Entering password twice in pin setup: please refer to appendix B6.s8 as this overlaps with pin setup requirements with requirement code S8.

- E2 – Hiding Keyboard when click on black space or away from text fields. Please refer to appendix B6.e2.

- E3 – Showing barcode in add card and edit card: Please refer to appendix B6.e3.

- E4 – Screen rotation disallowed: when device is rotated the screen only rotates while scanning the barcode. Please refer to appendix B6.c7 for scanning barcode example. All other activity do not allow screen rotation.

- E5 – About page: when user selects the 'About' item from the menu in the cards activity (CardActivity) about page is shown. Please refer to appendix B6.e5 for screenshots.

6. Professional Issues

During design and implementation of the project, great concern has been shown to abide by the Code of Conduct & Code of Good Practice which is issued by the British Computer Society (BCS).[7] With any project, code of conduct should be abided by because if not can have serious legal and ethical implications. Rules also help protect work an intellectual property from being used unlawfully. Using material without authorisation of the owner is regarded unethical. Great care has been taken in this project to make sure any Open-Source code or libraries used are explicitly stated. This project consists of: my own work; Open-Source libraries; and Open-Source code provided on the internet. This has allowed speeding up the development process significantly and tweaking it to meet our requirements for the project. This has also allowed producing better quality of work and thus any external work used is referenced to recognise it. The software consists of my own work unless explicitly stated so.

Security and privacy is one of the main focuses for this project and great deal of work has gone into keeping the contents of the application secure.

Application produced is fully compatible with android accessibility features and settings: talkback service; magnification gestures; colour inversion; colour correction; high contrast; and text-to-speech. Although the last feature, text-to-speech may not be wise to use for security reasons but if a user cannot type then they can speak into the device and still use the application. These features allow users with special needs and disabilities to use the application as well.

[7] www.bcs.org/upload/pdf/conduct.pdf

7. Evaluation

7.1 Application Evaluation

LockCard, the application produced was compared with the system requirements and tested as detailed under section 5.5. It was compulsory that the system met the requirements and non-functional requirements which were analysed in section 5.5.2.

7.1.1 Limitations

No project is perfect. The following details the limitations for the project. This subsection will also focus on limitations as mentioned under specifications section 3.2.3. Solutions to these limitations are offered under the future work section.

- Application produced cannot scan two-dimensional barcodes nor generate two dimensional barcodes. ZXing library used to scan and generate barcode only supports linear, 1D barcodes. So if a loyalty card has a multidimensional or 2D barcode, user will have to manually enter the card number as shown in the card. Although this type of barcode is not common for loyalty cards, it is still a possibility.

- If the loyalty card requires a swipe or has a swipe strip this cannot be integrated into the application of the device. In this case the cashier or user has to manually enter the loyalty card number in retailers system at checkout. This is the case for Nando's card however Nando's offer their own phone application which includes card details in it.

- LockCard, application produced for our project, only accepts numbers for card number entry. According to research it was evident that loyalty cards have numbers and do not have any alphabetical characters in them. If a loyalty card does contain an alphabetical or non-numerical character in the loyalty card number, then user it will not be able to store it. This is because system only allows numbers to be stored for the card number field.

- There is no functionality available to save a favourite card. Although search service is available to search for cards it may not be enough during hurried transactions.

- A card can only have one card number and barcode that can be scanned and stored in the application. So if a card has more than one set of numbers on it then the user either stores only one of them, or stores two instances of same card with different name and numbers. If there is only one barcode then the barcode represents the card number and upon user scanning the card this will be obtained automatically.

- Some barcodes may not scan so user is still able to store the card number of the card. Unfortunately without scanning the barcode it will not be possible to generate barcode from user input as barcode type cannot be manually saved. This is mainly because user is not expected to know the type of barcode by just looking at a barcode. So to generate barcode, the user must save the card by scanning the barcode rather than manually typing the details.

- Regarding more with security limitations, the password of the system cannot be changed. Only pin is allowed to be changed. To change the

password would require resetting the application and setting it up and adding cards again. To solve this, an extra feature could be added: user can be sent an email or SMS message with link to reset the password. Alternatively user could be asked security questions to which answers can be used in conjunction with the correct pin to change the password.

7.1.2 Security

Security elements of this application are similar to a banking application for android devices. No other loyalty card application for android exists on the market that allows such security level with pin protection. We have seen in detail how activity shortcuts can be made in android bypassing the pin authentication. Such a case is with an existing application Beep'n Go. Bypassing the pin authentication is not possible in LockCard application produced for our project.

Producing a secure application was difficult because it required security elements as featured in banking applications which are produced professionally with a team of developers, but had to be done in short period of time and by a single developer with limited experience in developing. Due to these reasons basic security elements were introduced in the application, however more features could be added which are discussed under future work in the next chapter.

7.1.3 Overall

Application is compatible with most android devices: running Jelly Bean; Kitkat, or Lollipop versions of Android.

The aim of the project was to store loyalty cards security with some sort of user authentication and not allow unauthorised access. LockCard application does

meet this basic requirement. This does not imply that application is perfect but has areas for improvement; these are discussed in next chapter under heading future work.

7.2 Project Evaluation

Development methodology used for the project ensured maximum flexibility to meet requirements. Planning was vital to the well-developed system and design stage helped in doing so. As with any project there were implementation issues, which are discussed in detail under section 5.3.

Background research of the project helped to understand the problems and in ways they could be solved. If more research was undertaken it would have helped to reduce the amount of issues faced during implementation of the project. Dealing with these issues required spending extra amount of time on the project. Issues such as 'resuming application to ask for pin authentication' required further research and took vital amount of development time.

Although the design section proved to be very helpful by providing design diagrams and graphical user interface design for the system. It helped speed the development process as developer required less imagination for interface design and more of implementation using designs of interface. It would have been helpful if a system flowchart was also designed during the design section. Such a flowchart would have helped to understand the flow of the application when user made a selection on the interface. Data flow diagram helped with this by revealing how the data would flow through the system but lacks system flow of application. Which

interface/activity to be shown was not very clear in the design and was left to the developer's imagination.

Short development sprints allowed working on one work package at a time. This made sure they were fully developed, implemented and tested before moving on to next work package. This allowed the developer to gain a refreshing experience of working on a new work package once previous one was completed. Adding work packages to the system allowed the system to be built features by features until all the work packages were implemented and a suitable application was produced that fulfilled the requirements. Timescale for each work package was not defined which allowed some flexibility during the development process. However this proved that developer was required to estimate in advance how long a work package might take according to the skill and effort taken to complete a work package. This flexibility proved somewhat helpful as a sprint could be short or longer depending on time availability. This still required some idea of when a work package will be completed. Project was aimed to be finished thirty days (a month) before the project submission deadline. However due to unexpected issues arising and time spent fixing errors this was stretched further that the aim but still finished before the project deadline.

Testing process in ensured the application was met system requirements. This also revealed any issues. Fixing errors in code and work packages at an earlier stage made sure that when the work packages were integrated together, the system would be free of errors. Although this was not guaranteed it helped minimise the errors.

The end result is the application which is easy to use with graphical user interface similar to android user interface. It allows storing loyalty cards and viewing them from the phone, saving wallet weight and space.

In the next chapter we will discuss the project conclusion and future work.

8. Conclusion & Future Work

8.1 Conclusion

Although the project was completed it was not without errors which were amended during implementation or testing. Other concerns remain however, such as database accessing methods. These methods may require some optimisation by rewriting them.

LockCard has substantial amount of integration with ZXing library. Having never worked on ZXing library before, using it in the project proved to be a challenge. Learning how to use it was important in order to get it to work. During implementation it was found that generating barcode required to store a list of barcode formats in the application to match them with the scanned barcode data. Fixing build errors required understanding the build structure of project and build tools. New android studio uses Gradle as build tool.

Graphical user interface for android required manually altering and adding XML code. An interface similar to android was only achievable using android developer guidelines. Rather than hardcoding string, the strings file was used to achieve an application compatible with android accessibility settings. Writing all the strings was very time consuming but a short amount of work to make sure users of all needs can truly use the application.

If more time was allowed to complete the project and/or a team of developers was behind the developing the application then more additional features could have been implemented.

8.2 Future Work

LockCard is not a perfect application and still has room for improvement. It is designed in such a way that extra features can be added to it. Referring to the limitations in the application it is evident that application can be improved to meet more commercial requirements. Possible improvements and how difficult they can be:

- Allow backing up of cards to a cloud storage or export cards. This would require some additional encryption of data to maintain integrity. The data will be required to be compressed to make it smaller in size so it can be transmitted over the network. User may require signing in with their cloud storage. This will require the cloud storage platform to access the data being uploaded to it. Most used cloud storage platforms, such as Google Drive (17) and Dropbox (18), synchronise data across all devices signed in. This could lead to backed up cards data made available across all devices user is signed in. This feature can be used for sharing cards within a group of authorised persons such as within a family or friends. Such an implementation will require encryption and decryption of data in the LockCard application.

- Add shopping list feature. This would expand the loyalty card application into a shopping and savings category. Implementation for this would be relatively easy as would require an additional activity or fragment view to add and view shopping list. Would require additional database functions in the DataHandler class. Regarding security of data stored, shopping list can be within the application so opening and accessing it will require the pin authentication. A shared shopping list can be expansion for a basic shopping

list which can allow a group of authorised persons, such as family or friends to edit the shopping list. This feature could allow the person to shop for household of family or friends residing with, in one go. This can be linked with payment integration.

- Payment integration for shopping. Allowing user to pay using their bank card also stored in the application in addition to collecting/using loyalty points. This feature will require background research into payments and integration into the application. Implementation can be more challenging task than other additional features discussed here because this will class the application into a shopping and finance sector. It will require storing bank cards data securely.

- Adding navigation buttons to all system activities. As in the current system this feature has been implemented for most activities however for not all. EditCard does not have such navigation because it resets the activity stack so CardActivity appears on the top after saving the card. To implement the system fully more research will be required to understand the problem and try to solve this functionality for existing activities in the system. It should not be very challenging task as currently most activities in the system host such functionality.

- Allowing user to take picture(s) of loyalty cards storing. This will be challenging from storing pictures point as will require the pictures to be stored and viewed only by the application. Such capability will disallow other android applications from accessing them. Possible implementations can be: storing the card images in the database as it is not accessible by

other applications on the device; storing images in android internal storage and hiding from other applications. The former and later options will require more research into correct implementations. Former option may require compressing the image and/or using an object based database than rather current implementation of SQLite. And later would require setting the application resources only accessible from the application.

- Allow user to save favourite cards which could appear on the top of the list. This feature could have been added to the system however due to short amount of time it was not possible to implement. This would not be very challenging task as would require some modification and additional database methods. These methods would allow saving and returning cards marked as favourites. List view could show the favourite cards first and then append the rest of the cards to the list view.

- Two factor authentication. This feature may be among the most challenging features. It will require to send the user a Short Message Service (SMS) message. The two steps can be SMS message and pin entry, or email message and pin entry. The former and later approach will require a server to send messages via SMS or email. This will add costs of running and maintaining the service. Implementation will require additional research and adaptation of new skills. End user may have added costs for such a service inform of a subscription. Realistically the user can have additional device locks such as pin, pattern or password which are offered on the device itself. This would require the user to first unlock the device and then unlock the application through the pin entry system that exists. It is likely

that this is the case during the use of current system. Two factor authentication will require more effort and time to use and hence may deter the user from using the system.

- Improve Graphical User Interface to allow user to change theme: colour scheme of application to their desire. This implementation will require the application to allow change the colours of the application dynamically and adapt to user choice. Will require adding a settings or theme activity. If such a feature is implemented then pin reset option can also be moved under settings to be accessible from within the application. This feature will not be related to loyalty card nor the security of it, but user experience and adaptation to the application.

- Optimise application for use on tablet and larger screens. This feature will be relatively simple to implement as will only require additional layouts for different screen sizes for each activity. It will be a time consuming development but relatively simple. Current GUI designs can be adapted to larger screens. This feature will not pose any security threats within the application as only graphical user interface will be adapted. Although user ability remains questionable with such feature: would the user carry and use a tablet to scan loyalty card at the retailer in store? The part aim of the project was to provide a solution to carrying extra weight and use existing device, phone, to solve the problem. Although it seems vague to carry tablet when one would go shopping, however market research will be required as to if the users genuinely require such feature.

- Allowing user to change or reset the password. Currently, the password of the system cannot be changed. User can be sent an email or SMS message with link to reset the password. Alternatively user could be asked security questions to which answers can be used in conjunction with the correct pin to change the password. Later approach may seem more viable option as can still maintain the data local and no server would be required as in the former approach.

Other improvements can be developing system interface classes and use them in the application. A loyalty card API can be a major extension to this project. Such of an API would allow storing loyalty cards and produce barcode for them. This can allow other developers to use it in their loyalty card applications.

9. Definitions

As used in the report:

1. “PIN” stands for personal identification number.

2. “App” common abbreviation for ‘application.’ Means the software program in compiled format.

3. “API” abbreviation for ‘Application Programming Interface.’ Are a set of functions, procedures, and tools that allow the creation of software programs or applications.

4. “GUI” abbreviation for Graphical User Interface. It is the front end of software program, with which user interacts with: makes choices and inputs or gains output and feedback from.

5. “Activity” in android is similar to a new screen or page. Consists of a different content and GUI than the preceding activity.

6. “Actionbar” quote: “is a window feature that identifies the user location, and provides user actions and navigation modes. Using the action bar offers your users a familiar interface across applications that the system gracefully adapts for different screen configurations.”[8]

[8] Android developer: http://developer.android.com/guide/topics/ui/actionbar.html

10. References

1. Ofcom. [Online] 29 April 2014. http://stakeholders.ofcom.org.uk/market-data-research/other/research-publications/adults/adults-media-lit-14/.

2. YouGov. [Online] 2014. https://yougov.co.uk/profiler#/Android/.

3. icrossing. [Online] January 2014. http://connect.icrossing.co.uk/infographic-android-pulls-in-twice-as-many-users-as-apples-ios_11372.

4. **Google.** Android Developer API guides. [Online] http://developer.android.com/guide/topics/data/data-storage.html.

5. Nectar. *Nectar.* [Online] http://www.nectar.com/spend.points.

6. Wikipedia. [Online] http://en.wikipedia.org/wiki/Barcode.

7. Barcodes Inc. [Online] http://www.barcodesinc.com/info/buying-guides/barcode-scanners.htm.

8. **Ranger, Steve.** *ZDNet.* [Online] 16 January 2015. http://www.zdnet.com/article/ios-versus-android-apple-app-store-versus-google-play-here-comes-the-next-battle-in-the-app-wars/.

9. telegraph. *telegraph.* [Online] 28 April 2011. http://www.telegraph.co.uk/technology/sony/8478404/PlayStation-hack-Sony-users-urged-to-change-passwords.html.

10. Zxing. *github.* [Online] 2015. https://github.com/zxing/zxing/.

11. **Roach, Nick.** flat icons. *elegant themes.* [Online] 16 October 2014. http://www.elegantthemes.com/blog/freebie-of-the-week/beautiful-flat-icons-for-free.

12. Developers. *Android.Developer.* [Online] Google. http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle.

13. howtodoinjava. *howtodoinjava.* [Online] http://howtodoinjava.com/2013/07/22/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/.

14. Android Developer. *Developer.* [Online] Google. http://developer.android.com/reference/android/content/Intent.html.

15. Stackoverflow. *Stackoverflow.* [Online] 2014. http://stackoverflow.com/questions/19451395/how-to-hide-the-soft-keyboard-in-android-after-doing-something-outside-of-editte?rq=1.

16. Developers. [Online] Google. https://developer.android.com/training/articles/memory.html.

17. Google Drive. *Google.* [Online] https://www.google.co.uk/drive/.

18. Dropbox Cloud storage. *Dropbox.* [Online] https://www.dropbox.com/.

19. Nova Launcher Prime. *Google Play* . [Online] TeslaCoilSoftware, 2015. https://play.google.com/store/apps/details?id=com.teslacoilsw.launcher.prime&hl=en_GB.

20. **dm77.** barcodescanner. *GitHub.* [Online] https://github.com/dm77/barcodescanner.