Charity Shop Inventory
Final Project Report

TU856
BSc in Computer Science

Alex Bang

C19409486

Damian Bourke

School of Computer Science

Technological University, Dublin

05/07/23

Abstract

The initial goal of the project is to assist a charity shop in managing its inventory. This will help it be more efficient and save time. It is influenced by my experience working in a charity shop.

The software project will aim to be a toolbox for the charity shop to sort out some of these problems. Before the software project can be developed there need to be some answers to some questions which will be explored in the incoming sections(chapters).

There will be a lot of places to improvise and create new ideas.

e.g. Tracking staff training level.

e.g. Point of Sales

e.g. Waste Management

e.g. Inventory Price Calculation

e.g. Machine Learning supported action

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Alex Bang

Alex Bang

05/07/23

Acknowledgements

Damian Bourke,

Dr Susan McKeever,

Table of Contents

1. Introduction

1.1. Project Background

I used to work in a charity shop during my time in transition year and in my spare time.

I worked in SVP during my summertime for a couple of months. This specific shop was made up of young volunteers and a manager.

Volunteers take turns to manage the cash register with the other volunteers. One volunteer would have to manage the cash register, and another would have to manage the floor. The volunteer who managed the floor had to merchandise the goods, collect the donations, and also try to upsell the patrons.

All volunteers used a sign-in sheet to record in time, break time and out time

The goods were tagged with new tags and put out into specific storage units or out into a stand/ holder on the floor. The donation was put into a pile until the manager inspected the item. With upselling, the volunteer would have to pick one item which is related to the item that the patron is buying e.g., a necklace and ring match together.

The manager did most of the calculations of the price of an item. Factors included brand, size and age of the item. Since the manager knew some of the patrons, they would be able to give discounts or bargains. They would categorise items using pre-existing labels that they had printed.

Some problems would be noticed:

There was some waste produced that had to be thrown away. For example, there was an old toy house was looked obsolete, therefore could not be sold, and had to be thrown away. There was a significant number of things thrown away, therefore costing black bin fees.

Sometimes a transaction was cancelled or other problems such as an item being double counted. Since the cash register was an old mechanical machine, only the manager knew how to fully operate it. The volunteers would have to call the manager out to the till if they needed help with the cash register. The transactions are important in keeping track of the shop's account.

Some of the staff lacked training on site as they did not know what to do in certain situations. For example, one of the staff undersold an item which was not authorised by the manager. There were a lot of situations where the staff need to call out the manager. Another example is when a donor was trying to donate a large table to the shop and the volunteers did not know what to do.

A charity shop would not have the resources to invest in new technology nor have the resources to invest in staff training. It is focused on community values, unlike a commercial setting.

A computer would replace the need for the staff to consistently ask the manager for help. It would also perform better actions e.g., calculating a price of an item or managing the schedule of a volunteer. The computer could also track the performance of a volunteer.

The term “Charity inventory management solution” was searched on Google Scholar but did not mention any solution.

There are books and articles about how to better manage an inventory e.g. (Muller, 2019), (Schreibfeder, no date) and others. There are inventory management systems on the internet.

No project is specially crafted for charity shops yet.

1.2. Project Description

This project is designed to automate some of the manual processes of a charity shop. The student will visit a charity shop or draw from their experience. From this experience, the student will be able to see some problems.

From these problems, the student will design an application using various skills to improve the interaction between the charity and the problem.

An inventory management application will be built to handle those inefficiencies. It will automate some of the tasks, therefore, removing some of the possible human errors. It will be tailored towards the charity shop which is not done by retail inventory application. The application will be used by the volunteers of the shop and the manager. It will also be tested by them.

This application will consist of a front end connected to a database service provider e.g., Firebase. It will have a screen for inputting information, a screen for editing information and other screens.

1.3. Project Aims and Objectives

Aim:

Assist a charity shop with improving the efficiency of their day-to-day application. Create a full functionally android app which will manage the inventory of a charity shop that can be used by charity volunteers. Ensure that the android application will be open to modification in the future.

- Plan a project in order to create a work schedule needed to complete the project. This should be done by the first month of semester 1.

- Research alternative solutions to replace the manual work of stock management done by the manager by using primary and secondary sources about charity, the aim is to produce user requirements that can help with the design of an application. This should be done by Christmas.

- Design a solution which will compete with alternative solutions of a stock management system. The aims are to build a system design which will meet the user requirements identified in the first milestone. This will include an architecture diagram, class diagram and ERD The solution design will then be used to produce a prototype stock management system. This should be done by Christmas.

- Code a working system which will take into account the user requirements, successes and failures of the prototype and improves on these. The aim is to create a working system which should be an android application consisting of a front end and back end.
  The working system will be using a solution to tackle the manual work of stock management done by the manager. This should be done by the end of March.

- Develop a testing and evaluation regime which would also involve an end user. The testing will aim to check that the working system is working based on a set of test cases as intended and the evaluation checks if the working system is meet the user requirements via a checklist that we are discussing later. This should be done by the end of March.

1.4. Project Scope

This project will be about how to best manage the inventory of a charity using a phone. The project will automate some of the manual work done by a charity shop.

The project will aim to search for different solutions to the problems that are not addressed by private entrepreneurs. The project is not about making a charity shop 100 % efficient. This project will not produce a profitable application. This is because the application will not try to answer profitable questions. It will not be a performance application either. It cannot track the performance of the volunteers using the application. The project will not be a point of sales application. The system will not be designed with a payment system in mind. The project will not be a charity profile application. The application may not even be viewed by a customer.

1.5. Thesis Roadmap

Chapter 2 -The Literature Review is about researching existing solutions and technologies to complete the project. The user requirements will be revised in this stage.

Chapter 3 - Experiment Design is about expanding on the blueprints in terms of methodology and system. Artefacts will be used in this stage to create designs.

Chapter 4 -Experiment Development is creating new a piece of work used to demonstrate the project can work. This will check the feasibility in continuing the project.

Chapter 5 -Testing and Evaluation is the plan to ensure that the software project will run as expected. The software performance will be checked against user requirements.

Chapter 6 -Conclusion and Future Work is an outline of the next part of the process which is developing the application. This is the last stage.

2. Literature Review

2.1. Introduction

The tools needed to complete this assignment will be explored in this section.

Existing solutions which could have completed this assignment will be researched as well. Common features of this solution will be identified. Nielson's heuristics will be used to generate a rating for each of the systems. It will be further used to identify features that resulted in the rating. Additionality features that could have improved the rating will also be listed.

The main purpose of this section is that a set of user requirements will be generated.

This section will also include programming languages, frameworks, and service providers. This is to pick the most suitable language for the job i.e., which advantages are better.

2.2. Alternative Existing Solutions

This solution will be visited using Nielson's 10 heuristics. Nielson's 10 heuristics are a set of guidelines which should be followed which help develop an application's UI. User requirements will importantly be developed at this stage too from the features discussed. These guidelines are.

1. Visibility of System status,

The systems need to be provided feedback to the user.

2. Match between the system and the real world,

The system needs to follow the logic of the real world.

3. User control and freedom.

The system needs to be open for the user to explore what they need it to do.

4. Consistency and standards

The words in the systems need to have the same meaning through the systems.

5. Error Preventions

There should be "guardrails" to stop the user from making errors.

6. Recognition rather than recall

The user should not have to use a lot of brain power to remember things in the system.

7. Flexibility and efficiency of use

The system should be straightforward and allow for some discrepancies.

8. Aesthetic and minimalistic design

Only content that is needed should be added to the page and the design should only be for what is needed.

9. Help users recognise, diagnose, and recover from errors.

The user should be warned directly should they make a mistake and there should be suggestions on how to fix the problem when possible.

10. Help and documentation.

As a last resort the user should be able to find a guide or contact someone should they continue to be stuck with their problem.

Systems such as Stock and Inventory Simple (SIS), Square Point of Sale (POS) and Zoho. Excel Spreadsheets, Microsoft Access, and Other Databases were investigated. By using Nielson's heuristics, ratings out of tens for each system were generated. Features that resulted in these ratings were identified. To avoid being biased, a sample of charity shop members were invited to fill in an online survey. Since different people have different opinions, it was a good idea to ask multiple people. An average of the collected opinions was then generated from the charity shop personnel. From this survey ratings for each of the systems were also generated. Each heuristic has a value of 10. 10 meaning that it follows those guidelines well and 1 meaning it does not. Since there are 10 heuristics, the overall rating should be 100.

From the heuristics rating, it was decided a further investigation was to be performed i.e., brainstorming features that could have improved the heuristic rating. (This is in red writing on the chart) These features helped generate the user requirements for the new system that will be designed in the next section.

Stock and Inventory Simple (SIS)

SIS was the most influential solution to the project. However, some of the features that were flagged were the following: SIS lack some key elements such as being able to store things online. It lacked a couple of guardrails to guide beginners through the system e.g. There was a lack of error messages if a mistake was made. SIS did not have any features that helped beginners use their application. They assumed that the user was already good at using mobile applications as well as mobile devices. Hence there was a lack of help and documentation.

This system was therefore given an overall heuristic rating of 63 out of 100.

	Heuristics	1. Visibility of system status	2. Match between system and the real world	3. User control and freedom	4. Consistency and standards	5. Error prevention	6. Recognition rather than recall	7. Flexibility and efficiency of use	8. Aesthetic and minimalist design	9. Help users recognize, diagnose, and recover from errors	10. Help and documentation
Stock And Invenotory		5/10	8/10	7/10	8/10	3/10	6/10	7/10	7/10	7/10	5/10

		1. Visibility of system status	2. Match between system and the real world	3. User control and freedom	4. Consistency and standards	5. Error prevention	6. Recognition rather than recall	7. Flexibility and efficiency of use	8. Aesthetic and minimalist design	9. Help users recognize, diagnose, and recover from errors	10. Help and documentation
App1 : Stock And Invenotory		Navigation Branch Name at top of page What's New tab Reports Icon in mid of page Ask before linking Loading Screen	Invoice Logically Flow Icons Small calculator screen	Short Flow Back button Home button Redo button Free	ListViews Android template -------------- Words	Snackbar:Attention! With Ok button Prominent button Blocking page until correct input Autocorrect mistakes	Mostly Inputs Saved data Summary page	Easy to use design Barcode auto generater Sorting customers Turn into Excel Add google drive Copy and paste	Small number of inputs High signal to noise no advertisements	Plain language Popup Snackerbar: Attention! History Navigate to problem Use colours to hightlight danger	Knowledgebase Helpful pop ups Question or suggestion Quick help context pop Hover Help with context involved

Zoho

Zoho was another very influential solution to the project. Some of the features that were complained about were: that it lacks help and documentation within the application. Some ideas that were suggested in improving Zoho is it should improve its “visibility of system status”. There are some messages such as “No Invoices created so far” as a way of indicating that there are no invoices immediately after scrolling into the page. The user will not have to scratch their heads to figure out what happens. However, it does not have enough feedback to guide novice non tech-savvy users. It cannot modify the systems to be tailored for just charity shops.

This system was therefore given an overall heuristic rating of 65 out of 100.

	Heuristics	1. Visibility of system status	2. Match between system and the real world	3. User control and freedom	4. Consistency and standards	5. Error prevention	6. Recognition rather than recall	7. Flexibility and efficiency of use	8. Aesthetic and minimalist design	9. Help users recognize, diagnose, and recover from errors	10. Help and documentation
Zoho		5/10	8/10	8/10	7/10	7/10	8/10	5/10	9/10	6/10	2/10

		1. Visibility of system status	2. Match between system and the real world	3. User control and freedom	4. Consistency and standards	5. Error prevention	6. Recognition rather than recall	7. Flexibility and efficiency of use	8. Aesthetic and minimalist design	9. Help users recognize, diagnose, and recover from errors	10. Help and documentation
App2 : Zoho		No Invoices created so far “Confirmed” label Dashboard	Top to bottom terminology	clearly marked back button	ListViews Android template	Block pass to the next page if you have not filled in an	summary page a saveable file	Easy to use design	excellent signal to noise ratio	popup which alerts the user History	Anything

Square Point of Sale (POS)

POS was a complex system which did more than the inventory management systems needed. There was a couple of reason why this system was not suitable. Its design was too complex, there was a lack of “User control and freedom” which would make it hard for non-tech-savvy volunteers to use. For example, there was a lack of an escape button. This system was also tailored towards business-savvy users. This shows a lack of consistency and standards. Words such as “SKU” – Stock-keeping unit is not a word everyday people use. It would be more useful to use the word “stock amount”.

There was no heuristic rating done on this system. This system acted as an extra system to identify features that would increase the heuristic rating.

		1. Visibility of system status	2. Match between system and the real world	3. User control and freedom	4. Consistency and standards	5. Error prevention	6. Recognition rather than recall	7. Flexibility and efficiency of use	8. Aesthetic and minimalist design	9. Help users recognize, diagnose, and recover from errors	10. Help and documentation
App 3: Square			Big Calculator screen		common colour		icons	Addons Important feature on startup			Quickstart tutorial Website with pictures with context
My App											

Below is a summary chart of heuristics ratings.

	Heuristics	1. Visibility of system status	2. Match between system and the real world	3. User control and freedom	4. Consistency and standards	5. Error prevention	6. Recognition rather than recall	7. Flexibility and efficiency of use	8. Aesthetic and minimalist design	9. Help users recognize, diagnose, and recover from errors	10. Help and documentation
Stock And Invenotory		5/10	8/10	7/10	8/10	3/10	6/10	7/10	7/10	7/10	5/10
Zoho		**5/10**	**8/10**	**8/10**	**7/10**	**7/10**	**8/10**	**5/10**	**9/10**	**6/10**	**2/10**

2.3. Technologies Researched

In this section, various technologies will be discussed that could be used to implement this project. The advantages of the technologies will be mentioned. The technology will be explored in terms of front-end technologies and possible back-end technologies.

Front-end technologies are used to create screens that directly interact with the user. For example, programming languages such as ReactJS Native, Java (Android Studio) and Flutter. Platforms include web and mobile applications.

Front end

The front-end interface is part of the system that will face the user. There are two main platform choices discussed here. Mobile applications and web applications. Programming languages that are used to code front-end applications will be discussed after.

Mobile application

There were many design considerations such as what platform to use. Mobile applications have features such as “mobile apps offer better personalization”, “Ease of sending notifications”, “Making use of smartphone’s utilities” and “Ability to work offline” as spoken about by (Magenest, 2021).

Some mobile applications are to be tailored to certain personalities which is where better personalization comes in. By downloading the application, it is possible to roughly identify which devices, that will be targeted.

Mobile applications should be easy to send notifications in. This is because they create have the ability to send messages easily to other mobile devices.

Mobile applications contain utilities. This includes using the “camera”, and “contact list” as identified by (Magenest, 2021) and other smartphone utilities such as the barcode scanner.

The “Ability to work offline” spoken by (Magenest, 2021) is another important component of the application. As the mobile application can still work when there is no internet around.

Web Application

With a web application, most devices can access the web. Therefore, there is a need for generic features that would fit all. This is the “compatibility” that is offered by the web application as listed by (Magenest, 2021). This means that a single mobile website can be accessed by multiple different types of devices.

Mobile websites can be discovered easily. This is because search engines can show websites in “list items” and registered in “industry-explicit registries”, which can be used to target a group of users as according to (Magenest, 2021).

The “shareability” offered by web applications is as mentioned (Magenest, 2021). This means that the website URL which contains the location of the website can send to different people fast, therefore, making it easier.

Websites have the power to be upgraded fast. Mobile websites can be updated immediately due to the ability to “publish” the modification once as highlighted by (Magenest, 2021).

Java

Java is a “general-purpose, robust, secure and object-oriented programming language” according to (Javatpoint, no date). Currently maintained by Oracle. It is simple to learn about and initiative. The syntax is borrowed from the C++ language. One awesome feature of java is the automatic garbage collection helps to clean out objects that are not used from the RAM. Java did not inherit explicit pointers nor operator loading and other features from C++ so it makes it beginner friendly.

(Javatpoint, no date) says that objects are a key part of Java. It is very easy to spot “objects, classes, inheritance, encapsulation and polymorphism” in java according to (Javatpoint, no date).

Java’s security is impressive as the explicit pointers are not used as mentioned above. Java also has a special environment in which to run its program which is a “virtual machine sandbox” mentioned by (Javatpoint, no date). JRE’s class loader is said by (Javatpoint, no date) to be able to insert a class into the JVM while the code is running. There is a divider between the “class packages of the local file system” and the packages imported by the user according to (Javatpoint, no date).

Flutter

Flutter is a software development kit which creates apps with fast developability and beautiful widgets. One of the most impressive features is the hot reload functionality that allows the developer and designer to see the changes in the code quickly on the application. This allows the user to get immediate feedback for their work according to (Beladiya, no date). Flutter is also said to have high performance according to (Beladiya, no date). The CPU usage, frames per second, and average response time are said to be excellent.

With regards to the same hot reload feature (Beladiya, no date) it allows for “immediate updates” to the code this allows an update can be changed in the current moment and fixes can be made immediately “without having to restart the app”. Hot reload circumvents long delays with quick changes. (Beladiya, no date) mentions that there is boilerplate code which can be used to further increase the speed of code production. The boilerplate widgets are customizable and are of high standards made by experts. (Beladiya, no date) believes that the process of learning flutter is easier. Lessor-experienced developers should have little difficulty in creating an application with Flutter.

ReactJS Native

ReactJS is a JavaScript library that incorporates speed and a new innovative way to render pages. According to (Vadalia, 2021) this results in a highly changeable and responsive application. ReactJS use a virtual DOM instead of the real document object model. This virtual DOM allows for enhanced speed and quicker fix viewing according to (Vadalia, 2021). It is possible to reuse React components which result in time savings. ReactJS components are “isolated” which means components are independent of changes in other components according to (Vadalia, 2021).

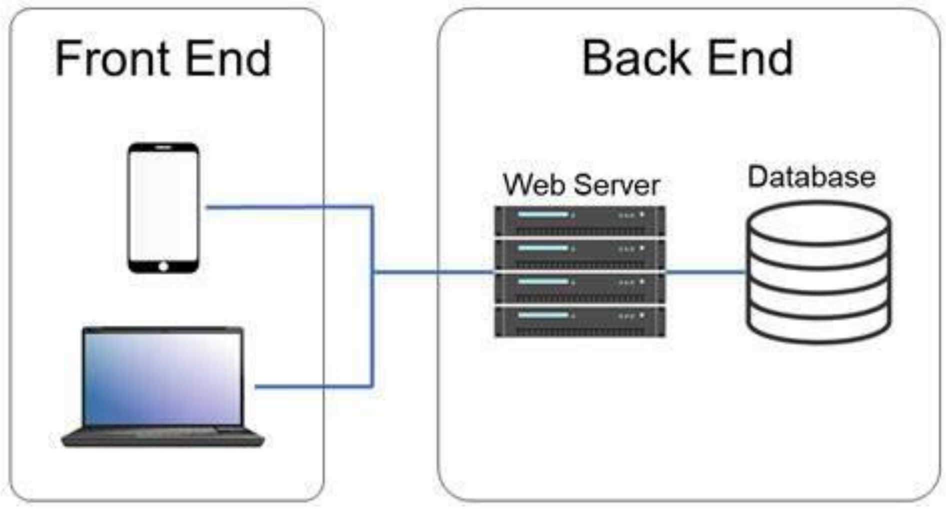
(Vadalia, 2021) mentions that there is a property in ReactJS which is data backlinking. In this context, ReactJS components can be directly manipulated, however, changes in child structure data do not affect the parent structure data.

ReactJS also has a vibrant community due to it being an open-source Facebook library. This allows anyone with approval to make modifications to the source code. There are a lot of “useful apps” and “additional tools” as described by (Vadalia, 2021).

From the discussion of the front-end platforms, mobile applications were chosen over web applications due to their advantages e.g. ability to work offline. Java was chosen as the programming language of choice due to its advantages e.g. Object focused.

Backend

This is part of the system that contains the most inner works of a system. The below tier diagram illustrates the relationships between the two tiers.



SQLLITE

SQLite is a widely used database which is constantly used in “disk file format” for desktop applications e.g. “version control systems” and “financial analysis tools” as mentioned by (TheDeveloperBlog, no date). It also mentions that SQLite is a “lightweight database” so it is typically used in television, mobile phones, etc. This means that is smaller in file size space taken and features. (TheDeveloperBlog, no date) states that SQLite’s performance is superb. Its “reading and writing operations” are very fast. Additionality it is “35% faster than the file systems” which is impressive.

(TheDeveloperBlog, no date) mentions that “SQLite is very easy to learn”. SQLite only requires libraries to download and is ready to go. That means that is fast to set up.

(TheDeveloperBlog, no date) states that SQLite is reliable. It automatically saved your files, and in the event of a power failure or crash the damage is minimal. SQLite has lower risks of bugs. One of the reasons is that the queries it uses are smaller therefore the risks of bugs are also smaller.

(TheDeveloperBlog, no date) states that is very accessible using many different “third-party tools”. Should the data be lost, it should still be possible to recover this data.

(TheDeveloperBlog, no date) mentions that SQLite also has “reduced cost and complexity” due to the same smaller queries as listed above. It is also possible to extend the functionality of SQLite due to its adhering to extension principles.

Firebase

Firebase is a backend-as-a-service (BaaS) platform. Google owns this service. Developers are provided with a wide range of tools e.g., real-time databases in order to develop interactive applications. This is a service that allows faster development according to (Back4app, 2021).

(Back4app, 2021) states that firebase has a couple of types of innovation databases such as the Realtime Database and Cloud Firestone. Some advantageous features of the Realtime Database include the database working without internet connectivity, adding data permissions to the real-time database and synchronization after internet re-connectivity.

(Back4app, 2021) mentions that firebase has “fast and safe hosting”. Firebase uses zero-configuration SSL which adds a layer of security to the content being sent/ received. SSL certifications are also able from Firebase. Firebase hosting allows many web content types. E.g. web applications, and dynamic and static content according to (Back4app, 2021).

(Back4app, 2021) mentions that firebase gives some leeway to a beginner who intends to experiment with the service. This means that they do not need to “invest a single penny” when they start creating their application. The user is given a chance to test their application in a real-life simulation. According to (Back4app, 2021) are many free testing features which are connected to firebase e.g. previewing the project. However, once the usage of these services or database memory consumption passes a certain point, it costs money.

(Back4app, 2021) states that google analytics has integration with firebase. This gives the developer oversight of the usage of their application.

(Back4app, 2021) also states that firebase provides “cloud messaging for cross-platform”. This increases the accessibility of the device platform.

From the discussion about the back end, Firebase was chosen over SQLite due to its advantages e.g., innovative databases.

2.4. User Requirements

This is the main purpose of this section to generate user requirements. User requirements will be needed to shape the artefacts in the chapter3. They are carefully crafted by reviewing all the data gathered from Chapter 2.2 (Existing Systems).

System requirements will then be created to handle these user requirements from the system perspective. These system requirements will be directly incorporated into the system design.

Here below a user requirement will be mentioned and a corresponding system requirement will be discussed on how to handle the user requirement.

1. The user can store inventory using this application, they can create, read, update and delete items in the application. Images for items will be available,
   a)-> The systems need to allow CRUD and images for charity shop items.

2. The user can be assured that the application will follow Nielson's heuristics e.g. System Status Indicators like loading pages included, and error prevention like confirmation buttons included. The UI will be guaranteed to be good.
   a)-> The system will have Nielson's heuristic in its design.

3. The user will be able to search for items as well as sort them. This will allow them to find specific items more quickly.
   a)-> The system will have the ability to use sorting or filtering on its data using special algorithms.

4. Since the charity shop inventory is online, there is a possibility for extra backups to protect user data from damage.
   a.)-> The system must be connected to a database, the database will have the option to create extra backups or the database provider could provide it by default.

5. The application will allow the manager to approve volunteers and approve items, giving extra oversight over the charity shop.
   a.)->The system needs to keep track of volunteers and managers in some form.

6. The application will provide some information about the charity shop to the volunteer e.g. Floor procedures.
   a.)->The systems will need to store embedded data about the charity shop via the shop's online documentation.

7. The application will be able to notify either the manager or other volunteers about changes to the shop via messaging.
   a.)->The systems will use any form of message in order to communicate with users' phones.

2.5. Conclusions

Existing solutions were discovered not to be able to satisfy the problems that Charities have. Due to these new systems is to be designed and developed. The design of the system will need to be explored in the next section.

A mobile application was chosen for the front-end due to the advantages e.g. the ability to send notifications easier than a web application. This will affect the complexity of the database in the design section.

Java was chosen for the languages due to its advantages e.g., OOP properties and ease to learn, etc, which makes it easier to code the project. Firebase was chosen for the backend due to the wide variety of tools it provides. This will only affect the classes in the design phase.

User requirements were discussed and written down. Corresponding system requirements were created and recorded. These system requirements will be used throughout chapter 3 to shape the design of the front-end and back-end.

3. Experiment Design

3.1 Introduction

In order to design the software a mental framework i.e., Software Methodology will be used to keep track of the system designing phase and then be used to further track other phases. The main purpose of the section is that a higher-level design with more detail will be designed within this section. The system requirements from chapter 2.2 will have a direct impact on the design of the system. The implementation of the system will be guided by this higher-level design.

3.2. Software Methodology

A software methodology is a mental framework which becomes a habit for the developer. It helps developers to schedule their workload in a way that maximises productivity and reaches their target of developing software.

Feature Driven Development

Feature Driven Development was used as the software methodology to keep the project on schedule. FDD uses a series of engineering rules in order to create a small series of features within a 1-to-2-week cycle. According to (Lucidchart, 2019) this is completely different to other iterative frameworks e.g. scrum and XP. FDD manages to expand a project quicker and onboard new team members quicker too.

According to (Lucidchart, 2019) FDD gives the team gives a better knowledge of the parameters of the project and what the project is trying to achieve.

FDD demands fewer meetings than other frameworks. According to (Lucidchart, 2019), too many meetings is one of the downsides of other iterative frameworks. While using scrum meetings are necessary to brief members. FDD uses documentation to brief users.

(Lucidchart, 2019) mentions that FDD uses a “user-centric approach”. Compared to other iterative frameworks e.g. scrum the product manager is thought about as the end user. Meanwhile back in FDD, the client is the end user.

(Lucidchart, 2019) mentions that FDD is strong when projects are large, long term and continuous. The methodology can be scalable and melded into a framework which develops with the company. Team members/ new team members will be well briefed due to the five steps and guidelines that are described by the company.

Since features are cut into smaller pieces and forged into deliverable releases. (Lucidchart, 2019) states that this is easier to manage and therefore record. As a result, coding errors, risk and quick adaptions can be managed.

Figure 3 is a diagram of FDD.

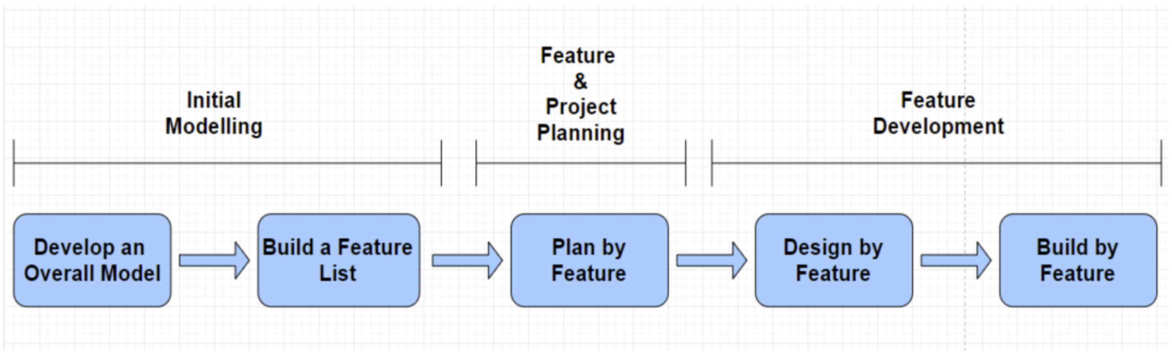
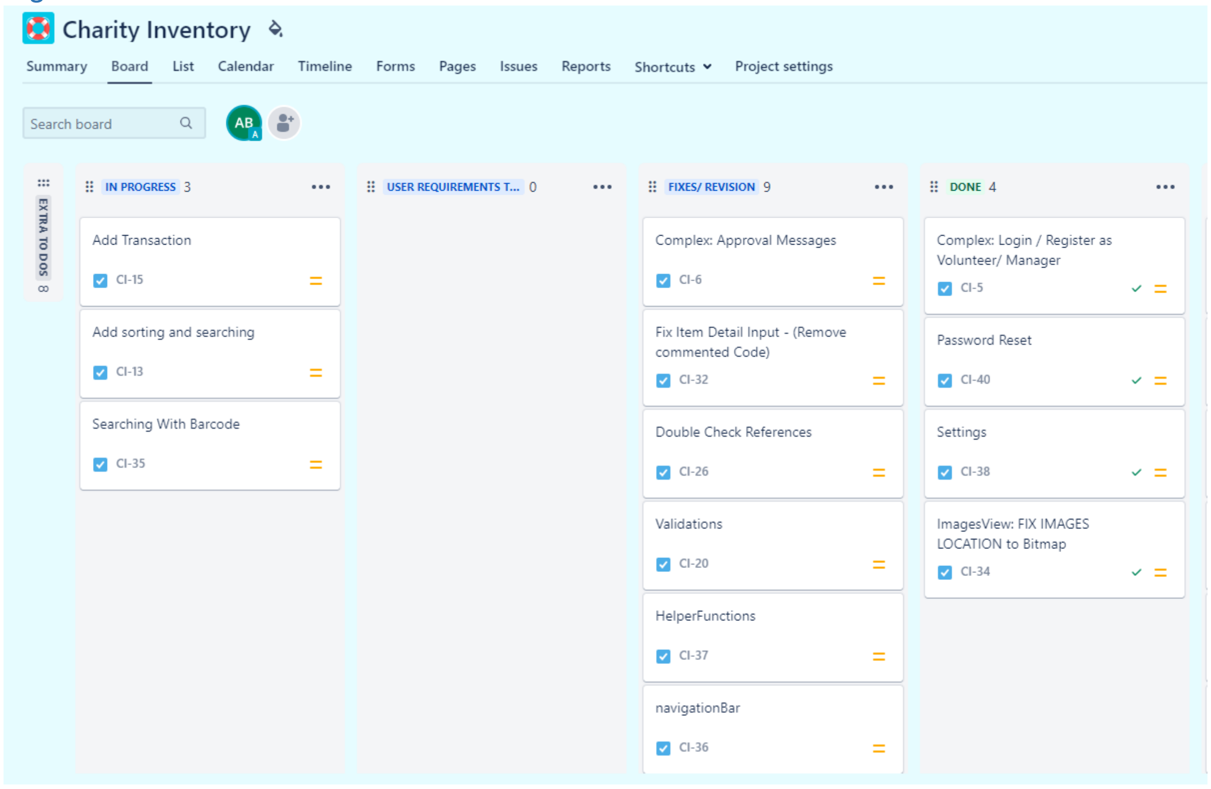


Figure 3 - Feature Driven Development

With FDD in mind with Figure 3.2 shows the use of a kanban board in order to implement FDD. An "Overall Model" will be created in the next section 3.3 Overview of System. The "Feature List" in shown in the "User Requirements" column. Each ticket records a feature and is the main focus of the project. Each ticket acts a "Plan" in order reach a stepping stone in the project within a certain timeframe. In order to complete a ticket a design and corresponding code is needed to satisfy the ticket requirement.

Figure 3.2

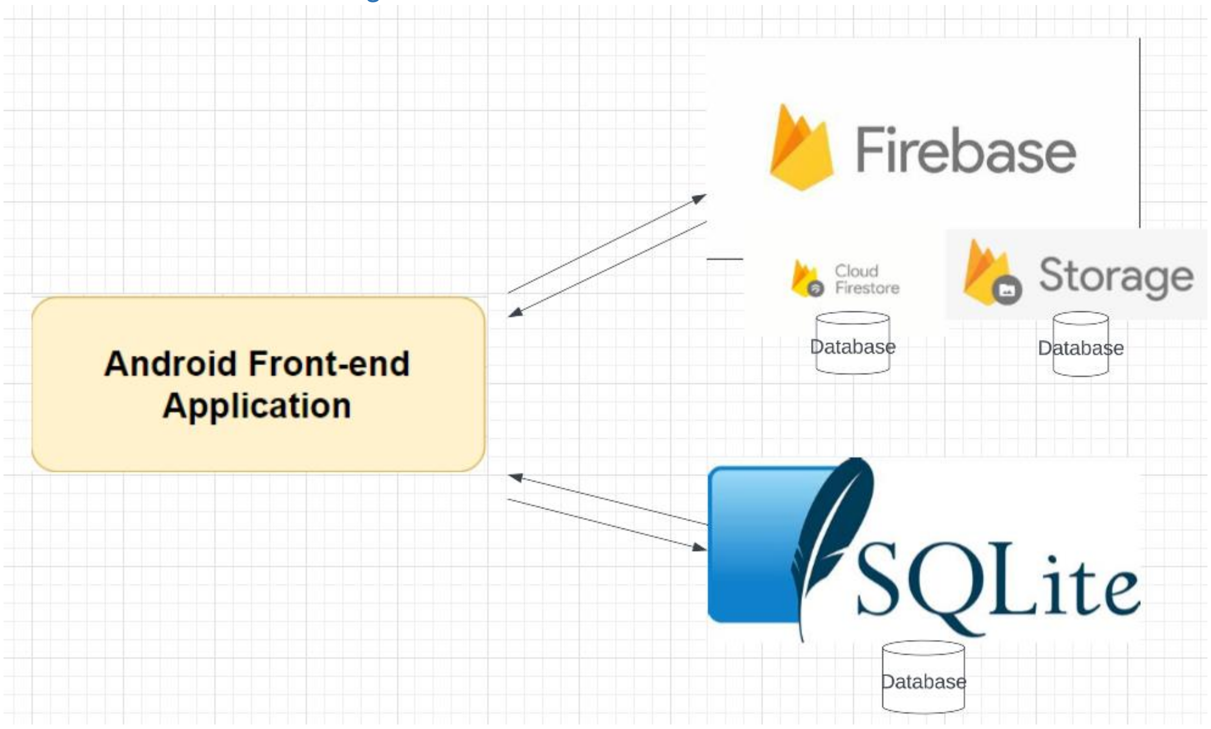


IN PROGRESS 3	USER REQUIREMENTS T... 0	FIXES/ REVISION 9	DONE 4
Add Transaction CI-15		Complex: Approval Messages CI-6	Complex: Login / Register as Volunteer/ Manager CI-5
Add sorting and searching CI-13		Fix Item Detail Input - (Remove commented Code) CI-32	Password Reset CI-40
Searching With Barcode CI-35		Double Check References CI-26	Settings CI-38
		Validations CI-20	ImagesView: FIX IMAGES LOCATION to Bitmap CI-34
		HelperFunctions CI-37	
		navigationBar CI-36	

3.3. Overview of System

The experimented systems consisted of a front end and a back end. The design of the front end and the design of the back end will be explored.

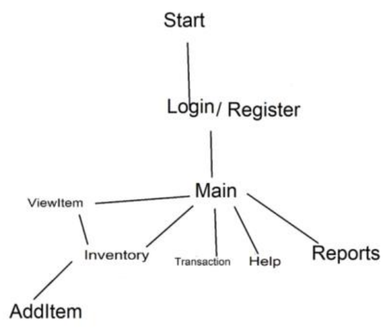
Software Architecture Diagram



This diagram above illustrates the software architecture. Here there is an android front-end application connected to Firebase. The Firebase Realtime Database will need to be accessed to store data about the charity shop online. Additionality the Firebase "Storage" database will be accessed to store the images from the charity shop. The Firebase Cloud Firestore was used to store a copy of the data, this for a feature that will be added in the future. The SQLite Database which was mentioned in the Interim Report was dropped due to the decision to keep the app online.

Front end

Screen Flow Diagram



Screen Flow Diagram
(Fig.4)

All of the screens/ activities are shown above in Fig.4 Screen Flow Diagram. Some of screen’s name changed from the Interim Report as shown in brackets. The main screens are the AddItem (input page), Transaction(invoice page), Main and Inventory page. Input page will be used to input details for new items. There will be a button for increasing the stock level, a button for generating a barcode and an option to select a picture from the gallery or camera. The barcode button will bring the user to a sub screen to scan a barcode.

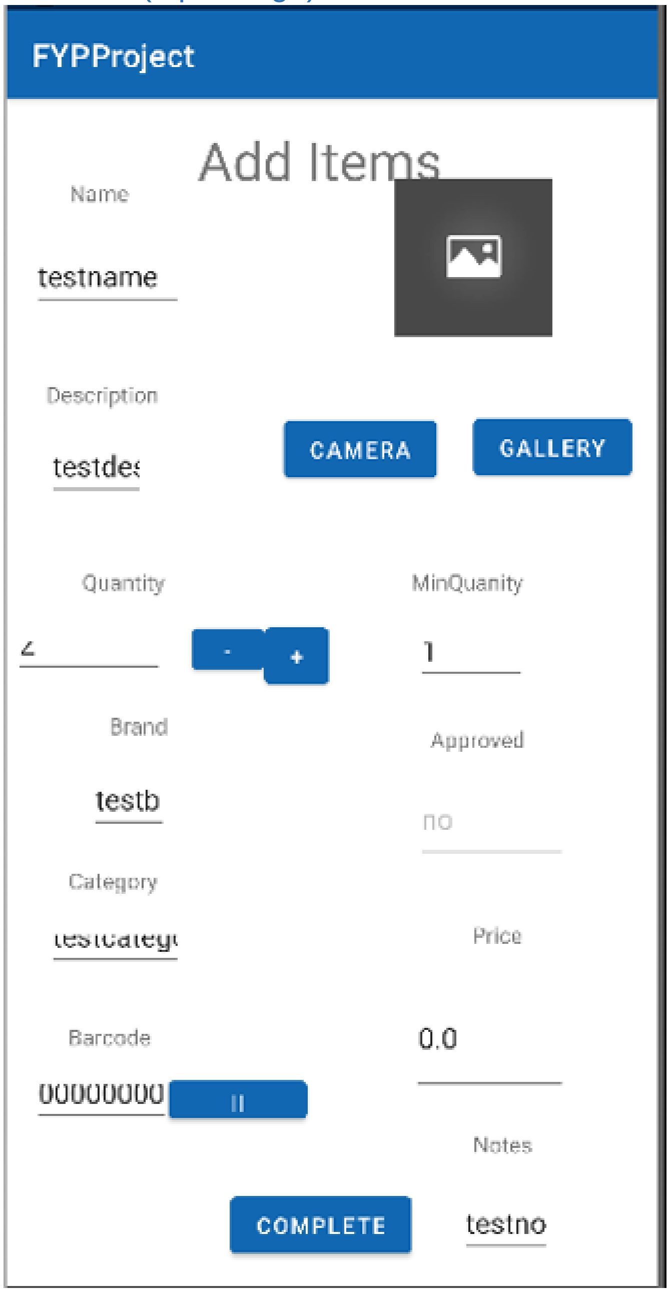
Transaction page will keep track of each transaction, the plan is to have each transaction autogenerated. There is a plan to add a filter button and search functionality too on the invoice page.

Main page will contain various navigation options. This will be main portal to access the other functionalities. This will work in par with the navigation sidebar to ensure that the user have multiple ways to do one thing. After careful review, it was decided that the main page should contain a “Quick Action” button which contains a way to access an item through searching a barcode.

Inventory page will be used to manage all the items e.g., perform Create, Read, Update and Delete operations via buttons. There is a plan to add a filter button and search functionality too. There will be a button for quickly deleting an item. There will be a button to group items, duplicate items and finally add a new item. There will be an automatic item counter below. To further describe the inventory page, it is a passageway to the AddItem page which as mentioned above handles the Create options. The Update functionality is handled by a new page called ViewItem. The Inventory page will handle deletion. The read functionality is common to all of them.

High Fidelity Prototypes

Add Item (Input Page)


FYPProject

Add Items

Name

testname

Description

testdes

Quantity

4

MinQuanity

1

Brand

testb

Approved

no

Category

testcategory

Price

0.0

Barcode

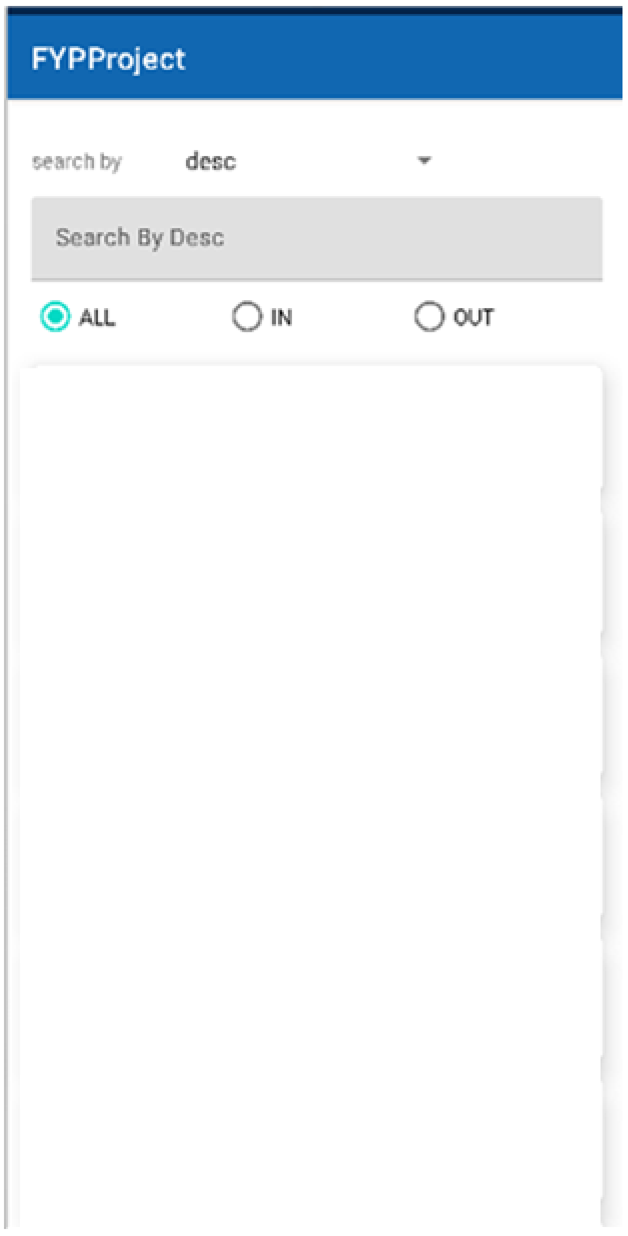
00000000

Notes

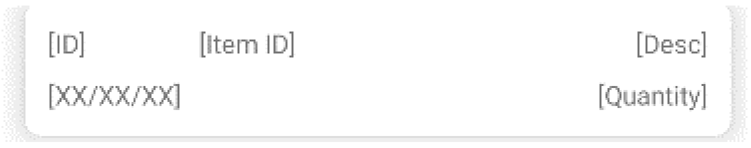
testno

Add Item Page contains all the necessary fields to enter data into a new item. An item will represent an item in the charity shop inventory that can be sold. The Name, Description, Brand and Price are self-explanatory. The field Quantity has two matching buttons which are plus and minus quantity amount. The Category field will be used to join items into a group to help them be located faster. The barcode field is generated by taking read a barcode. The image field can either take an image from the phone gallery or phone camera. The note fields can be used to write down extra notes. The note page is used for storing notes.

Transaction Page (Invoice Page)



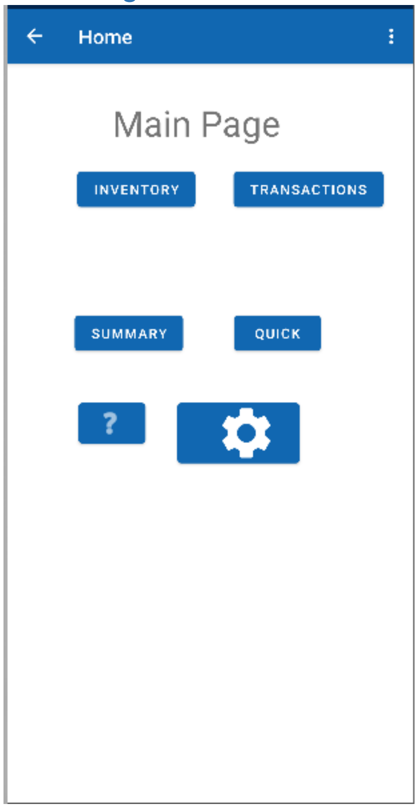
Transaction Row



This page will be used to keep track of the flow of items coming in and out of the charity shop. There is a search bar as shown above. The “search by” shows all the fields that can be filtered by e.g., desc (description/ description of Action). The search box allows the input of “words” that can be filtered for within a field. There are a couple of radio buttons “All”, “In” and “Out” which are all part of the same radio group. These are the most common “word” are shown as radio buttons.

A transaction will have a TID(Transaction Id), Related Item(This is the id of the Item which is tracked by the transaction), Description of Action(what is happening to this item), Date of Transaction and quantity of that item involved. The plan is to have this auto-generated when a volunteer sells an item or trashes an item or receives a new item from a donor.

Main Page



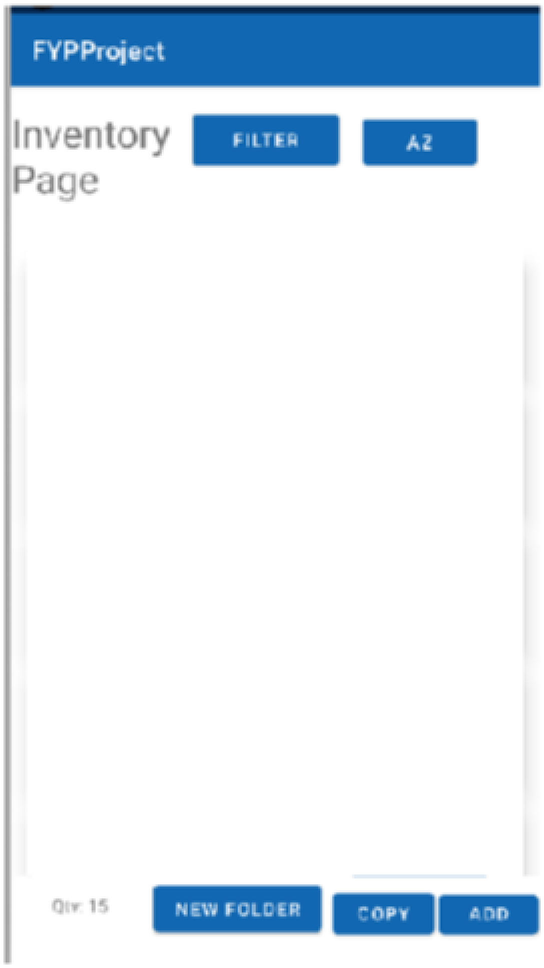
This page will be the main page once a user enters the application after the login/register page. The main page is basically a navigation page. It will also access to a navigation bar as another way to navigate. The "Three dots" icon will contain important destinations in this case it is another way to access the "Settings page.

The buttons are: "Inventory", "Transaction", "Reports" (Called "Summary"), "Help" (Question Icon) and "Settings" (Gear Icon) will also lead to their respective pages. The Inventory Page as mentioned before will allow the user to delete a item record but ultimately it is a passage to Add Item and View Item pages. The Transaction page has been mentioned in detail above. The "Reports" page will give a summary of some of the activities e.g., "Items In".

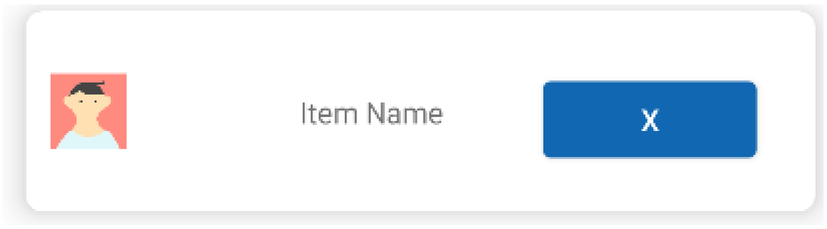
The "Quick" button stands for "Quick action", this will allow the user to scan an item and then view the details of this record.

The "Help" Page will contain several articles within the application on how to solve various problems and tips on how to best use the application. The "Setting" Page will contain a list of settings that the user can modify e.g., Skill level.

Inventory Page



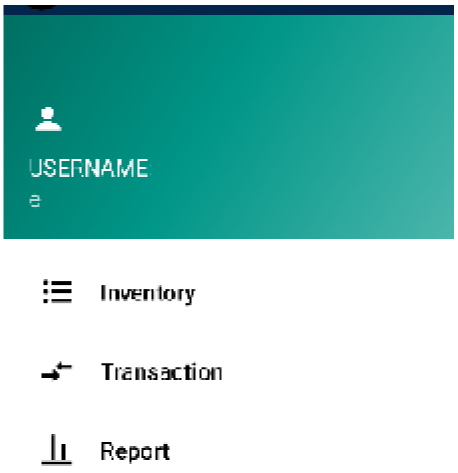
Item Row



The inventory page is the page where the user can view the inventory in the charity shop. They can also seek more information about that item by clicking on the item records. There is also an opportunity for the user to edit and delete items. From the top middle, there is an icon to access a "search bar", this can be used to search for items using any of the item's values. There is also an "order icon" which will order the icon by default from ascending order based on the name of the item. This can be reversed by clicking the "order icon" again. In the future, there are plans to add more ways to order items. The icon records are stored in a list view. In this scenario, there are 15 records e.g., "testname", "ztestname", and etc. Each of the items will come with a record which contains a picture, its name and a delete circular icon marked as an “X”. By clicking on the records, there is more in-depth information about the particular item such as a bigger image and more parameters about it. The parameters are the same as in the “Input Page” e.g., Name, Description, Quantity etc.

Below is the listview of records. There are 3 icons (buttons) i.e. New Folder, Copy and Add New. The “New Folder” icon will generate a folder for the user to order their items in a better way. It will be like the folders in Windows 10. The “Copy” Icon once clicked in conjunction with clicking a record will duplicate a new item with the same parameters as the item that was chosen. However, it will be a different item with a unique id. The “Add New” Icon will be a button that will lead to the “Input Page” as mentioned above in the previous pages.

Sidebar Menu

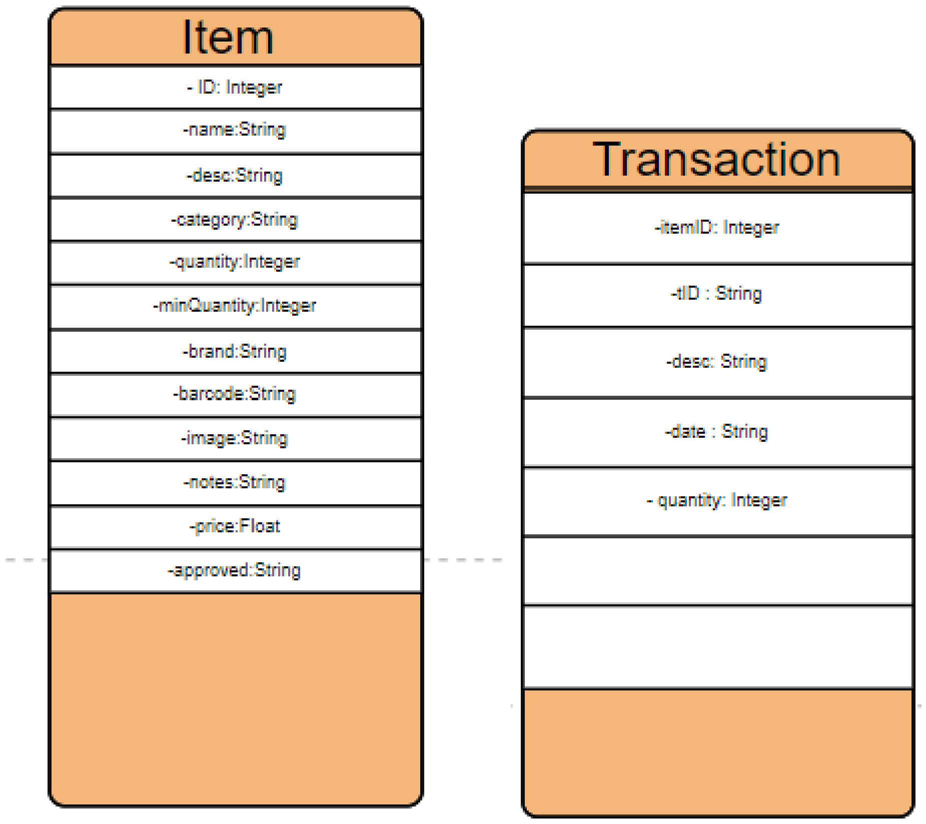


The sidebar menu is basically a navigation bar with buttons which will bring the user to the respective pages e.g., “Inventory” button will bring the user to the “Inventory” and etc. This is just another way to navigate which gives more control to the user.

Back End

The plan is to use Firebase as the database provider. Therefore, the ERD here will be used to create a structure in the Firebase Realtime database to store the data. The Firebase “Storage” functionality could be used too.

Class Diagram



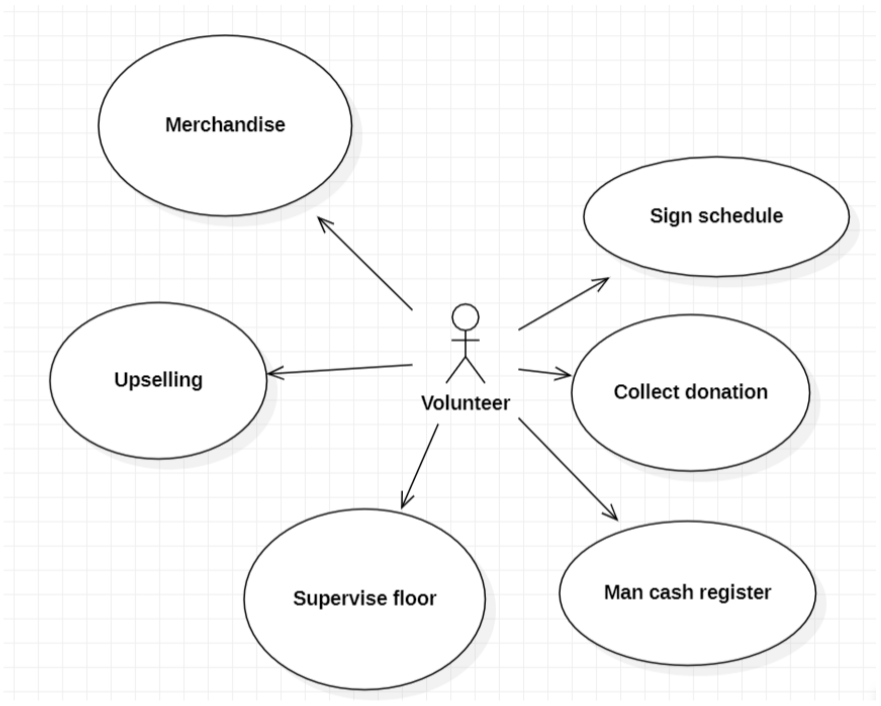
There are two main entities stored: Item and Transaction. These classes will be the most modified class in the application. The other entities: Account and Prevalent used less often.

3.4. Use Case Diagram

Use cases capture some of the system requirements that the system needs to have.

Use Case

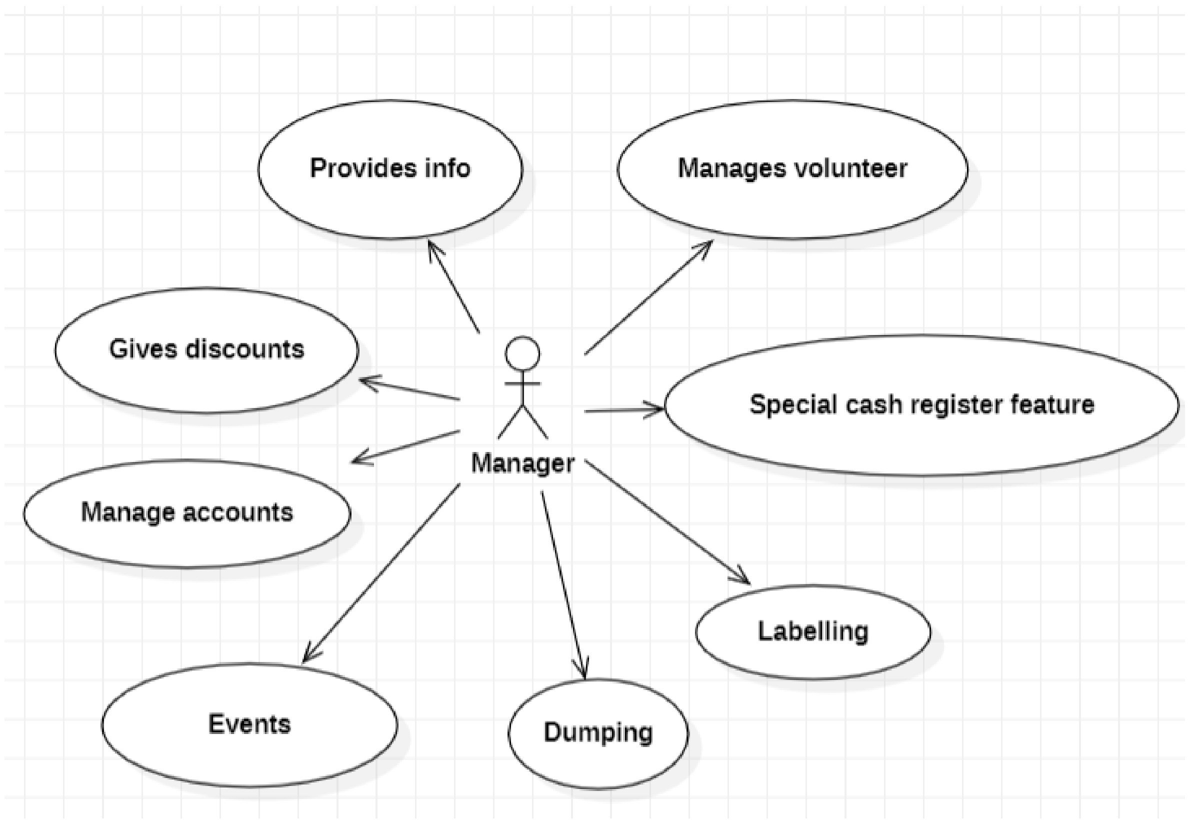
Below are the use cases of the volunteer and manager.



Volunteer

As a volunteer,

1. They need to be able to sign-in to the charity shop.
2. Collecting donations is another duty of a volunteer from donors coming through the door.
3. Volunteers may be asked to man the cash register as well.
4. They must supervise the floor as well.
5. They need to be actively upselling to the patrons of the charity shop.
6. They need to merchandise if an item is not looking at its best.



Manager

As a manager,

1. They need to provide information to the patrons of the system.
2. They give discounts to the patrons.
3. They manage the money accounts of the charity shop.
4. They set up event promotions.
5. They oversee the of dumping unsellable goods.
6. They need to print labels for the categories of goods.
7. They have full access to all the features of the cash register.
8. They manage the volunteers.

3.5. Complexity

Within the project some features of application are more complex than others. This section is to highlight them and prevent confusion. This is a repetition of the system requirements as listed below:

1. The systems need to allow CRUD and images for charity shop items.
   A)-> Barcode Scanner

2. The system will have Nielson's heuristic incorporated in its design.
   A)-> Cardview

3.The system will have the ability to use sorting or filtering on its data using special algorithms.

A)-> Filtering and Sorting

3. The system must be connected to a database, the database will have option to create extra backups, or the database provider could provide it by default. A)->Firebase

5.The systems needs to keep track of volunteers and managers in some form

A)-> Login and Register with Firebase.

6.The systems will need to store embed data about the charity shop via the shop's online documentation.

A)-> Help Activity

7.The systems will use any form of messages in order to communicate with user's phones

A)-> Messaging

The basic code while fit into one of each of this categories.

3.6. Conclusions

The plan is to choose Feature Driven Development (FDD) as the advantages that come with it fit the software methodology needs of the project e.g., using documentation over daily stands to explain the progress and the client-centred ness of the methodology.

FDD will be used to manage the workflow of this section as well as the testing and evaluation section which will be used to verify that the system design works as expected.

A blueprint of the front end e.g., High Fidelity Prototypes, Class Diagrams and Screen Flow Diagrams have been obtained and a blueprint of the structure of the back end has been created e.g., ERD. A tier diagram which depicts the connection between the front end and back end has been completed too.

These blueprints will help in the implementation of the project in code. Additionally, some of the tests and evaluations will be created with the blueprints in mind in the next chapter.

4. Experiment Development

4.1. Introduction

This section is about creating a code that will furtherly demonstrate that the software and software design is worth continuing, worth doing and possible. This section, the principles discussed in the Literature Review and Experiment Design are used to development a working application.

4.2. Software Development

The following snippets capture some of the working code that is used to power the most important features. From the front-end to back-end.

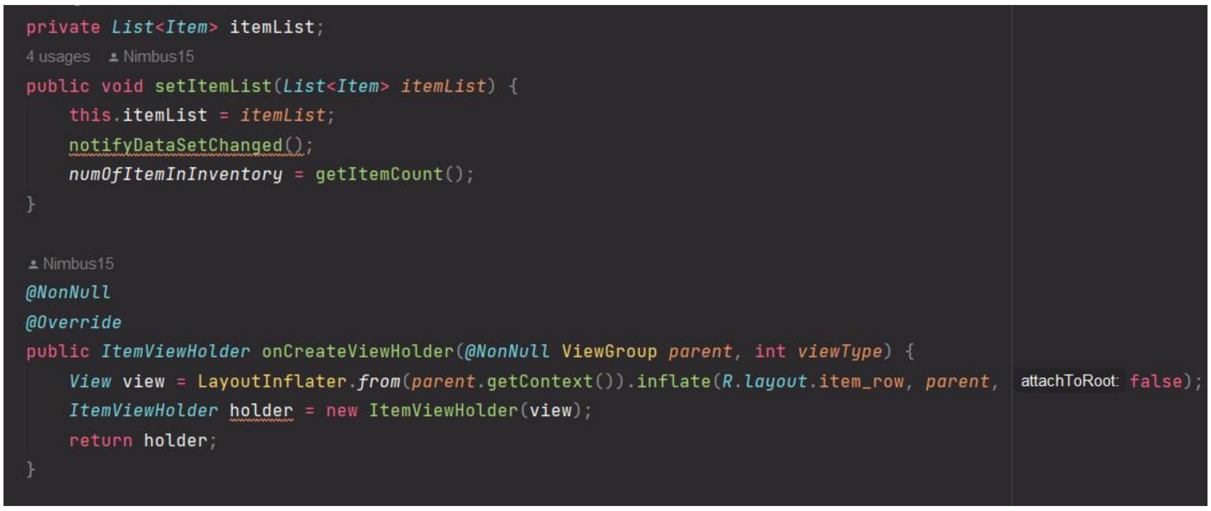
Front end skills

Recycler View Adapter

The Recycler View and Recycler View adapter is one the most important snippets of code. This is because the Recycler View is the element which display a quick preview of each record as a row as shown in the Inventory Page. The Recycler View allows for the user to scroll back to the first row after the user scrolls past the last element. Thus, it makes displaying data easier.

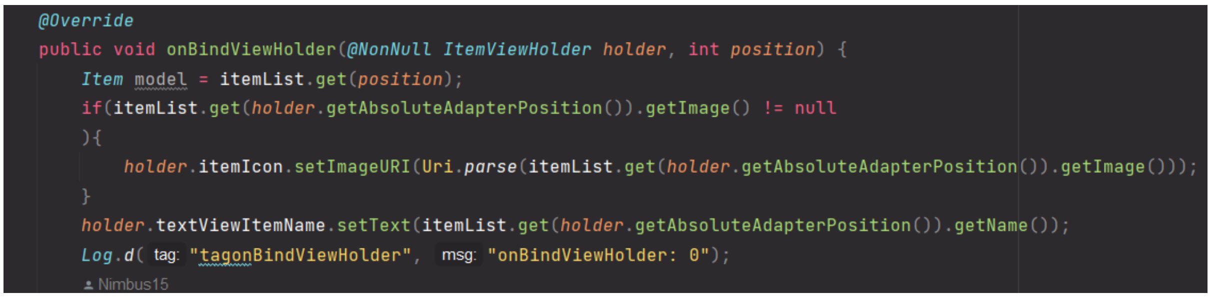
The Recycler View adapter or any other type of adapter is essential because it populates the details of each row from a data source. Data from a data source e.g. database is put into a view holder and that view holder along with a row layout is used to inflate a Recycler View with rows.

This Recycler View adapter has a couple of components as described below.



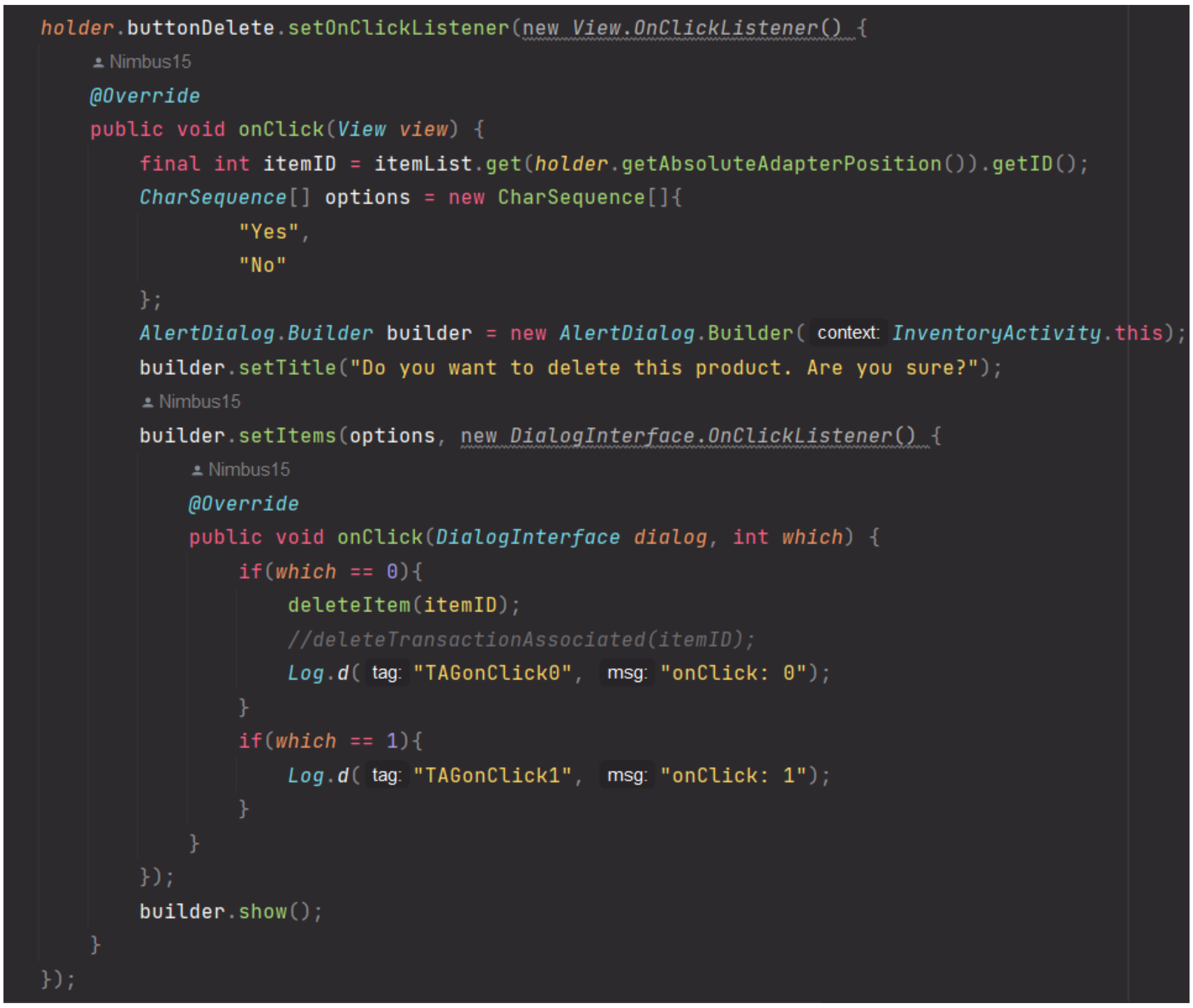
```
private List<Item> itemList;
4 usages   Nimbus15
public void setItemList(List<Item> itemList) {
    this.itemList = itemList;
    notifyDataSetChanged();
    numOfItemInInventory = getItemCount();
}

Nimbus15
@NonNull
@Override
public ItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_row, parent,  attachToRoot: false);
    ItemViewHolder holder = new ItemViewHolder(view);
    return holder;
}
```

The setItem List is similar to a constructor, it the first point of entry of a data source.
The onCreateViewHolder function in this case is used set up the instructions needed to inflate the RecyclerView.



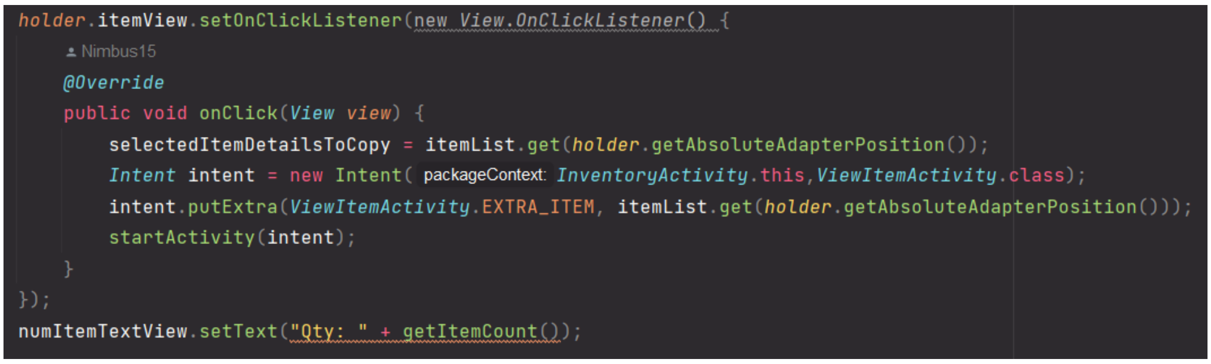
```
@Override
public void onBindViewHolder(@NonNull ItemViewHolder holder, int position) {
    Item model = itemList.get(position);
    if(itemList.get(holder.getAbsoluteAdapterPosition()).getImage() != null
    ){
        holder.itemIcon.setImageURI(Uri.parse(itemList.get(holder.getAbsoluteAdapterPosition()).getImage()));
    }
    holder.textViewItemName.setText(itemList.get(holder.getAbsoluteAdapterPosition()).getName());
    Log.d( tag: "tagonBindViewHolder",  msg: "onBindViewHolder: 0");
    Nimbus15
```

The onBindViewHolder function is the main worker of this Adapter because it is used to populate the RecyclerView with data.



```
holder.buttonDelete.setOnClickListener(new View.OnClickListener() {
    Nimbus15
    @Override
    public void onClick(View view) {
        final int itemID = itemList.get(holder.getAbsoluteAdapterPosition()).getID();
        CharSequence[] options = new CharSequence[]{
                "Yes",
                "No"
        };
        AlertDialog.Builder builder = new AlertDialog.Builder( context: InventoryActivity.this);
        builder.setTitle("Do you want to delete this product. Are you sure?");
        Nimbus15
        builder.setItems(options, new DialogInterface.OnClickListener() {
            Nimbus15
            @Override
            public void onClick(DialogInterface dialog, int which) {
                if(which == 0){
                    deleteItem(itemID);
                    //deleteTransactionAssociated(itemID);
                    Log.d( tag: "TAGonClick0",  msg: "onClick: 0");
                }
                if(which == 1){
                    Log.d( tag: "TAGonClick1",  msg: "onClick: 1");
                }
            }
        });
        builder.show();
    }
});
```

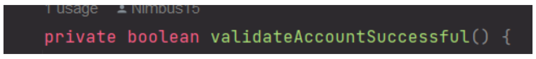
From the same onBindViewHolder, the deletion button is being assigned its deletion functionality.



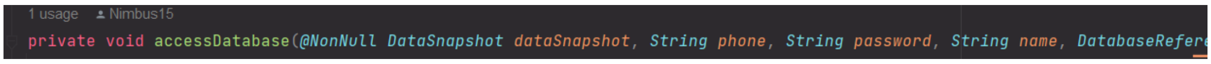
```
holder.itemView.setOnClickListener(new View.OnClickListener() {
    Nimbus15
    @Override
    public void onClick(View view) {
        selectedItemDetailsToCopy = itemList.get(holder.getAbsoluteAdapterPosition());
        Intent intent = new Intent( packageContext: InventoryActivity.this,ViewItemActivity.class);
        intent.putExtra(ViewItemActivity.EXTRA_ITEM, itemList.get(holder.getAbsoluteAdapterPosition()));
        startActivity(intent);
    }
});
numItemTextView.setText("Qty: " + getItemCount());
```

From the same onBindViewHolder, from clicking on a specific row, the application will be being the user to the corresponding expanded record of the row.

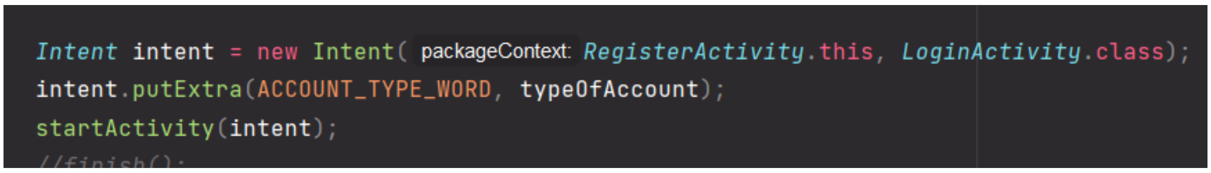
Login/ Register



```
  1 usage   Nimbus15
  private boolean validateAccountSuccessful() {
```



```
1 usage   Nimbus15
private void accessDatabase(@NonNull DataSnapshot dataSnapshot, String phone, String password, String name, DatabaseRefere
```



```
Intent intent = new Intent( packageContext: RegisterActivity.this, LoginActivity.class);
intent.putExtra(ACCOUNT_TYPE_WORD, typeOfAccount);
startActivity(intent);
//finish();
```

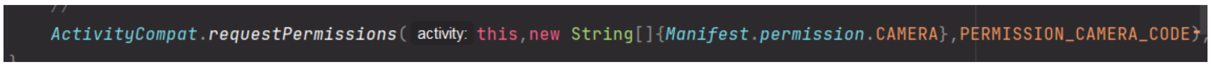
With the login and Register subcomponents. There are 3 recurring steps: Validation of the data inputted whether it is creating a new account or login into a existing account. There must be code to ensure that all important fields are filled in.

This is allowing the database to record data in a standard which makes it easier to find after creating a new account and makes the database faster to perform verification. Verification is checking whether an account's credentials match the credentials in the database.

Finally, the user is moved to the next activity if successful or asked to repeat the process again should they fail this stage.

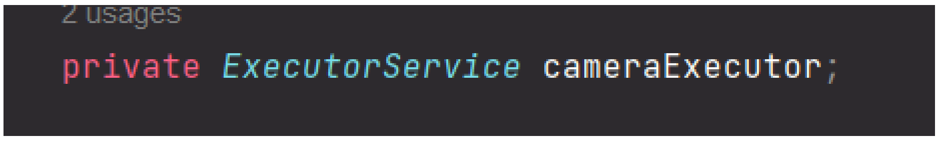
Barcode Scanner

The barcode scanner is another prominent part of the application. There are 5 main parts it: Permission, Thread, Barcode Option, Preview and Returning Barcode.



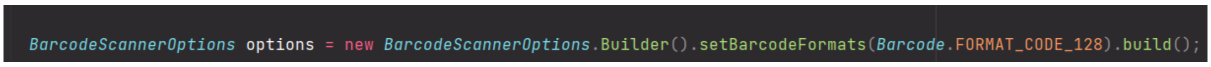
```
    ActivityCompat.requestPermissions( activity: this,new String[]{Manifest.permission.CAMERA},PERMISSION_CAMERA_CODE);
```

Asking for permission to access the features of a phone is required for any application. Without permission any interaction between the phone feature e.g. Camera and application will not work.



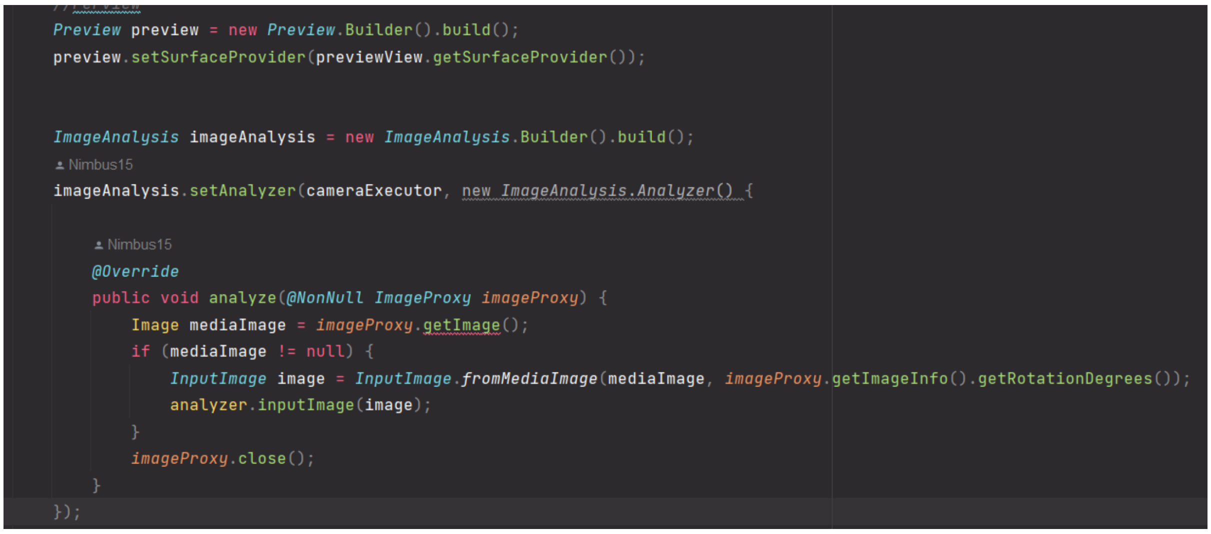
```
2 usages
private ExecutorService cameraExecutor;
```

It is a good practise to use a separate thread to handle the barcode activity. This reduces the strain on the main thread. The separate thread can be killed should it be unresponsive.



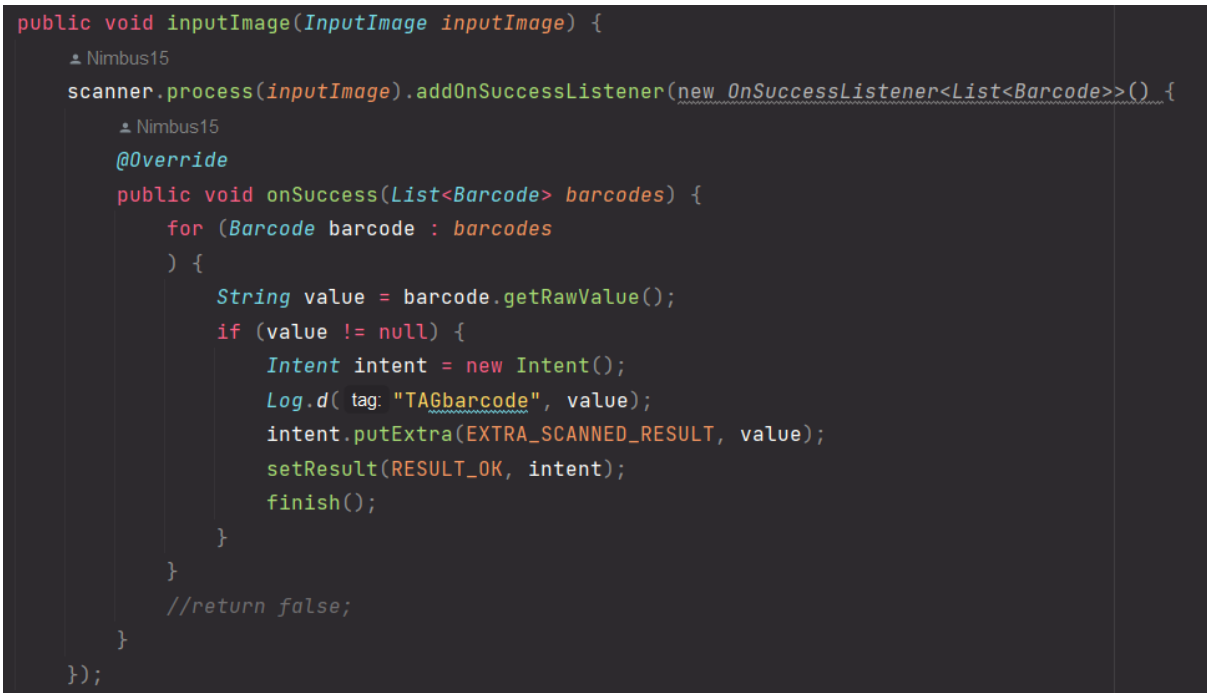
```
BarcodeScannerOptions options = new BarcodeScannerOptions.Builder().setBarcodeFormats(Barcode.FORMAT_CODE_128).build();
```

The barcode scanner options need to be sent before it can be used.



```
Preview preview = new Preview.Builder().build();
preview.setSurfaceProvider(previewView.getSurfaceProvider());


ImageAnalysis imageAnalysis = new ImageAnalysis.Builder().build();
Nimbus15
imageAnalysis.setAnalyzer(cameraExecutor, new ImageAnalysis.Analyzer() {

    Nimbus15
    @Override
    public void analyze(@NonNull ImageProxy imageProxy) {
        Image mediaImage = imageProxy.getImage();
        if (mediaImage != null) {
            InputImage image = InputImage.fromMediaImage(mediaImage, imageProxy.getImageInfo().getRotationDegrees());
            analyzer.inputImage(image);
        }
        imageProxy.close();
    }
});
```

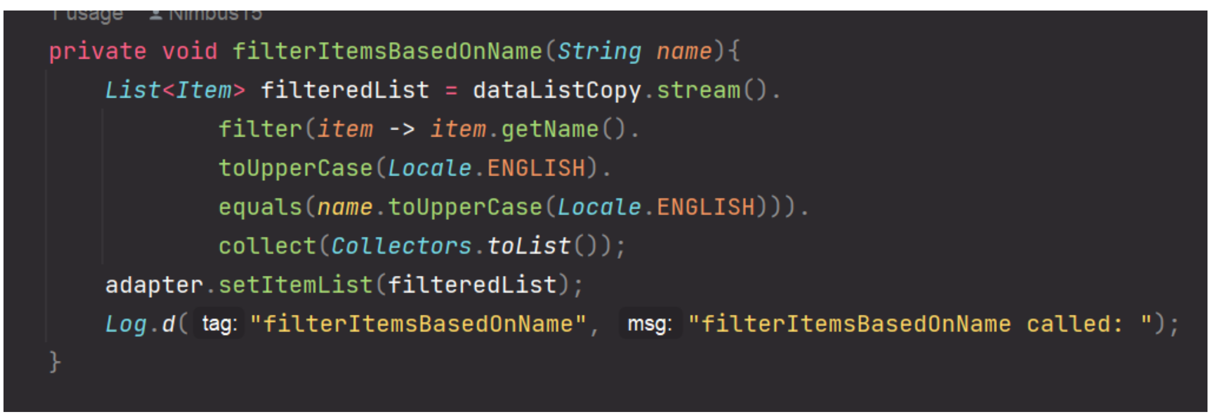
The barcode scanner will scan the barcode via the camera. A preview is used to show the user the output and allow them to aim the camera at the barcode.



```
public void inputImage(InputImage inputImage) {
    Nimbus15
    scanner.process(inputImage).addOnSuccessListener(new OnSuccessListener<List<Barcode>>() {
        Nimbus15
        @Override
        public void onSuccess(List<Barcode> barcodes) {
            for (Barcode barcode : barcodes
            ) {
                String value = barcode.getRawValue();
                if (value != null) {
                    Intent intent = new Intent();
                    Log.d( tag: "TAGbarcode", value);
                    intent.putExtra(EXTRA_SCANNED_RESULT, value);
                    setResult(RESULT_OK, intent);
                    finish();
                }
            }
            //return false;
        }
    });
```

Finally, a barcode needs to be return to the calling activity.

Filter / Search

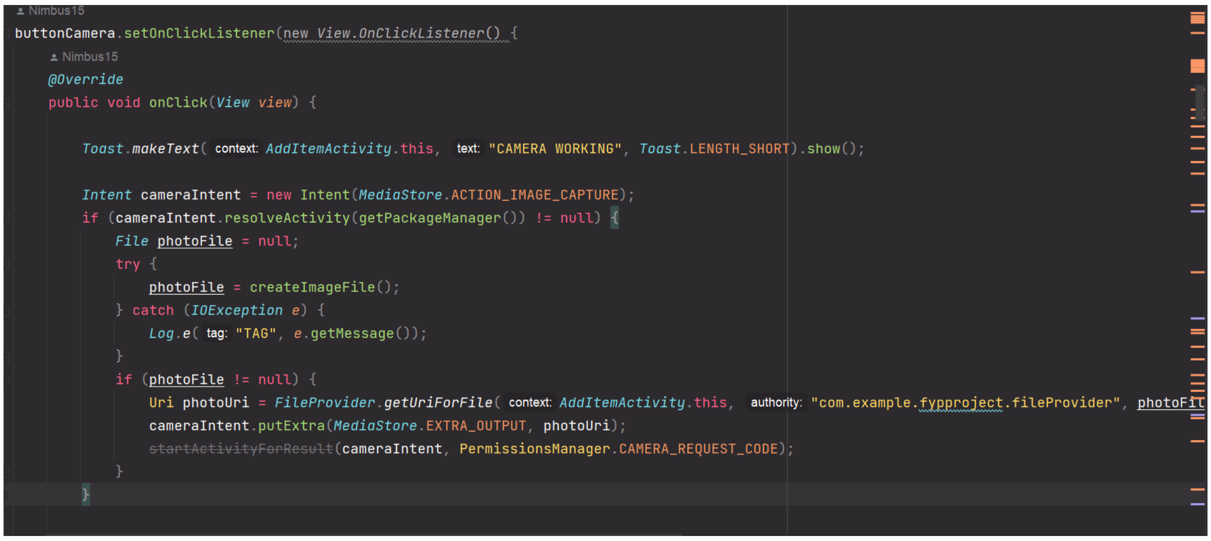


```
  1 usage   Nimbus15
  private void filterItemsBasedOnName(String name){
      List<Item> filteredList = dataListCopy.stream().
              filter(item -> item.getName().
              toUpperCase(Locale.ENGLISH).
              equals(name.toUpperCase(Locale.ENGLISH))).
              collect(Collectors.toList());
      adapter.setItemList(filteredList);
      Log.d( tag: "filterItemsBasedOnName",  msg: "filterItemsBasedOnName called: ");
  }
```

With filtering in general, it is a good idea to create a copy of the original data from the data source. All operations should be performed on that copy. To allow the filter to work operations such as changing everything had to be perform. This is because the original data would not be lost in the process and can be displayed after the filter is reset.

The adapter’s list then needs to be changed so it displays the changed data to the user.

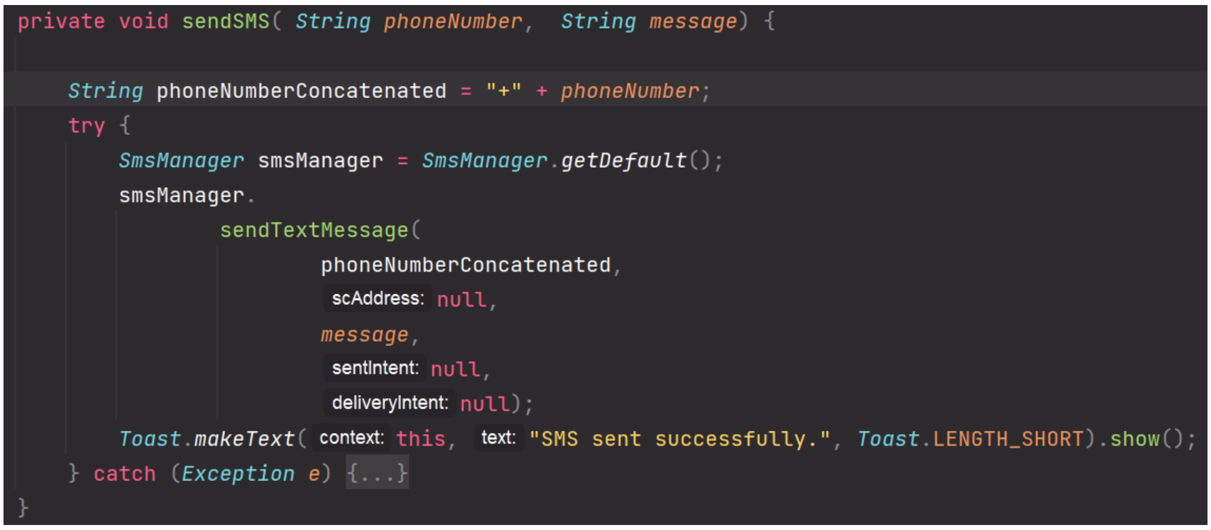
Camera



```
Nimbus15
buttonCamera.setOnClickListener(new View.OnClickListener() {
    Nimbus15
    @Override
    public void onClick(View view) {

        Toast.makeText( context: AddItemActivity.this,  text: "CAMERA WORKING", Toast.LENGTH_SHORT).show();

        Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (cameraIntent.resolveActivity(getPackageManager()) != null) {
            File photoFile = null;
            try {
                photoFile = createImageFile();
            } catch (IOException e) {
                Log.e( tag: "TAG", e.getMessage());
            }
            if (photoFile != null) {
                Uri photoUri = FileProvider.getUriForFile( context: AddItemActivity.this,  authority: "com.example.fypproject.fileProvider", photoFil
                cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri);
                ~~startActivityForResult~~(cameraIntent, PermissionsManager.CAMERA_REQUEST_CODE);
            }
        }
```

The way the camera is accessed in this application is dissimilar to the way the barcode scanner is called. The MediaStore is an android call which provides access to media files on the phone. While the FileProvider a “Provider” or interface in android which allows for the modification of media files. In this case, a photo uri is need which has the path of where the image taken by the camera is stored.

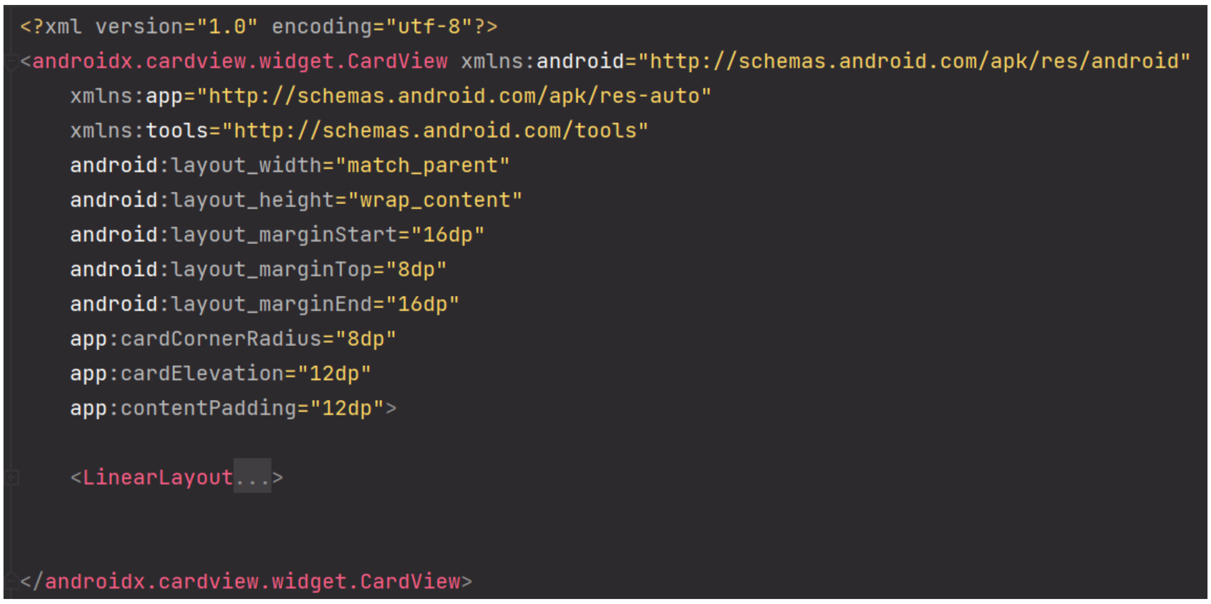
Send Messages



```
private void sendSMS( String phoneNumber,  String message) {

    String phoneNumberConcatenated = "+" + phoneNumber;
    try {
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.
                sendTextMessage(
                        phoneNumberConcatenated,
                         scAddress: null,
                        message,
                         sentIntent: null,
                         deliveryIntent: null);
        Toast.makeText( context: this,  text: "SMS sent successfully.", Toast.LENGTH_SHORT).show();
    } catch (Exception e) {...}
}
```

This code uses a library to send a message to a specific number. These can be passed into the sendSMS function here. (Google, no date) was used to help with the development of the barcode scanner.

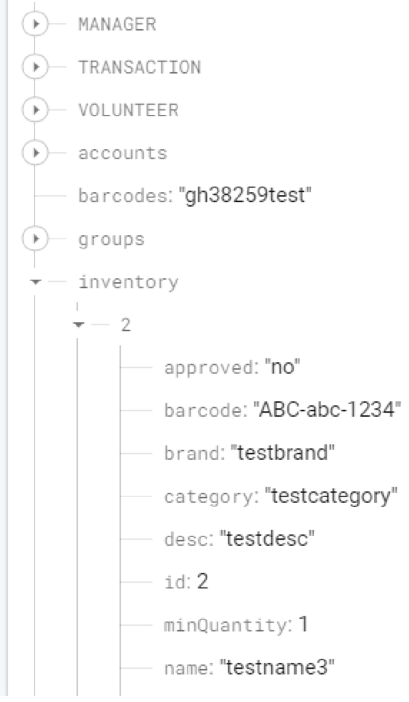
CardView



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="16dp"
    app:cardCornerRadius="8dp"
    app:cardElevation="12dp"
    app:contentPadding="12dp">

    <LinearLayout...>


</androidx.cardview.widget.CardView>
```

A card view is used to group elements together and acts as a panel. It will help with organising and beatifying a screen.

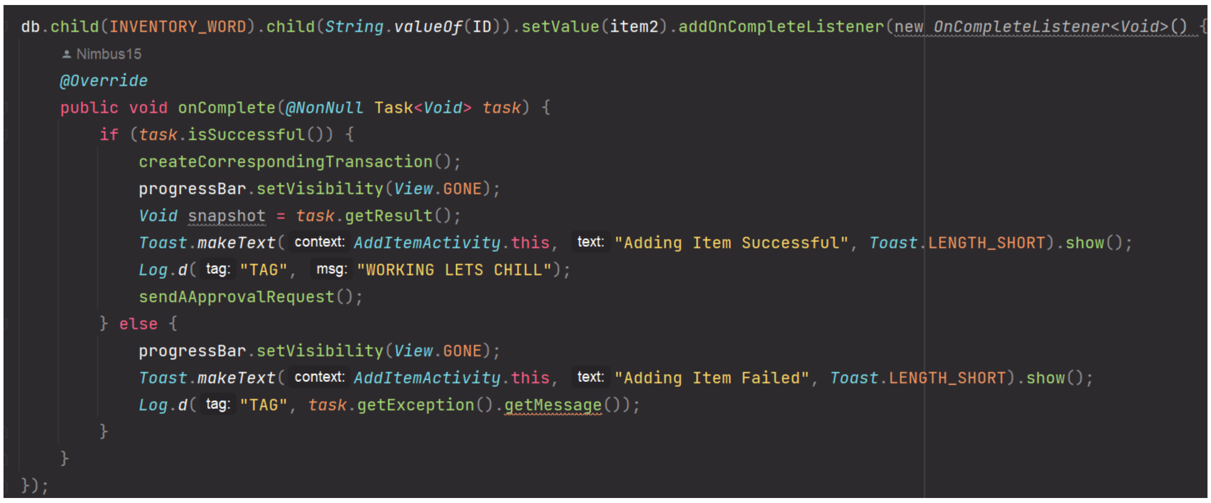
Database Skills



```
MANAGER
TRANSACTION
VOLUNTEER
accounts
barcodes: "gh38259test"
groups
inventory
  2
    approved: "no"
    barcode: "ABC-abc-1234"
    brand: "testbrand"
    category: "testcategory"
    desc: "testdesc"
    id: 2
    minQuantity: 1
    name: "testname3"
```

The above image shows a sample of the structure of a database in the Firebase Realtime database. By applying similar logic other database types such the Firebase Cloud Firestore and the Firebase Storage can also be accessed. There are multiple ways to perform C.R.U.D using Firebase. The way that is shown below gives greater customization.

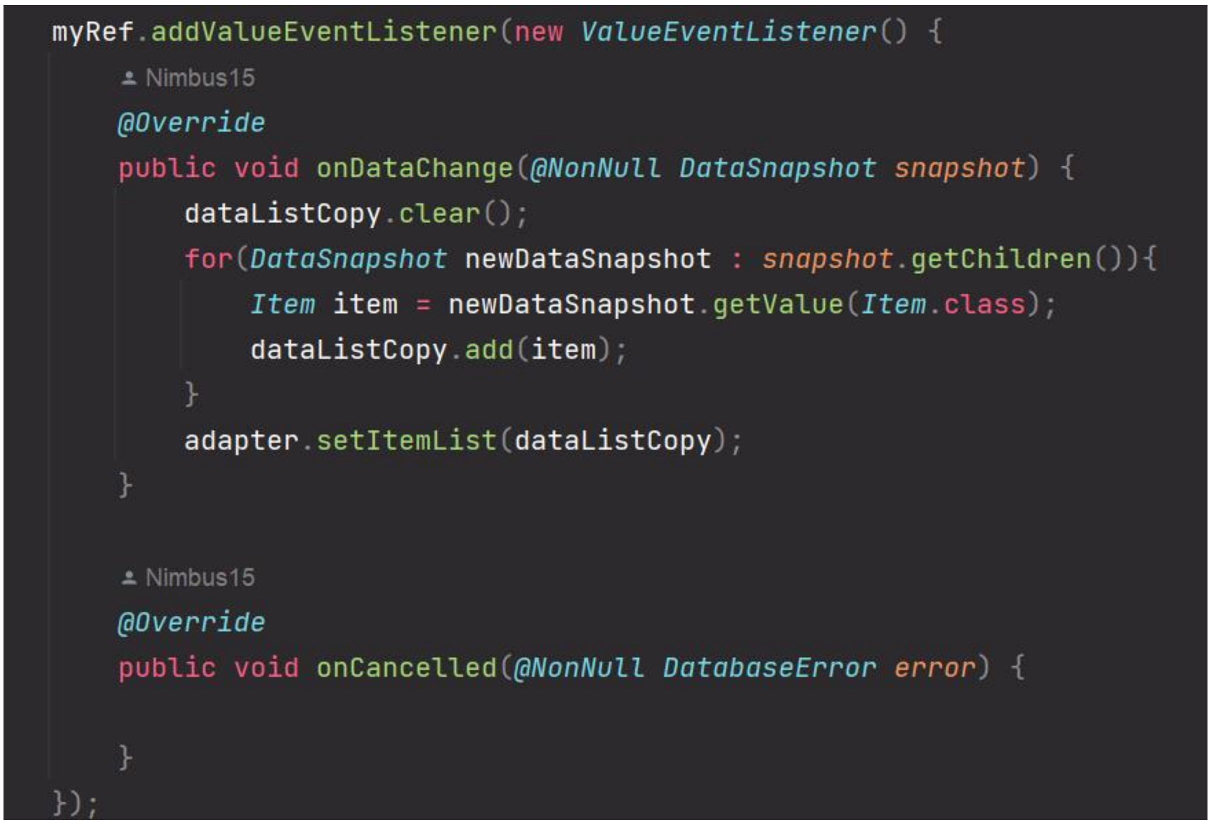
Create



```
db.child(INVENTORY_WORD).child(String.valueOf(ID)).setValue(item2).addOnCompleteListener(new OnCompleteListener<Void>() {
    Nimbus15
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if (task.isSuccessful()) {
            createCorrespondingTransaction();
            progressBar.setVisibility(View.GONE);
            Void snapshot = task.getResult();
            Toast.makeText( context: AddItemActivity.this,  text: "Adding Item Successful", Toast.LENGTH_SHORT).show();
            Log.d( tag: "TAG",  msg: "WORKING LETS CHILL");
            sendAApprovalRequest();
        } else {
            progressBar.setVisibility(View.GONE);
            Toast.makeText( context: AddItemActivity.this,  text: "Adding Item Failed", Toast.LENGTH_SHORT).show();
            Log.d( tag: "TAG", task.getException().getMessage());
        }
    }
});
```

In this case the root path for the database is needed, sub nodes/ branches/ children can then be set to the value of the class/ entity chosen. In this case it is an item class that was previous described in Chapter 3.

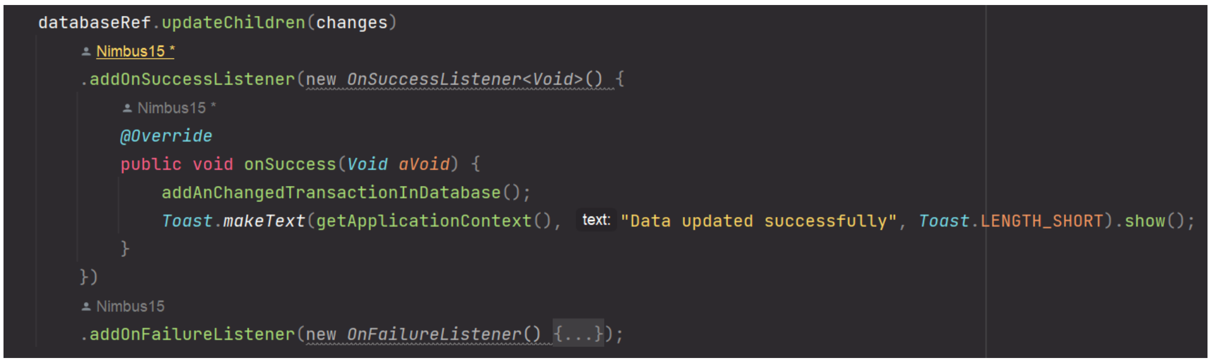
Read



```
myRef.addValueEventListener(new ValueEventListener() {
    Nimbus15
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        dataListCopy.clear();
        for(DataSnapshot newDataSnapshot : snapshot.getChildren()){
            Item item = newDataSnapshot.getValue(Item.class);
            dataListCopy.add(item);
        }
        adapter.setItemList(dataListCopy);
    }

    Nimbus15
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Using snapshot of the database and its sub branches, each record of an “Item” in the database can be retrieved as an “Item” object that was described in chapter3.

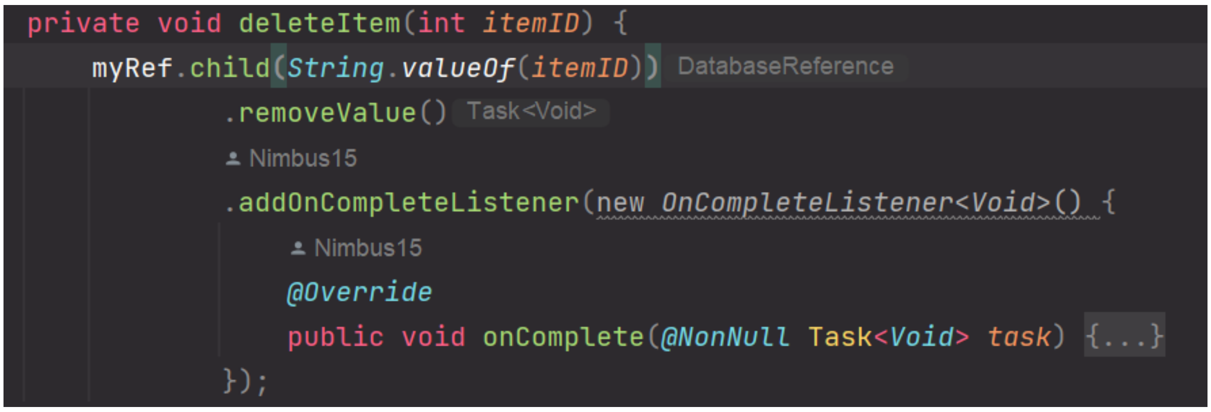
Update



```
databaseRef.updateChildren(changes)
    ≗ Nimbus15 *
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        ≗ Nimbus15 *
        @Override
        public void onSuccess(Void aVoid) {
            addAnChangedTransactionInDatabase();
            Toast.makeText(getApplicationContext(),  text: "Data updated successfully", Toast.LENGTH_SHORT).show();
        }
    })
    ≗ Nimbus15
    .addOnFailureListener(new OnFailureListener() {...});
```

Provided that database reference or the path to the database is correct. Use a hashmap to store all changes need to be perform on a node. This hashmap can then be passed to make changes to the database.

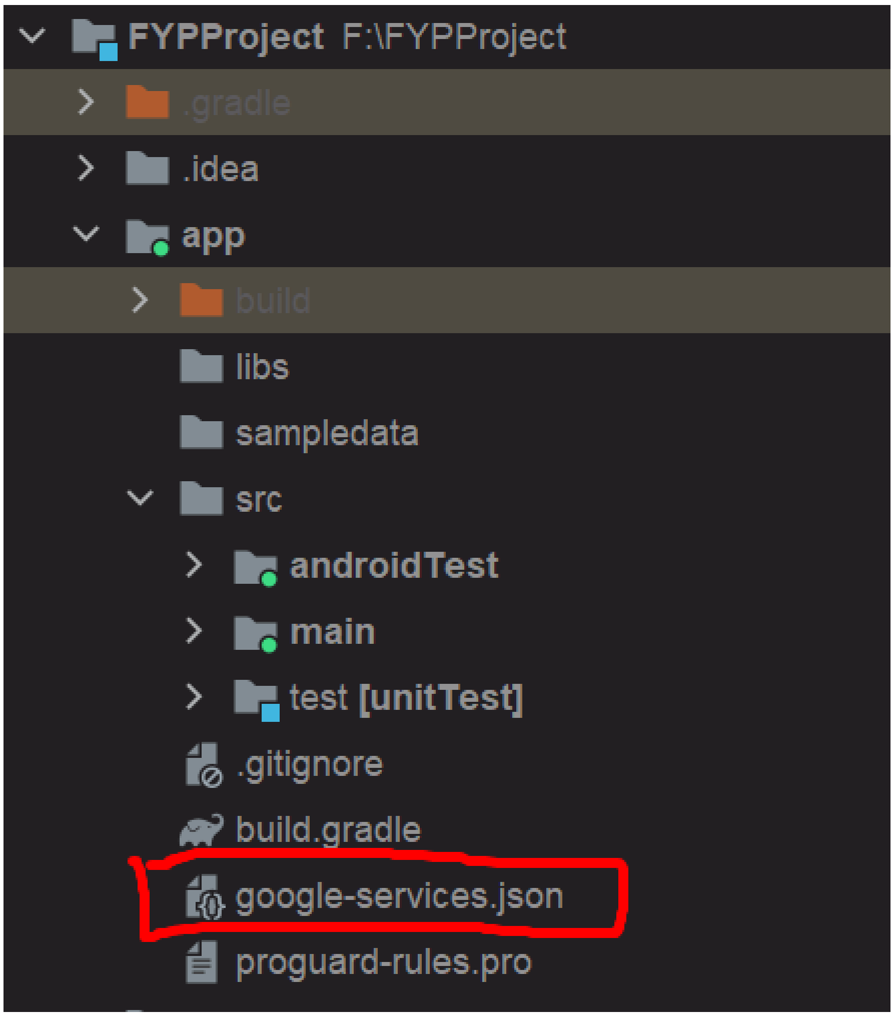
Delete



```
private void deleteItem(int itemID) {
    myRef.child(String.valueOf(itemID)) DatabaseReference
            .removeValue() Task<Void>
            ≗ Nimbus15
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                ≗ Nimbus15
                @Override
                public void onComplete(@NonNull Task<Void> task) {...}
            });
```

Similar to above a database reference/ path to the database is needed, on the path, the removeValue() method can be used to delete a node.

Connect Frontend to Backend



By following the instruction listed in the Firebase website for each particular type of database, it is possible to connect the Front-end to the Back-end. A Json file with all the configuration needs to be put into the application folder.

4.3. Conclusions

The Experiment Development has demonstrated that it is possible the implement the project in a code form. From these complex features samples, it is shown that it within capacity of the student to build this project. All the complexity of the project helps the project meet its user requirements.

From these same complex features most of the logic can be applied to tackle the creation of more basic features or of more complex features.

This is a clear signal that this project has made progress as well as been feasible. In the next chapter, the ways to test and evaluate this Experiment Developed application will be discussed.

5. Testing and Evaluation

5.1. Introduction

There are two parts to verifying that the software works. The main purpose of this system is to come up with a two-part plan.

One part is testing the software code for bugs, unexpected output, etc. This will be done through various testing types i.e., Through the test listed in the “testing triangle”. The tests need to demonstrate that the code will do exactly what the system requirements list.

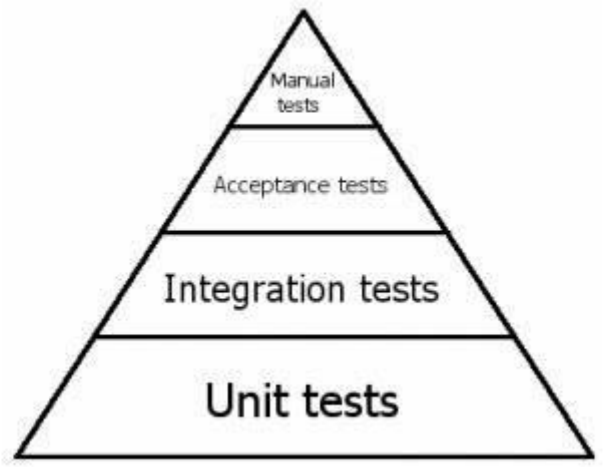
The second part is to test the software functionality that the code performs. This will entail testing the system functionality and enlisting some volunteers to try out the application. The feedback will help improve the functionality of the software. The functionality needs exactly what the user requirements needs.

5.2. System Testing

This means testing the code of the software. There were a couple of tests that needed to be performed. The goal is to look for bugs or code that does not do as expected.

Software testing

Software Testing means verifying that software does exactly what the programmer intend it to do and making sure that any error such as bugs is found and sorted. According to (Hamilton, 2020), it includes using software tools which can be automated in order to check an area of concern.



Manual testing

This means the developer will manually go through the code using an IDE. Pen and paper will be used to keep note of bugs and other problems.

Unit testing
Unit testing means testing each module of a project e.g. Within one class. The plan will be to use Junit5 to test the classes(modules) from within.

Integration Testing
Integration testing means testing the interaction of a module with another module e.g. Between two classes or more. The plan is to use Cypress to test the endpoint or use Swagger to test the endpoints.

User Acceptance Testing (UAT)
This means inviting potential customers to test the modules of the project. There will be a checklist of modules with a rating system to allow users to get feedback from the user.

5.3. System Evaluation
This means testing the functionality of the software. Two ways to do this are explained below. The goal is to make sure that the system meets the user requirements.

Heuristics Evaluation
This means evaluating the project using Nielson’s 10 heuristics. From above in section 3, there were a couple of charts with heuristics. There will be a heuristic rating of the project using Nielson’s 10 heuristics.

Usability testing
This means inviting a user to test the functionality of a project. The users could test to do a particular functionality e.g., Create 5 items, this will be used to test the speed of the interaction.

5.4. Conclusions
A plan to test the code and functionality of the code has been explored. They will be used to verify the software and software design will work and do as intend. It is

currently too early to determine the exact details of each test however an outline will help the project in the future.

The next section is to discuss any more complication that the project may reach and also help plan the application's future.

6. Conclusions and Future Work

6.1. Introduction

The purpose of this chapter is to think ahead and plan. This includes identifying issues and risks and then tackling them. Plan the future workload. Some of the information is still relieve from the Interim Report.

Finally plan the entire schedule e.g., Gantt Chart. In essence, it is the implementation plan which will be needed for the implementation. This chapter will help prime for the incoming workload. It will also help explain to others what workload is expected.

6.2. Issues and Risks

Issues are things that have happened or are currently happening. Examples of such issues are the following.

1. The application needs to be aware of non-tech-savvy users who need extra support when using the application. The plan is to follow the heuristic to fix this e.g., use everyday language.
2. The application might not be able to handle the amount of content of a charity shop. The plan is to make the application expandable.
3. Some parts of the code could be more difficult to write. The plan is to ask for help when needed.

Risks are things that may happen. Examples of such risks are the following:

1. Underestimating user hits, after a certain number of hits, a premium plan is need continue using firebase. The plan is to estimate the number of hits or use the application to track more important goods.
2. The application may be made too complex to use from the perspective of a charity shop.
3. The application may not be suitable for the charity shop. E.g., Data types. There is a need to double-check with charity shops systems.
4. Outside forces may interfere with the application data in firebase. Ensure that security is strict in the beginning using best coding practices.

6.3. Plans and Future Work

The tasks in this paragraph can be done in parallel. The code can be researched and then written. After each piece of features, it can be documented in the dissertation. The plan is to create the front-end application using Android Studio. Some of the features will be implemented using a library (java library/google library). This may provide to be a difficult part of the project therefore assistance from experts may be needed. Further research on how to implement front-end parts e.g., the input page will need to be done. Some examples such as adding images from the gallery or camera of the items can be quite difficult. The final year dissertation will need to be started too. Tests will additionality need to be written in between both development phases of the front-end and back-end. This will result in a partly finished front-end in Android Studio(Java), front-end testing in Junit, front-end research and begin of the dissertation i.e., software design section.

The second part of the development will need to be the back-end which cannot be done alongside the front-end as it leads to confusion. The back-end application is implemented using firebase. Firebase's database e.g., Realtime database needs to be set up e.g. Security rules. The Firebase "Storage" must be set up as well. The back-end software part, the back-end testing using (Cypress or Postman) and the over half of the dissertation i.e., software design section will be finished as a result of this part.

Finally, the frontend will need to be connected to the backend. The front connects to the backend using a firebase library which uses an SSH connection. This will be completed as well as cleaning any other part of the dissertation e.g., the Intro section, conclusion and etc.

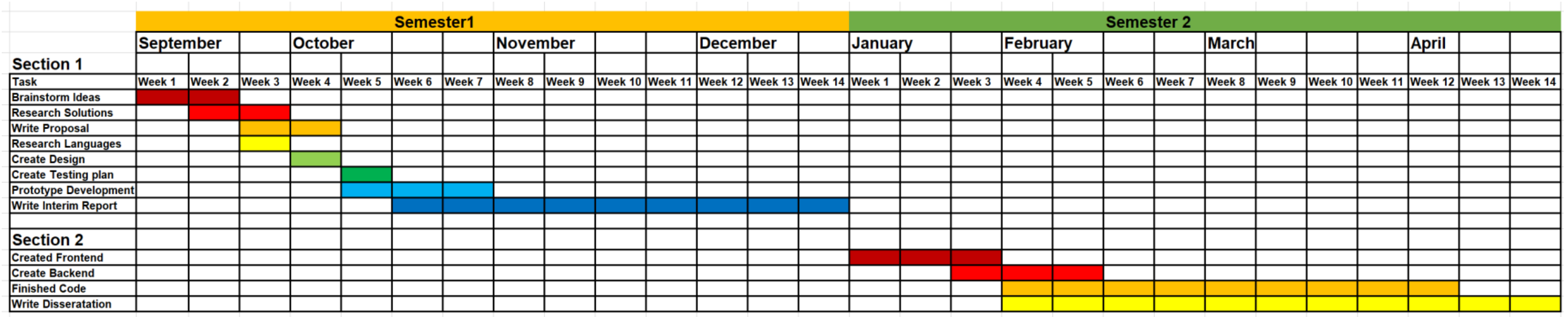
Throughout the project, the plan is to make the code open-ended to modification in the future.

All the complexity of the project helps the project meet its user requirements.

If there is extra time in the end. There is a plan to involve and discuss the project with other charities in order to get further feedback from them.

6.3.1. GANTT Chart

This is the plan for the process of the workload. The processes are split into two parts e.g., section 1 which is the research and documentation part and section 2 which is the code development part. Section 1 all the research should be done before semester 1week 5. After week 6 begins, the interim report will be written in parallel with any finishing touches to the research. Semester 2 has a similar approach, most of the code started before semester 2 week 6. After this week, the development will be done in parallel with writing the dissertation.



	Semester1														Semester 2													
	September			October				November				December			January			February				March				April		
Section 1																												
Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Brainstorm Ideas	🟥	🟥																										
Research Solutions		🟥	🟥																									
Write Proposal			🟧	🟧																								
Research Languages			🟨																									
Create Design				🟩																								
Create Testing plan					🟩																							
Prototype Development					🟦	🟦	🟦																					
Write Interim Report						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦														
Section 2																												
Created Frontend															🟥	🟥	🟥											
Create Backend																	🟥	🟥	🟥									
Finished Code																		🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧		
Write Disseratation																		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

Bibliography

Back4app (2021) ‘Top 10 Advantages of Firebase’, 23 February. Available at: https://blog.back4app.com/advantages-of-firebase/ (Accessed: 17 January 2023).

Beladiya, K. (no date) *Flutter App Development - Advantages And Drawbacks*. Available at: https://www.c-sharpcorner.com/article/flutter-app-development-advantages-and-drawbacks/ (Accessed: 17 January 2023).

Google (no date) *Scan barcodes with ML Kit on Android | Google for Developers*. Available at: https://developers.google.com/ml-kit/vision/barcode-scanning/android (Accessed: 7 July 2023).

Hamilton, T. (2020) *What is Software Testing? Definition*. Available at: https://www.guru99.com/software-testing-introduction-importance.html (Accessed: 22 January 2023).

Javatpoint (no date) *Advantages and disadvantages of Java - Javatpoint*, *www.javatpoint.com*. Available at: https://www.javatpoint.com/advantages-and-disadvantages-of-java (Accessed: 6 December 2022).

Lucidchart (2019) *Why (and How) You Should Use Feature-Driven Development*. Available at: https://www.lucidchart.com/blog/why-use-feature-driven-development (Accessed: 19 January 2023).

Magenest (2021) ‘Mobile application vs Web application: What is different?’, *Magenest - One-Stop Digital Transformation Solution*, 2 April. Available at: https://magenest.com/en/mobile-application-vs-web-application/ (Accessed: 5 December 2022).

Muller, M. (2019) *Essentials of Inventory Management*. HarperCollins Leadership.

Schreibfeder, J. (no date) ‘The First Steps to Achieving Effective Inventory Control’. Available at: https://www.boyerassoc.com/wp-content/uploads/2013/07/white-paper-the-first-steps-to-achieving-effective-inventory-control.pdf.

TheDeveloperBlog (no date) *SQLite Advantages and Disadvantages*. Available at: https://thedeveloperblog.com/sqlite/sqlite-advantages-and-disadvantages (Accessed: 17 January 2023).

Vadalia, B. (2021) ‘ReactJS and React Native - Key Difference, Advantages, and Disadvantages’, *Medium*, 29 July. Available at: https://bhumin-vadalia.medium.com/reactjs-and-react-native-key-difference-advantages-and-disadvantages-ceb197f4d31d (Accessed: 17 January 2023).