Charity Shop App
Interim Report

DT228
BSc in Computer Science

Alex Bang

C19409486

Damian Bourke

School of Computer Science

Technological University, Dublin

14/11/22

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

14/11/22

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 created in this stage.

Chapter 3 - System Design is about creating the blueprints in terms of methodology and system. Artefacts will be created in this stage.

Chapter 4 -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 5 -Prototype Development is a piece/ existing project used to demonstrate the project can work. This will check the feasibility of the project.

Chapter 6 -Issues and Future Work is an outline of the next part of the process which is developing the application. This will create a further plan to be discussed.

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 to Your Problem

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. 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.

2.4. Technologies you've 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).

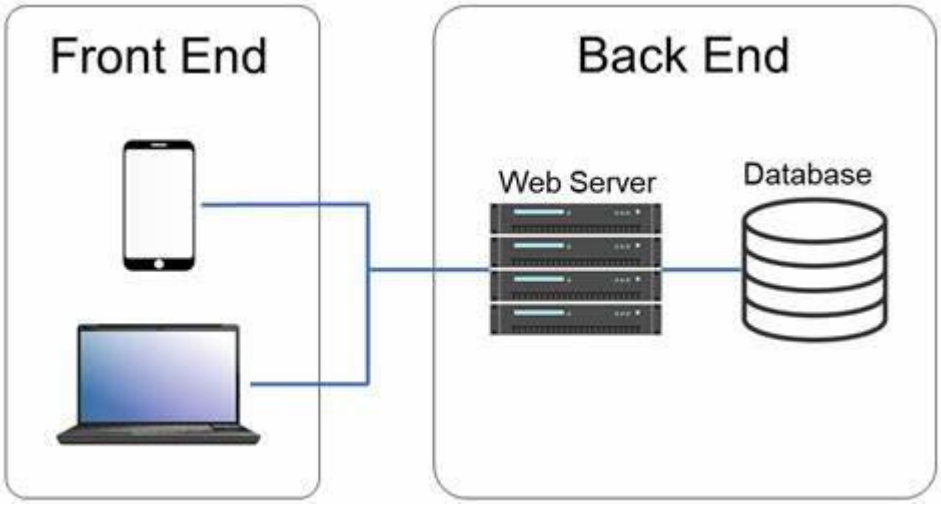
(Vadalia, 2021) mentions that there is a property in ReactJS which is data backlinking. In this context, ReactJS components can be directly manipulated, however, changes in child structure data do not affect the parent structure data.

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.

2.5. 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. System 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 the system's blueprint 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 the blueprint.

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.

Scrum

Is a development framework which is used to tackle the cycle of "product development" according to (AgileAlliance, 2017). Teams used scrum to forge a way of doing things e.g. writing code, testing the code and reflecting on the code.

According to (Vasiliauskas, 2022), scrum is very "adaptable and flexible" and is ideal for work that demands an adaptable and flexible approach e.g. When requirements are unconcise/ less known. Scrum promotes the creation of innovative ideas. Teams are encouraged to coordinate efforts and collate ideas from their team members.

According to (Vasiliauskas, 2022), scrum's delivery times are way faster than other approaches, which reach the market quicker. This approach is also known to be cheaper due to the less documentation and less oversight compared to other approaches.

According to (Vasiliauskas, 2022), scrum has often resulted in "higher quality work" due to the increased trust among team members and the lack of oversight. This results in the team members being given the full "responsibility and ownership of their work".

(Vasiliauskas, 2022) mentioned that scrum's daily stand-up meeting "ensures continuous feedback" which allows the team to manipulate their project using the feedback provided. Thus, increasing client satisfaction.

Figure 6 is a diagram of Scrum.

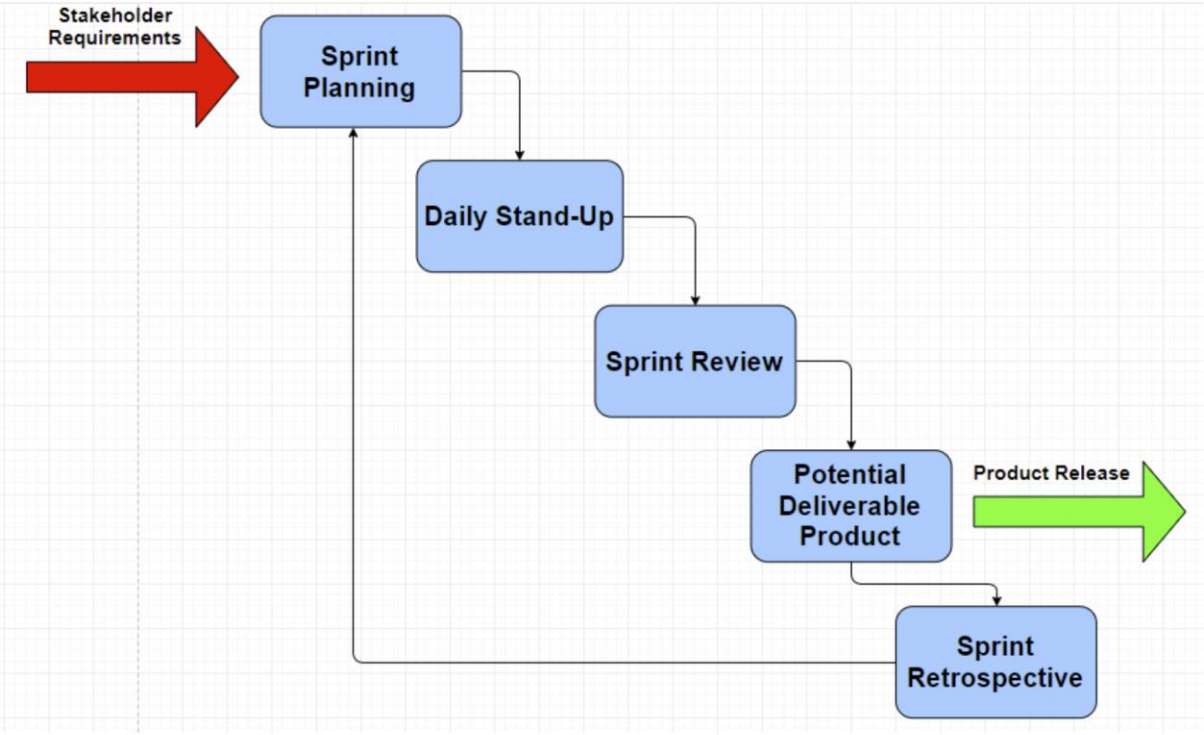


Figure 6 - Scrum Methodology

Feature Driven Development

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 Scrum.

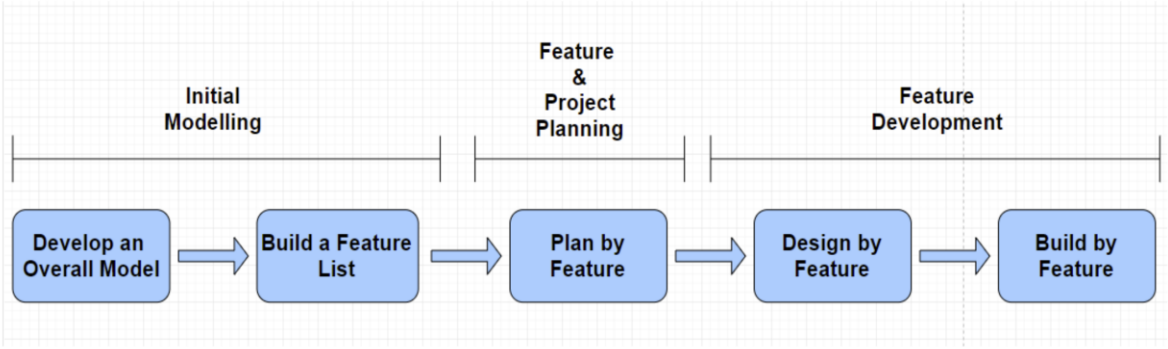


Figure 3 - Feature Driven Development

DevOps

DevOps is considered by some to be another framework for managing software. According to (Atlassian, no date) devops combines "practices, tools and cultural philosophy" which automates and combines the processes used by both software development and IT teams.

According to (Atlassian, no date) DevOps is known for speed. Teams that use DevOps have more frequent deliverables which are of higher standards. DevOps application allows continuous delivery of the building, testing and releasing software.

DevOps is known to induce a "culture of collaboration" within a business from developers to operation teams. According to (Atlassian, no date) all of the members carry a shared responsibility and workload of the project. This encourages team members and makes them more effective and time-saving.

More frequent and faster releases are attributed to devops according to (Atlassian, no date). New features release and faster bug fixing are a symptom of devops.

DevOps is also known to be secure due to the merging of software processes (integration and delivery and security process e.g. audits and testing into a single continuous pipeline as described by (Atlassian, no date).

Below is a diagram of DevOps.

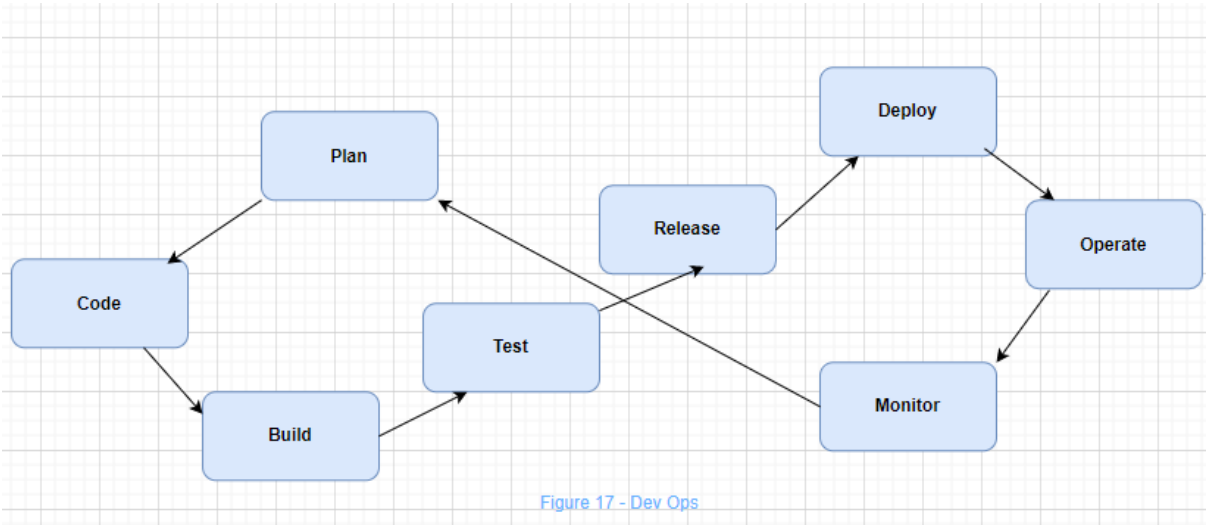


Figure 17 - Dev Ops

Waterfall

The waterfall model is a strict step-by-step phase methodology. Steps depend on inputs from the previous step. Construction and engineering design are more suited to waterfall according to (Gaille, no date). Waterfall uses a more precise and strict series of steps compared to other methodologies. The series of steps are the same for other projects due to the simple nature of the structure. Teams must finish one step completely in order to move to the next step as described by (Gaille, no date). Waterfall is known to be very intuitive. This means that there is a very “steep learning curve” as mentioned by (Gaille, no date). Beginners will find this model easier to stick with. Unlike other methodologies, waterfall does not need special training.

Waterfall binds a project with a determined deadline. All the steps in a waterfall model are well set. The company knows exactly what to expect from the completion of one step as described by (Gaille, no date). They can then compare this with their plan. This cuts out any hanging project which may result in unexpected delays to the step completion.

Waterfall is known to build good habits. Teams can craft their own style to do things within the steps. This builds a “organised and disciplined approach” to each step according to (Gaille, no date). This is because all tasks need to be finished within a set timeframe and cannot move to the next stage without completing the previous steps.

Figure 4 is a diagram of waterfall.

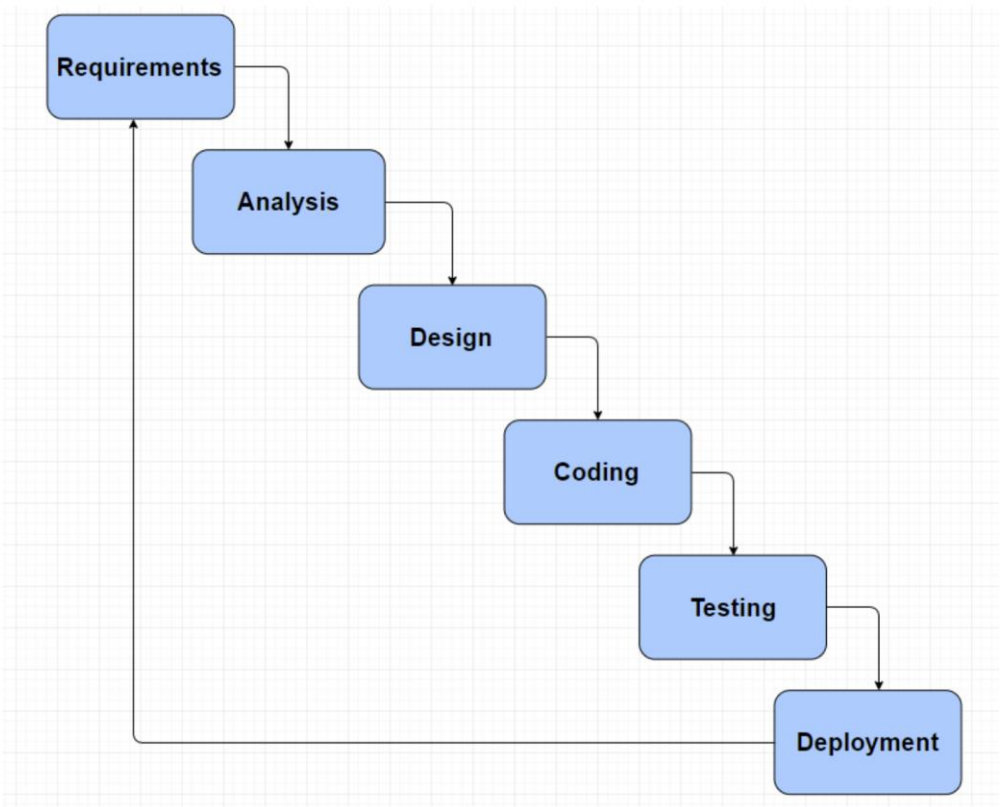
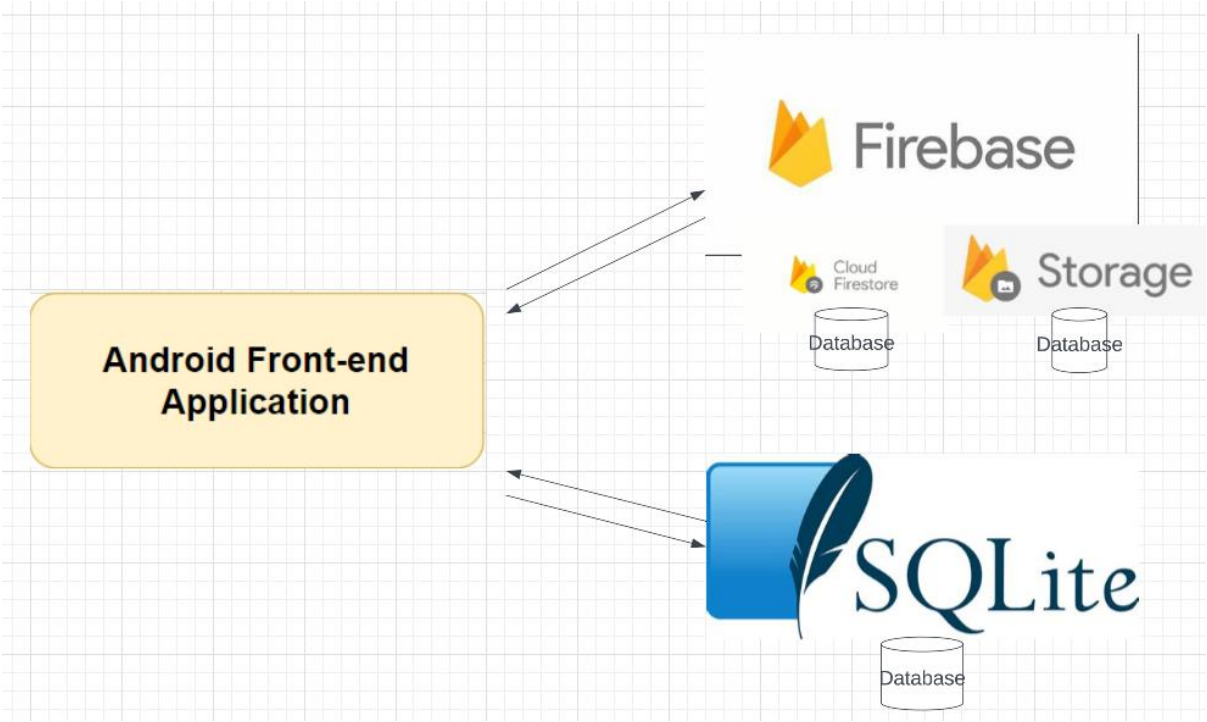


Figure 4 - Waterfall Model

3.3. Overview of System

The proposed systems will consist 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



This diagram above illustrates the software architecture. Here there is an android front-end application connected to an SQLite database which is the local database on the android device. This will allow the application to run offline. There is also a connection to a BAAS provider known as Firebase. The Firestore 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.

Front end

The front end consists of an input page, invoice page, main page 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.

Invoice page (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.

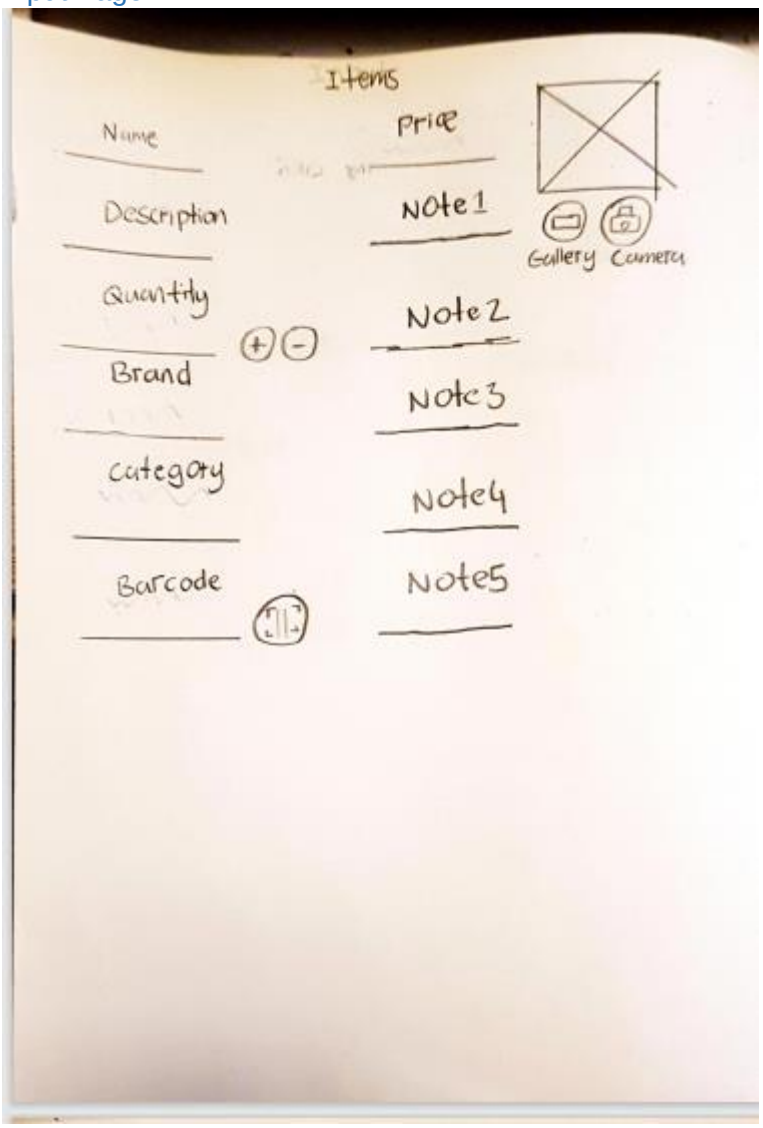
Main page will contain various navigation options.

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.

Sidebar Menu is an inbuilt menu sidebar which pops from the side and acts as quick navigation. Screen flow diagram shows the flow of the diagram. There will be a “three slash” button which is used to access the sidebar menu, it will be located on almost every page in the top left corner.

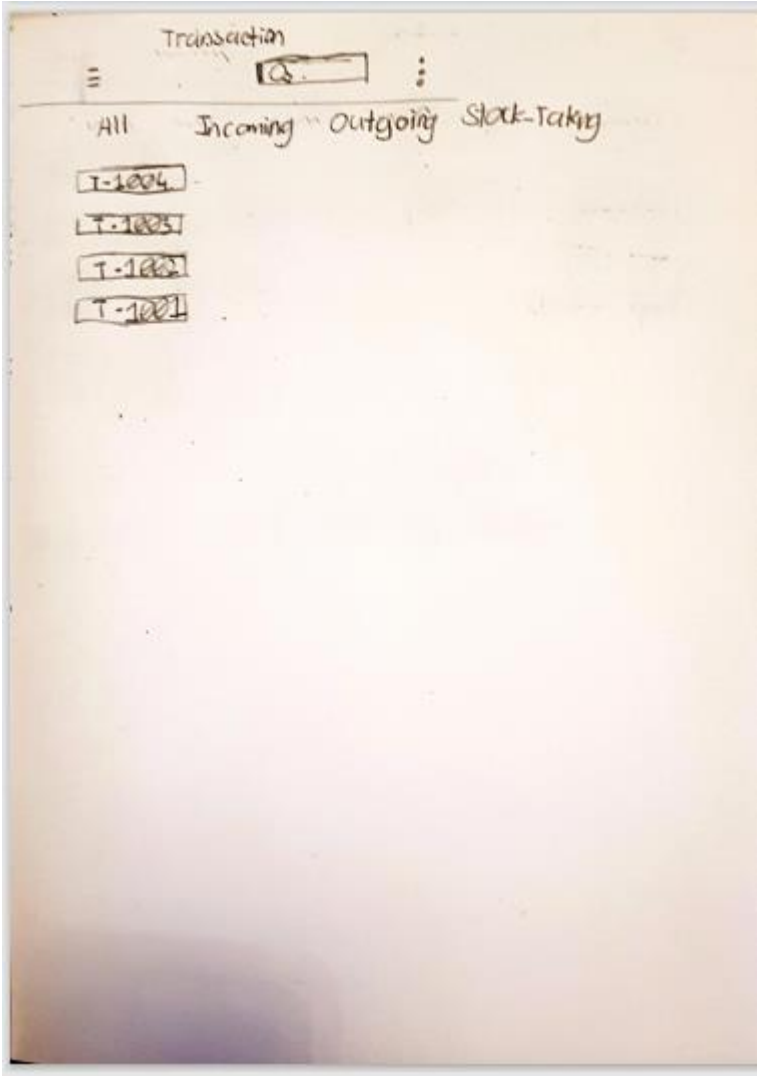
The below pages are the main backbone of the project. New pages and functionality could be added to the application.

Input Page



Items 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 a button radio group which consist of a 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 can be entered manually or automatically generated by pressing the button. 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 other notes fields can be replaced with other values if decided in the future.

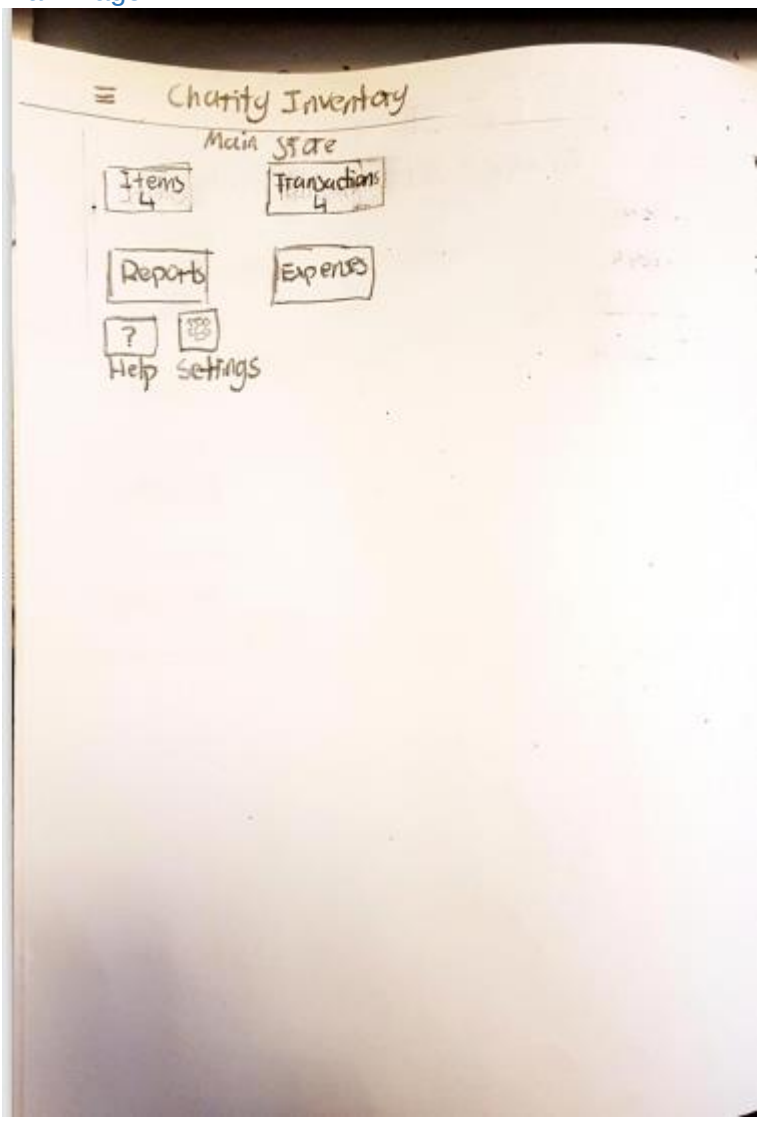
Invoice Page(Transaction Page)



This page will be used to keep track of the flow of items coming in and out of the charity shop. From the top left, there is a “three slash icon” to access the sidebar menu. Next to it is the search bar to search for any transaction using the transaction id or name. The “three-dot icon” will be another useful function that can be added in later. Below the top bar, there is a couple of heading buttons: “All”, “Incoming”, “Outgoing” and “Stock-Taking”. The “All” heading refers to all the transactions together. The “Incoming” heading refers to all items coming into the charity shop. The “Outgoing” heading refers to all the items that leave the charity shop whether sold or trashed. The “Stock-Taking” refers to keeping track of all the stock in the charity at that particular date.

A transaction will need a transaction id, name, description and date. 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

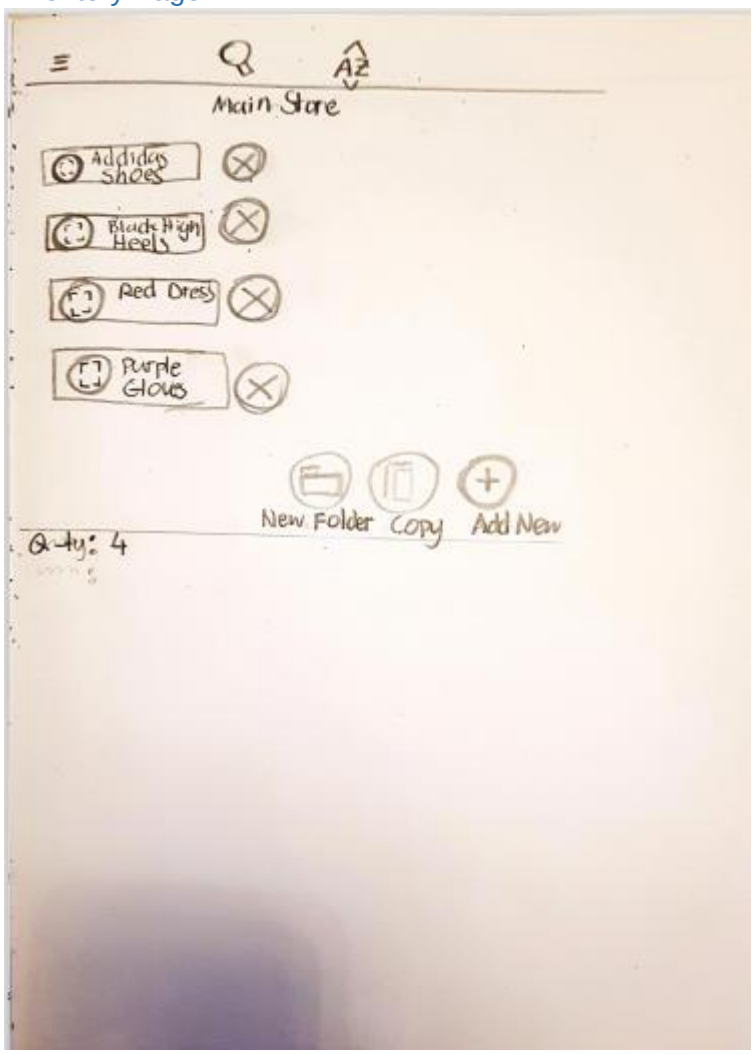


This page will be the main page once a user enters the application. There is a plan to add a splash page before, but it will need to be designed. The main page is basically a navigation page. It will also contain the same “three slash icon” that will access a side popup bar as said above on the previous page. The “Charity Inventory” title is a label that can be changed on the “Settings” page. Below it is another label “Main Store”, this can be changed in the future should there be a feature that allows access to multiple stores. The other buttons: “Items”, “Transaction”, “Reports, “Expenses”, and “Help” will also lead to their respective pages. The “Items” button will also have a quickly viewed “number label” i.e. in this case 4 for 4 items, which will keep track of how many items are in the charity shop inventory. The transaction button also has a similar label. Its label will keep track of the number of transactions done by the charity shop. In this scenario, it is 4 for 4 transactions.

The functionalities for the “Reports button” and “Expenses button” will be added in the future. However, they will lead to the “Reports Page” and “Expenses Pages” respectively. The idea is that the “Reports” page will give a summary of several activities e.g. “Item Receives” in a specific period e.g. a specific date. The “Expenses” Page will give all expenses up to date. There is a plan for the “Help Button” to lead to a “Help Page” which will contain several

articles within the application on how to solve various problems and tips on how to best use the application. Alternatively, this could also be changed to lead to an external page on the web browser which could contain the same content, or a knowledge base can be added. Additionally in the future, a “help chat” or contact number can be added to this website. This web page will only be active if the internet is connected, however.

Inventory Page

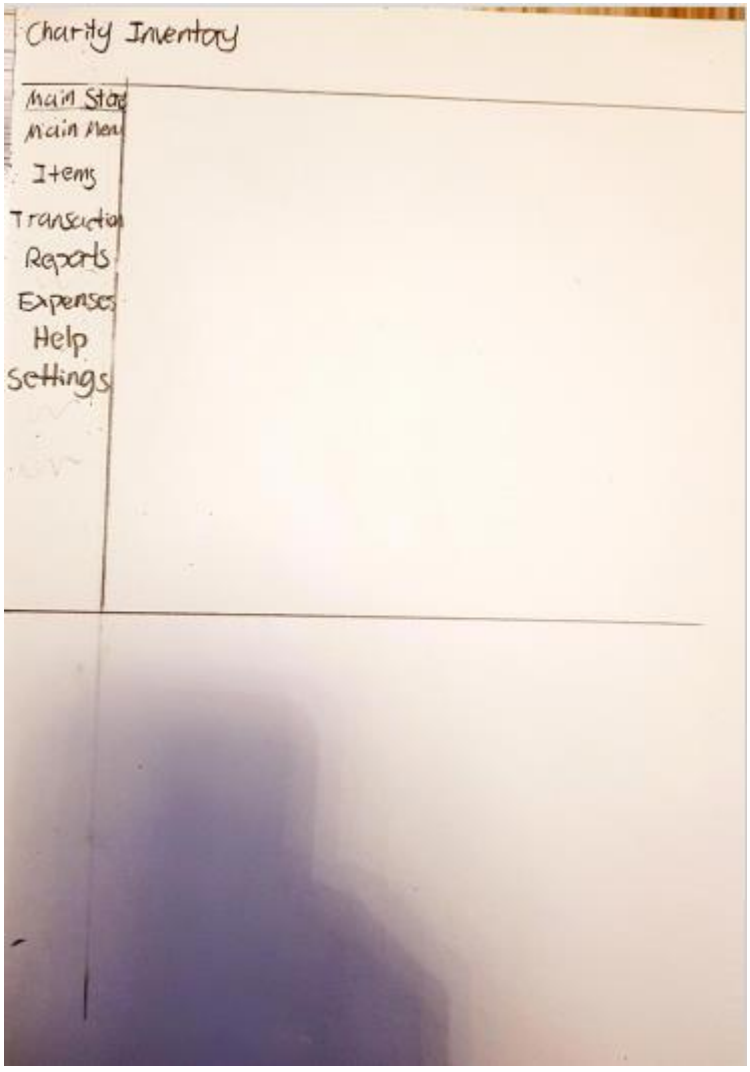


The inventory page is the page where the user can view the inventory in the charity shop. They can also seek more information about that item by clicking on the item records. There is also an opportunity for the user to edit and delete items. From the top left, as was spoken about earlier, is the icon to access the sidebar menu which will be navigation. 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. Below is a “Main store” label which can be modified as mentioned from before in previous pages. The icon records are stored in a list view. In this scenario, there are 4 records e.g., “Addidas Shoes”, “Black High Heel”, “Red Dress” and “Purple Gloves”. 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 images 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.

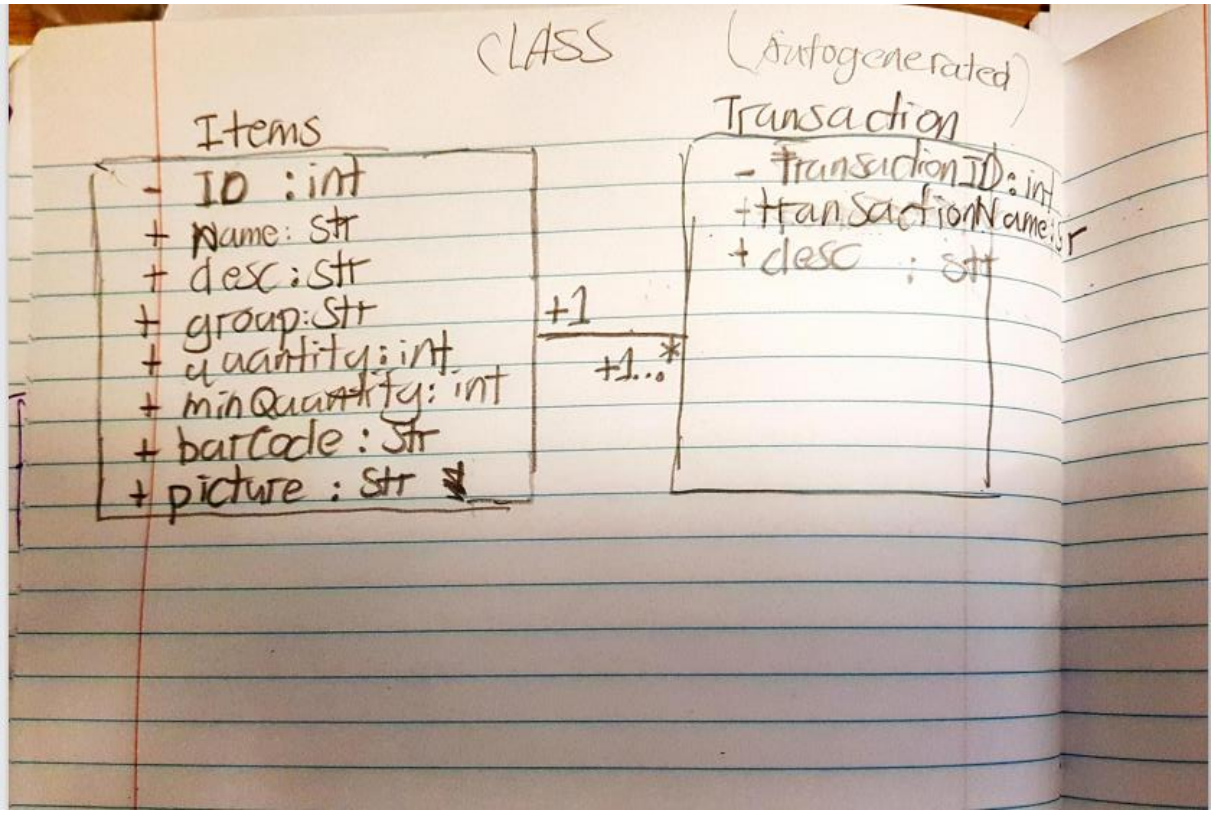
Sidebar Menu



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

Item Class

There are two classes here: Items and Transaction. There is a plan to autogenerate transaction instances using code.



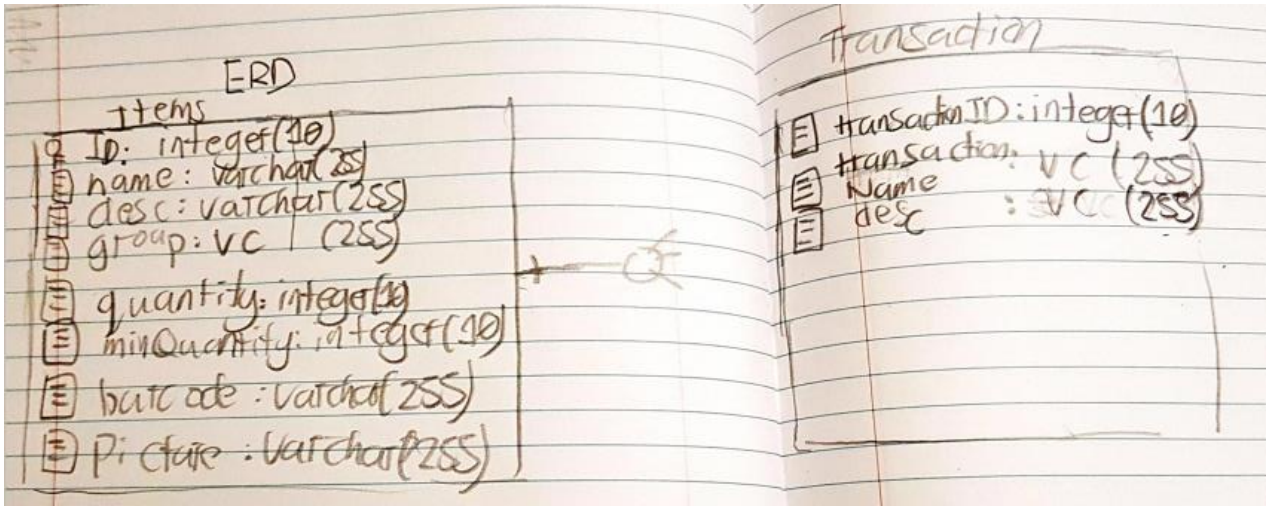
The item class will be used to hold information on items and the transaction class will be used to hold information on transactions.

Back end

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

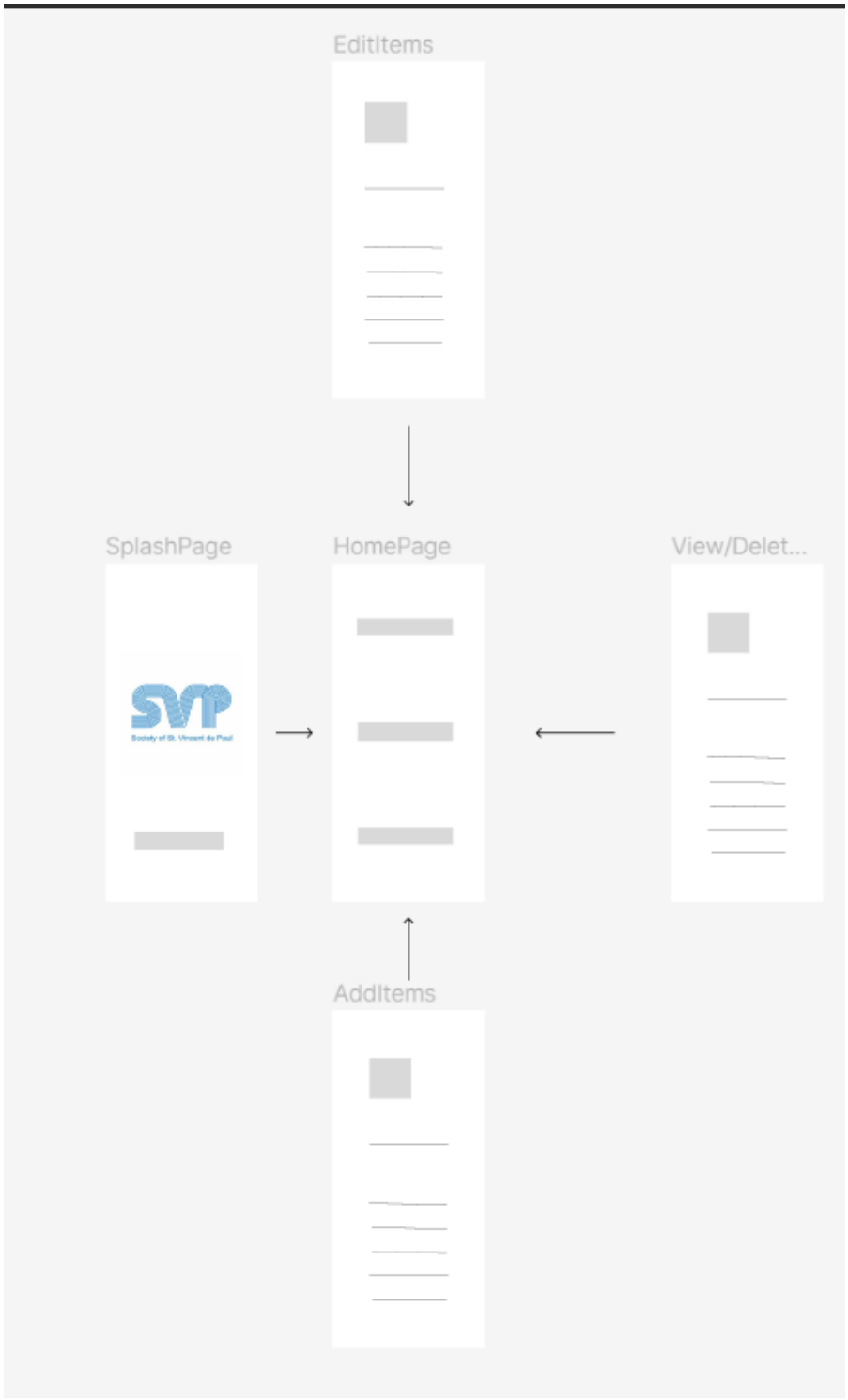
ERD

There are two entities here Items and Transaction.



Screen Flow Diagram

This diagram shows the flow of one screen to another screen. There is currently a centred design around the “Main” page. All pages currently lead to the “Main” page.


EditItems
SplashPage
HomePage
View/Delet...
AddItems

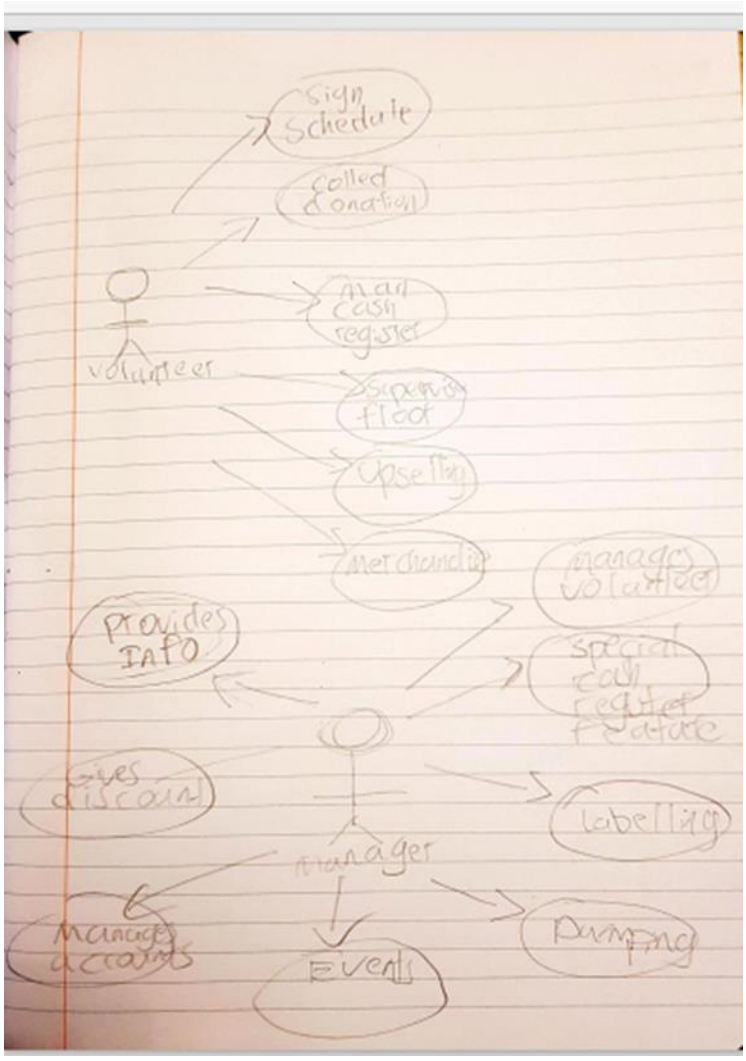
3.4. Use Case and Heuristics

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.

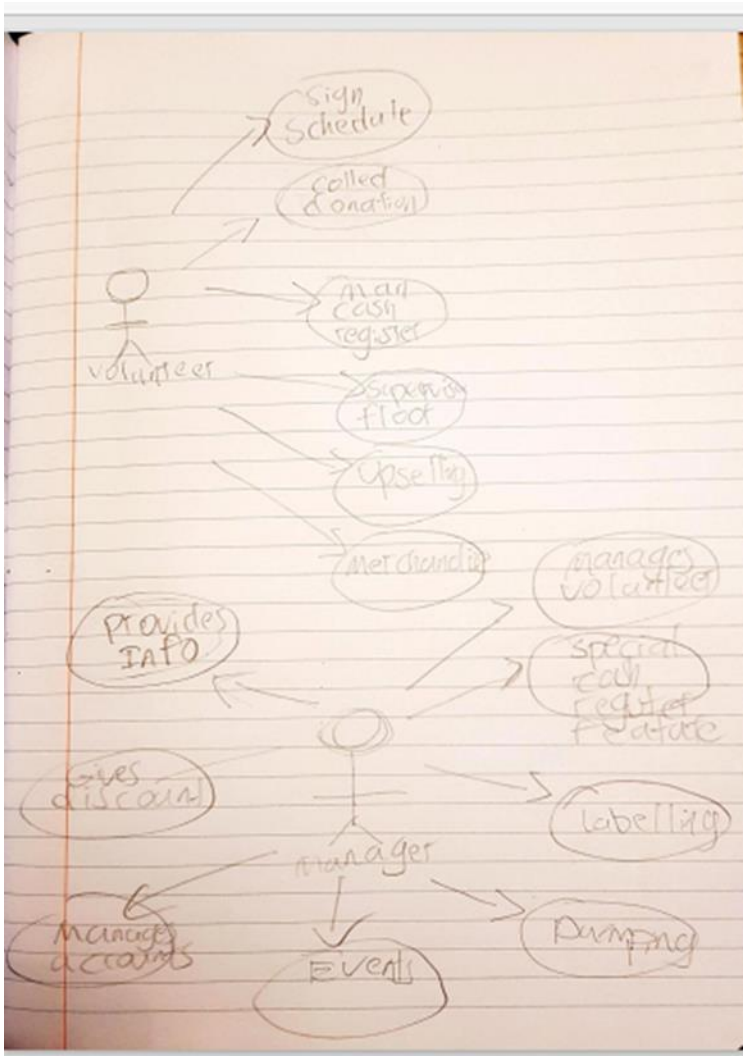
Volunteer and Manager



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.



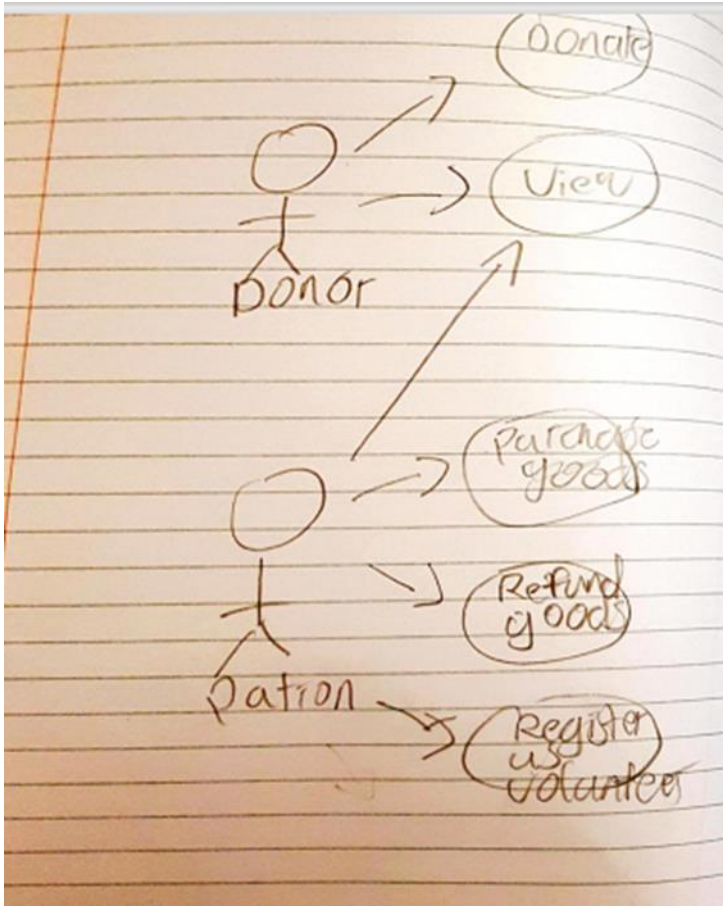
Manager

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.

These use cases below are of the donor and patron (of the charity shop).

Donor and Patron



Donor

As a Donor,

1. They need to be able to record donations to charity.
2. They need to be able to view information about the charity.

Patron

As a Patron,

1. They need to be able to purchase goods.
2. They need to be able to refund goods.
3. They may need to be able to register as a volunteer.

3.5. 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., Low 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. Testing and Evaluation

4.1. 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.

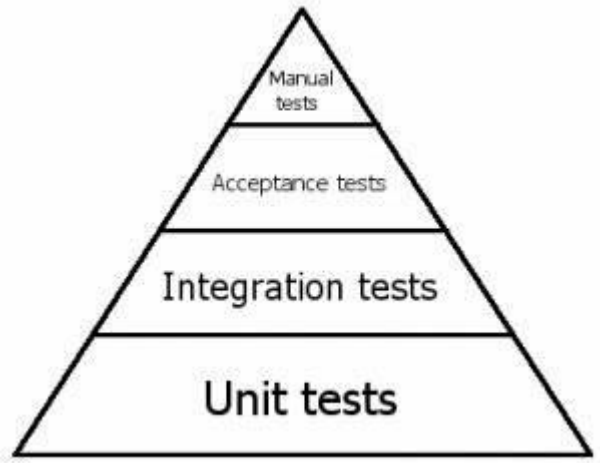
The second part is to test the software functionality that the code performs. This will entail testing the system functionality and enlisting some volunteers to try out the application. The feedback will help improve the functionality of the software. The functionality needs exactly what the user requirements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

4.2. Plan for 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.

4.3. Plan for 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.

4.4. 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 prove that the software and software design is possible. The prototype development section will complement the system design too as it will help determine certain parameters of that section e.g., whether the design is too complex or if it is possible at all.

5. Prototype Development

5.1. Introduction

This section is about creating a sample that will demonstrate that the software and software design is worth continuing, worth doing and possible. This is also known as testing the feasibility of the project.

5.2. Prototype Development

The following snippets illustrate the ability to prepare to Create, Read, Update and Delete (CRUD) operations using a front-end interface and a backend database.

Front end skills

Unit Test

```
 */
public class ExampleUnitTest {
    @Test
    public void addition_isCorrect() { assertEquals( expected: 4,  actual: 2 + 2); }
}
```

This shows a simple example of how to add a unit test. The system is testing an “actual” value which will be substituted for a variable from the code, with an “expected” value. This shows that Unit Testing is possible.

Using Libraries

```
4 usages
public class MainActivity extends Activity implements LocationListener {
```

This shows that it is possible to download and use different libraries in android studio

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

Some libraries will need to be defined in the dependencies section in the Gradle Build.

Animation

Slide In

```
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="-100%p"
    android:toXDelta="0%p"
    android:duration="1000">

</translate>
```

This will move into the screen view from the left side.

Slide Out

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="0%p"
    android:toXDelta="-100%p"
    android:duration="1000">

</translate>
```

This will move from the screen view back to the left side. This shows that animation is possible.

ListView

```
<ListView
    android:id="@android:id/list"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>
```

This above is a Listview which holds a collection of Linear layouts.

```
<LinearLayout
    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="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

LinearLayouts are a way to hold data which goes from top to bottom.

```
setListAdapter(new SimpleCursorAdapter(
        context: this,
        R.layout.row,
        cursor1,
        columns,
        rowIds
));
```

This will instantiate and populate the new ListView with data passed in as arguments. From these skills, any view type can be used.

Database Skills

Create

```
public long insertTask(String taskName, String taskDescription, int status)
{
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_TASKNAME, taskName);
    initialValues.put(KEY_TASKDESCRIPTION, taskDescription);
    initialValues.put(KEY_STATUS, status);
    return myDatabase.insert(DATABASE_TABLE,  nullColumnHack: null, initialValues);
}
```

This code above demonstrates that inserting into a table is possible. In this example taskName, taskDescription and status are values inserted into the database to form a new instance of entity “task”.

Read

```
public Cursor getAllTasks()
{
    return myDatabase.query(DATABASE_TABLE, new String[] {
                        KEY_ROWID,
                        KEY_TASKNAME,
                        KEY_TASKDESCRIPTION,
                        KEY_STATUS},
                selection: null,
                selectionArgs: null,
                groupBy: null,
                having: null,
                orderBy: null);
}
```

This code above demonstrates that reading all data from a table is possible.

```
public Cursor getTask(long rowId) throws SQLException
{
    Cursor mCursor =
            myDatabase.query( distinct: true, DATABASE_TABLE, new String[] {
                            KEY_ROWID,
                            KEY_TASKNAME,
                            KEY_TASKDESCRIPTION,
                            KEY_STATUS
                    },
                    selection: KEY_ROWID + "=" + rowId,
                    selectionArgs: null,
                    groupBy: null,
                    having: null,
                    orderBy: null,
                    limit: null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
```

This code above demonstrates that reading a specific data value from a table is possible. rowId is the value that is used to search. The cursor needs to move past lines should the record not be found in the current line.

Update

```
public boolean updateTask(long rowId, String taskName,
                          String taskDescription, int status)
{
    ContentValues args = new ContentValues();
    args.put(KEY_TASKNAME, taskName);
    args.put(KEY_TASKDESCRIPTION, taskDescription);
    args.put(KEY_STATUS, status);
    return myDatabase.update(DATABASE_TABLE, args,
            whereClause: KEY_ROWID + "=" + rowId,  whereArgs: null) > 0;
}
```

This code above demonstrates that it is possible to update data from a table. Uses rowId to find the correct rows and then updates it with new values from the function parameter.

Delete

```
public boolean deleteTask(long rowId)
{
    // delete statement.  If any rows deleted (i.e. >0), returns true
    return myDatabase.delete(DATABASE_TABLE,  whereClause: KEY_ROWID +
            "=" + rowId,  whereArgs: null) > 0;
}
```

This code above demonstrates that it is possible to delete data from a table. Deletes the record with the rowId value as an identifier.

Accessing the Firebase Database

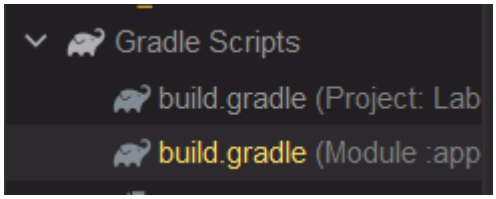
```
DatabaseReference driverRef = FirebaseDatabase.getInstance().getReference().child( pathString: "Users").
String customerId = FirebaseAuth.getInstance().getCurrentUser().getUid();
HashMap map = new HashMap();
map.put("customerRideId", customerId);
map.put("destination", destination);
map.put("destinationLat", destinationLatLng.latitude);
map.put("destinationLng", destinationLatLng.longitude);
driverRef.updateChildren(map);
```

This snippet above shows that it is possible to store data in variables. The variables can then be used to write to the Firebase Database using Java code.

Connect Frontend to Backend

```
DatabaseReference driverLocation = FirebaseDatabase.getInstance().getReference().child( pathString: "driversAvailable");
```

This shows that it is possible to connect a front end to a database in firebase. The library will need to be defined in the Gradle build.



5.3. Conclusions

The prototype here will be used to demonstrate that it is possible, feasible and within the ability of the student to create a software design which would be complete to some degree. In this case, it was proven that it is within the skills set of the student to manipulate data within a database e.g. Using CRUD methods.

This is a clear signal that is needed to begin developing the project. In the next section, the development of the project in code will be needed to be discussed.

6. Issues 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. 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., Cloud Firestore 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.

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.

Given the processes are defined, the code development will start.

	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

AgileAlliance (2017) ‘What is Scrum?’, *Agile Alliance* |, 7 April. Available at: https://www.agilealliance.org/glossary/scrum/ (Accessed: 19 January 2023).

Atlassian (no date) *What is DevOps?*, *Atlassian*. Available at: https://www.atlassian.com/devops (Accessed: 20 January 2023).

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).

Gaille, L. (no date) ‘15 Advantages and Disadvantages of a Waterfall Model’. Available at: https://vittana.org/15-advantages-and-disadvantages-of-a-waterfall-model (Accessed: 20 January 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).

Vasiliauskas, V. (2022) ‘14 Scrum Advantages and Disadvantages in 2022’, 8 September. Available at: https://teamhood.com/agile/scrum-advantages-disadvantages/ (Accessed: 19 January 2023).