**KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**KUMASI-GHANA**

**COLLEGE OF SCIENCE**

**DEPARTMENT OF COMPUTER SCIENCE**

****

**MINI PROJECT DOCUMENTATION**

**ON**

**STOCK KEEPING APPLICATION (STOCKPOT)**

**STUDENT’S NAME: ROBERT DOE**

**SUPERVISOR: DR. DOMINIC ASAMOAH**

**INDEX NUMBER: 1660717**

**CHAPTER ONE**

**INTRODUCTION**

* 1. **BACKGROUND TO STUDY**

Trading of goods and services has always been an everyday activity we undertake. With retailing being a significant form of trading, stock keeping is always needed. Retailers find it useful to take stock of goods sold and goods bought. Over the years this has always been done using a pen and a paper. As result, computations are subject to wrong calculations and loss of records for future references. Applications were created to help aid with stock keeping. Notwithstanding the fact that those application did much of the work, there are still few features they left untouched. The retailer receives no notification when he is running out of stock of a particular product. Available applications make no online backup of transactions. This in turn makes stock keeping complex tedious.

* 1. **PROBLEM STATEMENT**

Many retailers and shop keepers need to be prompted when they run of stock of particular products. In the case of data loss, retailers would love to get back information lost in order to continue transactions and properly make accounts. To aid with the issues listed, we propose a stock keeping android application. By using this software, retailers get to keep stock of goods, and have a local database and a remote database in the case of data loss. Retailers get notifications when they run of our stock. The retailer can monitor progress between dates and work toward bettering business strategies.

* 1. **OBJECTIVES**

The general objective of this project is to develop a Stock Keeping app that would help in prompting retailers when to do a refill and provide historical data of transactions to help with future strategizing.

This section focuses on how the general objectives of this project can be achieved. After building this project the retailer will be able to:

* Review current product information.
* Monitor profits and losses over a specified business period.
* Access backed up stock information.
  1. **JUSTIFICATION**

Over the years, retailers struggle with stock keeping. Due to misinformation, retailers are often unable to make quick refill due to late information of low stock. Retailers are unable to fully account for goods sold upon data loss. Alerting retailers when there’s a need for refill is very important. In the instance of data loss, it would be very much appreciated when there’s an online backup. This would reduce the stress retailers go through when refilling stock and accounting after a business week.

* 1. **RESEARCH QUESTIONS**
* How would retailers review product information of various stock.
* How would the retailer be able to monitor profits and losses over a specified business period?
* In the case of hardware damage or a phone loss, how would users have access to information saved locally on the phone?
* Would a user find it easier to navigate around the app to get things done?
* Would a user be able to monitor profits per time period?
  1. **LIMITATIONS OF STUDY**

In spite of the fruitfulness of this project, it leaves out a few concerns unaddressed. Below is a list of the observed limitations of this application.

* StockPot was created for use by only android platforms. Hence, users of iPhones cannot access this platform.
* In the case a user doesn’t have access to the internet prior to hardware damage, he ends up losing every data not backed up.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 REVIEW OF SIMILAR SYSTEMS**

Some similar existing apps are:

* Inventory Management
* CheckStockPro
* Rapid Inventory
* Storage Manager
* Stock and Inventory Management System

**2.2 SHORT COMINGS OF SOME EXISTING SYSTEMS**

* Most of these apps do not keep an online record of sales made hence making accounting efficiency of system breakdown difficult.
* Rapid Inventory is a sold app, hence most user would not be able to afford this service.
* In the bid to monetize these apps, they are flooded with ads with hinders better user experience.
* Complex navigation.

chapter 3

**SYSTEM REQUIREMENTS SPECIFICATIONS**

**3.1 REQUIREMENT SPECIFICATION REVIEW**

Requirement specifications is about all the functionalities needed for the design of the Stock Keeping Application. This is divided into two categories: the functional and nonfunctional requirements.

* 1. **FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS**
     1. **FUNCTIONAL REQUIREMENTS**

This outlines the intended behaviour of the Stock Keeping Application (StockPot).

These behaviours may be expressed as services, tasks or functions the system is expected

to perform. The Stock Keeping Application should be able to:

* Accept Retailer’s Information
* Input and Edit Product Details
* Modify Stock Quantities
* Record Transaction
* Notify Users of Low Stocks
* Monitor profits over given time periods
* Backup data generated
  + 1. **NON-FUNCTIONAL REQUIREMENTS**

These are requirements that are not primarily concerned with the functions performed by

the system. They relate to emergent system properties such as reliability, response time

etc. These requirements portray the general qualities of the system. Their malfunctions

affect the system as a whole are critical.

The non-functional requirements of the system are as follows:

**Usability**

The system is designed such that it can be used by Android systems that understand

its operation classes. User interfaces are user-friendly and easy to interact with: hence no extensive training is required.

**Compatibility**

The system was developed using Java, XML, Google Firebase, Room Database with Android studio. The system can therefore run on only Android platforms with a minimum SDK version being 15.

**Maintainability**

Third-party programmers who were not involved in the development of the system can

later add new features due to the availability of the accompanying documentation.

* 1. **USER AND SYSTEM REQUIREMENTS**
     1. **USER REQUIREMENTS**

These are requirements on the system user’s part, and they are as follows:

* + **Home screen**

Sorts retailer’s profit according to Year, Date, and Month

Displays existing inventory with varying colors distinguishing stocks base on their diminishing quantities.

{**red**=>finished**, lime**=>close to finishing, **green**=>enough}

* **Product screen**

Under the Add tab, retailers are able to add new products by providing:

Product name

Product description

Cost of Products per unit

Selling Price per unit

Threshold (Quantity of product necessitating a refill)

Under the View Tab of the Product Screen, retailer can update products or delete products by pressing on buttons on the respective card views of the recyclerview.

* + - * **Transaction Screen**

Under the Add Tab, retailers can record transactions made in the day by selecting the respective product and entering the required quantity purchased.

Under the Delete Tab, retailers can delete transactions. By so doing, the particular item which was bought gets an increase in its stock quantity.

* + - * **Stock Refill Screen**

Under the Add Tab, retailers can record refills made in the day by selecting the respective product and entering the required quantity refilled.

Under the Delete Tab, retailers can delete refills. By so doing, the particular item which was bought gets a decrease in its stock quantity.

* + - * **Count**

This screen shows the number of items in stock without any color bias.

* + - * **Back up screen**

The backup screen has a button which allows users to back up data accumulate in the database for future online. The platform aiding this service is know as Firebase. This would be discussed in later chapters.

* 1. **SYSTEM REQUIREMENTS**

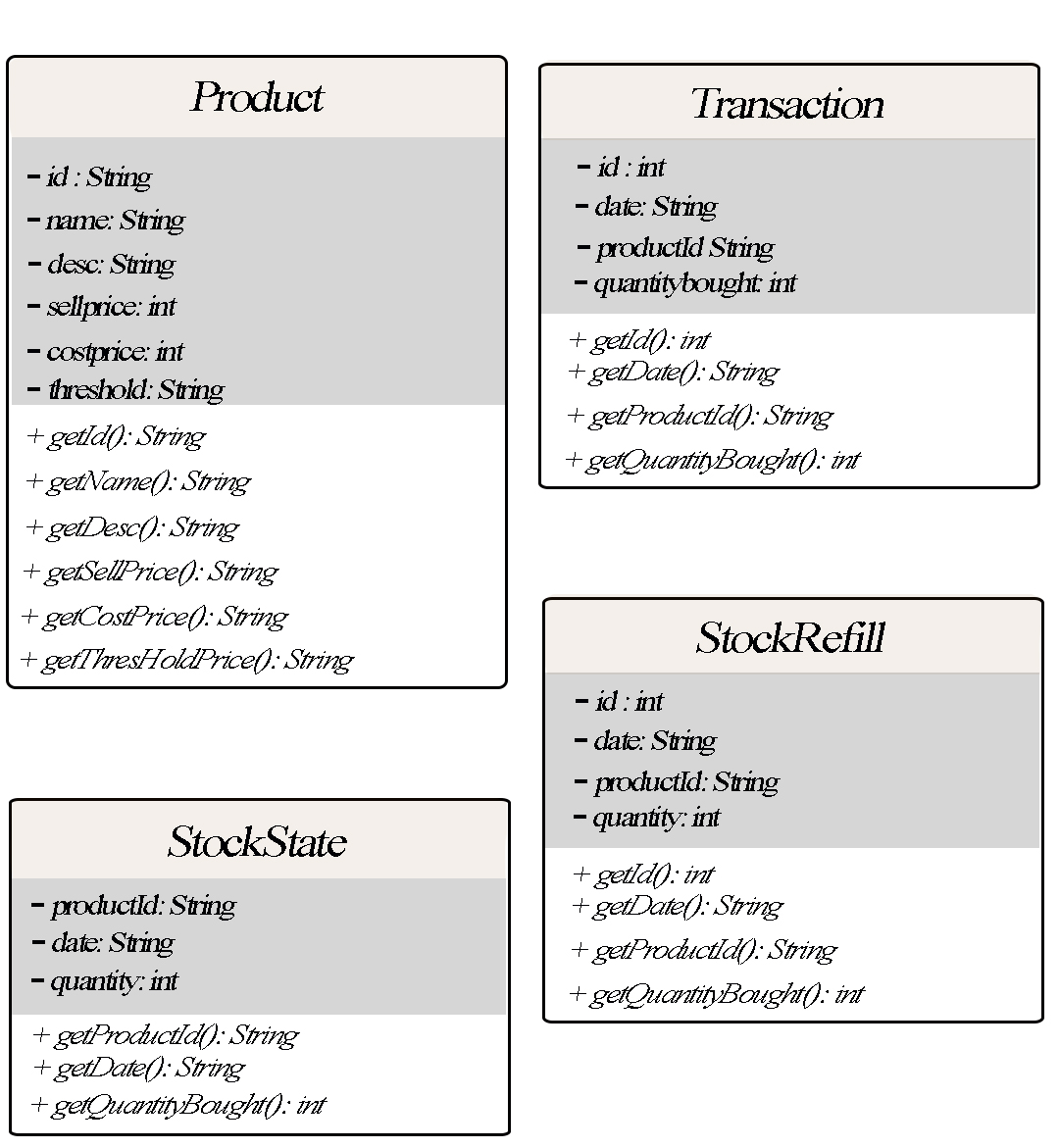
The proposed system requires an **Android phone** (partially with an internet connection) to operate.

* + 1. **UML DIAGRAM**
       1. **USE CASE DIAGRAM**

**System User**

**Stock Keeping Application**

* + - 1. **CLASS DIAGRAMS**

****

* 1. **THE WATERFALL MODEL**

The Waterfall Model was the first Process Model to be introduced. It is very simple to understand and use. In a Waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall model is the earliest Software Development Life Cycle approach that was used for software development.

In “The Waterfall” approach, the whole process of software development is divided into separate phases. The outcome of one phase acts as the input for the next phase sequentially. This means that any phase in the development process begins only if the previous phase is complete. The waterfall model is a sequential design process in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.

As the Waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a Linear-Sequential Life Cycle Model.

* + 1. **SEQUENTIAL PHASES IN WATERFALL MODEL**
* **Requirements:**The first phase involves understanding what needs to be designed and its function, purpose etc. Here, the specifications of the input and output or the final product are studied and marked.
* **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The software code to be written in the next stage is created now.
* **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
* **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. The software designed, needs to go through constant software testing to find out if there are any flaws. Testing is done so that the client does not face any problem during the installation of the software.
* **Deployment of System:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
* **Maintenance:** This step occurs after installation, and involves making modifications to the system or an individual component to alter attributes or improve performance. These modifications arise either due to change requests initiated by the customer, or defects uncovered during live use of the system. Client is provided with regular maintenance and support for the developed software.
  + 1. **ADVANTAGES OF THE WATERFALL MODEL**
* The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.
* The waterfall model progresses through easily understandable and explainable phases and thus it is easy to use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model, phases are processed and completed one at a time and they do not overlap. Waterfall model works well for smaller projects where requirements are very well understood.
  + 1. **DISADVANTAGES OF THE WATERFALL MODEL**
* It is difficult to estimate time and cost for each phase of the development process.
* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* Not a good model for complex and object-oriented projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.

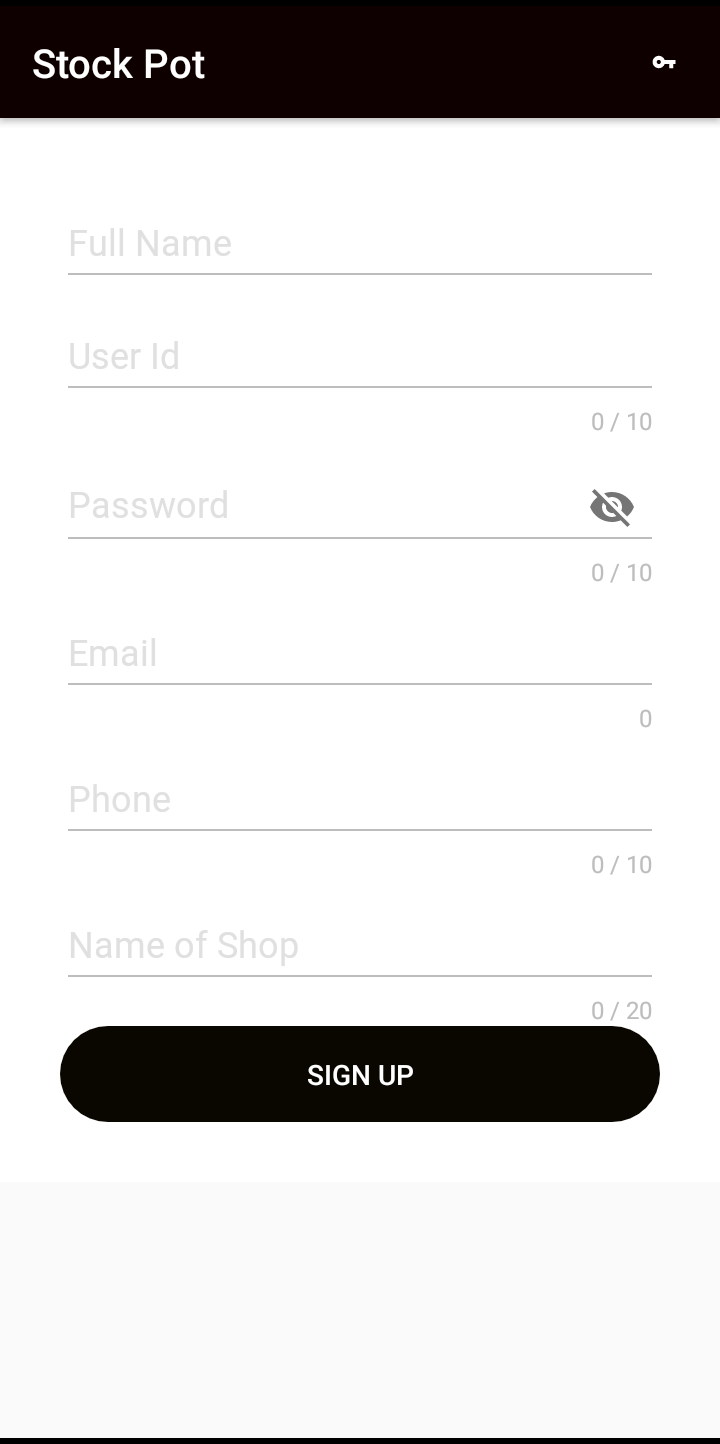
Following the waterfall model, this application was designed using the Android studio to design the user interface, make the system interactive, and user-friendly, and dynamic.

* + 1. **APPLICATION SCREENS**



**App Logo**

**Splash Screen**



Full Name EditText

User Id EditText

Password EditText

Email EditText

Phone EditText

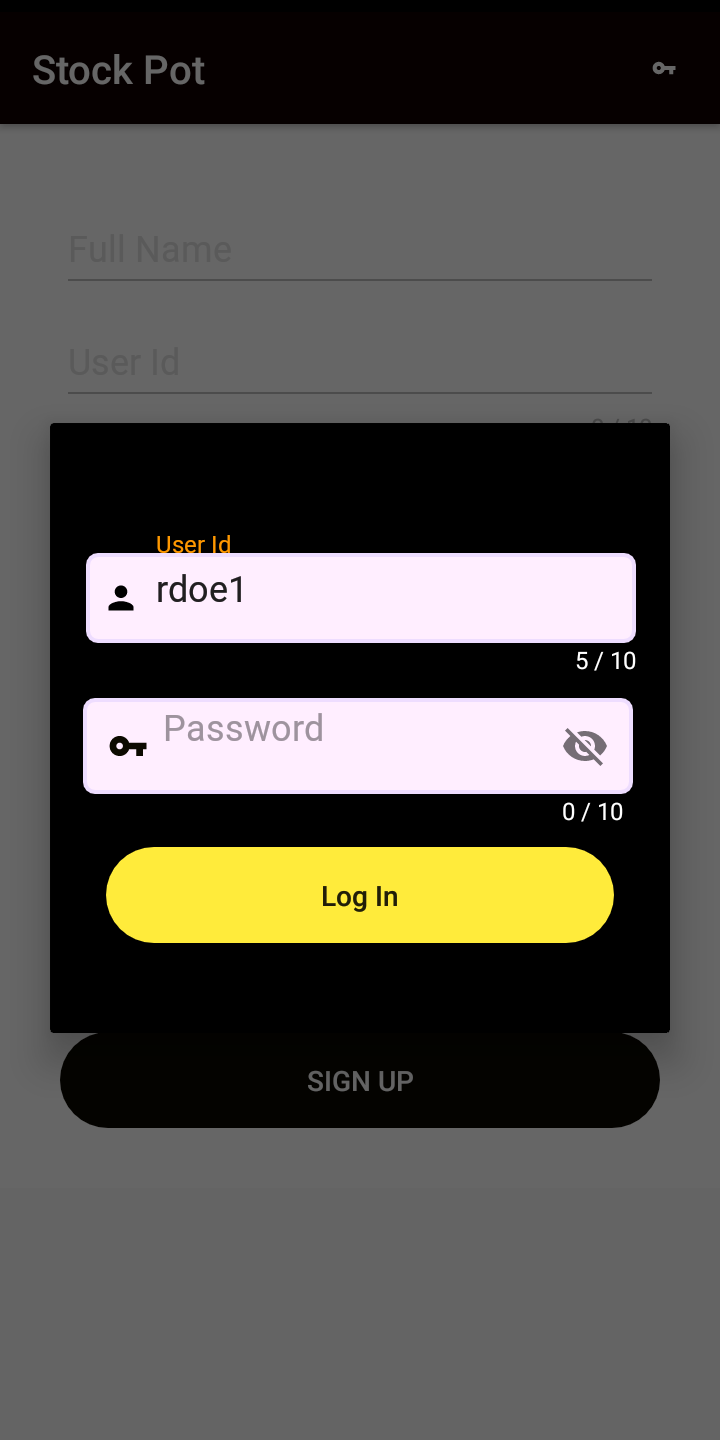
Name Shop EditText

Log in Button

**v**

**Sign Up Screen**

**Signup Screen**



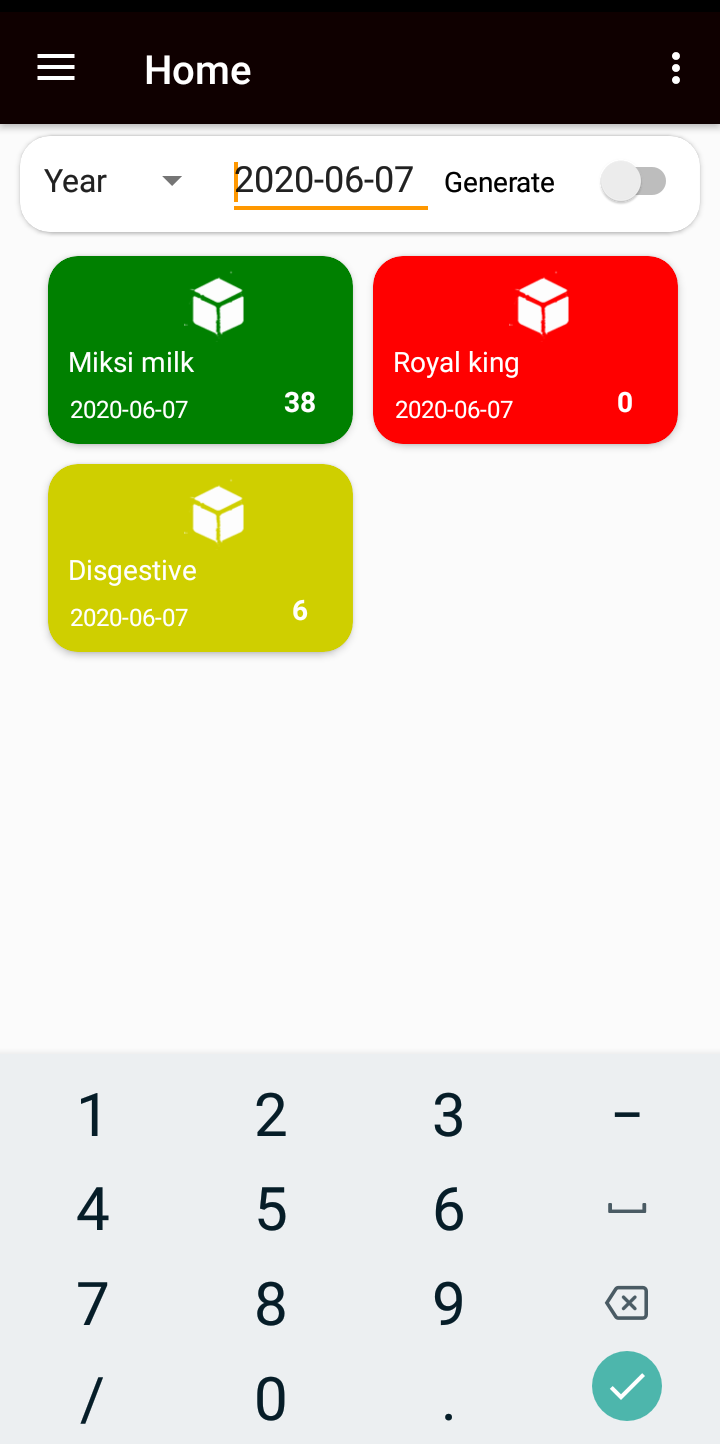
Login Dialog

User Id EditText

Password EditText

Login Button

**Login Screen**



Profit Generation Switch

Card Showing Product Below Threshold

(**Red**)

Sort Spinner

Card Showing Product Above Threshold

(**Green**)

Card Showing Product Equal Threshold

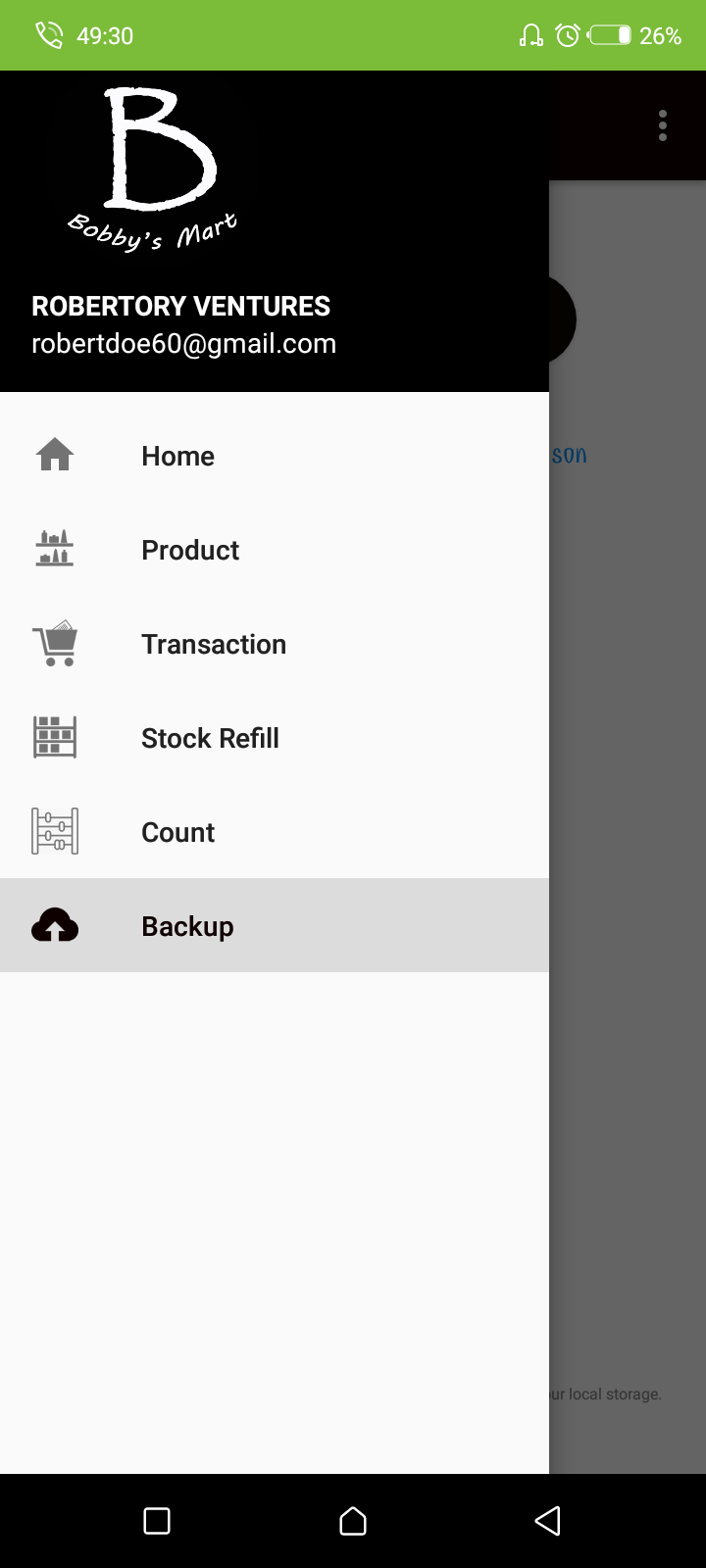
(**Lime**)



Card showing transaction and profits

Profit of 2 Generated on the transaction

**Home Screen Showing Generated Profit**



**Navigation Drawer to Various Screens**

Add Product Button

Threshold

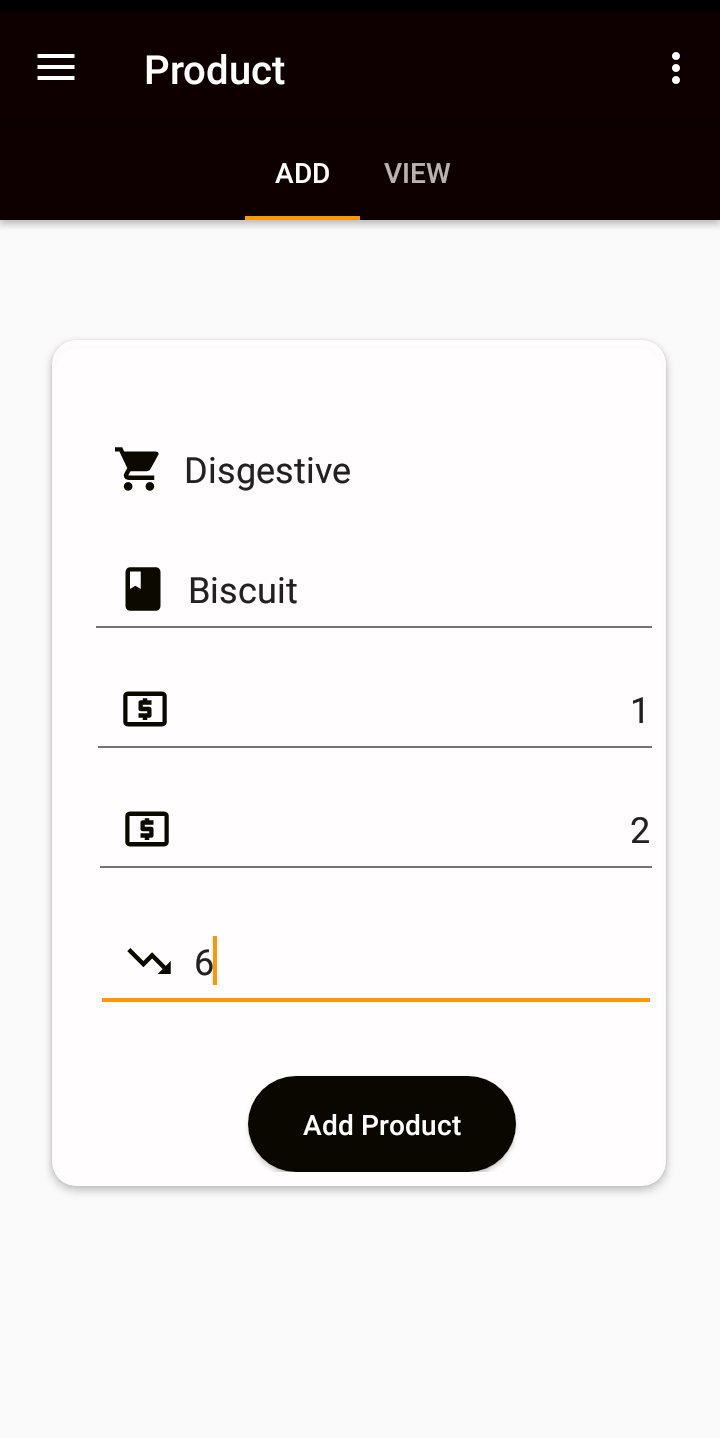
(Finishing Prompt)

Selling Price

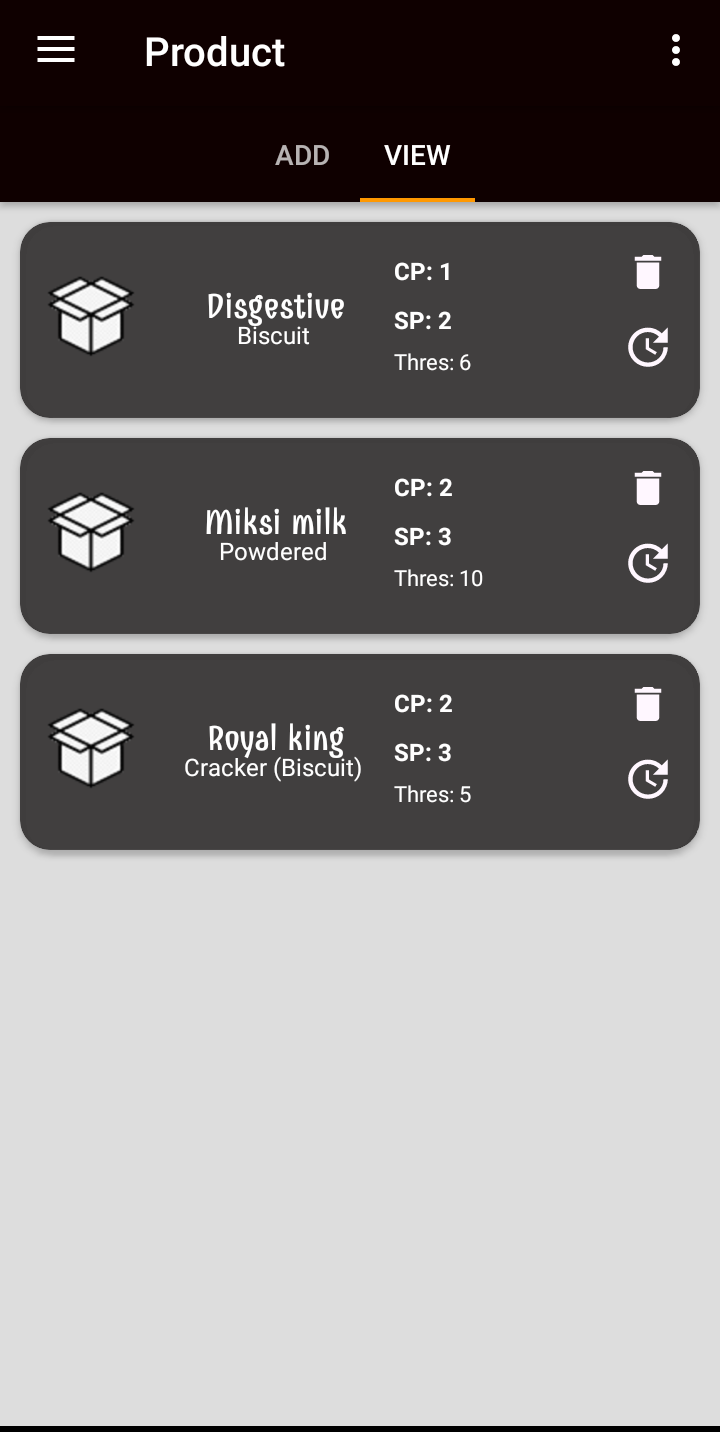
Cost Price

Product Description

Product Name



**Add Product Screen**

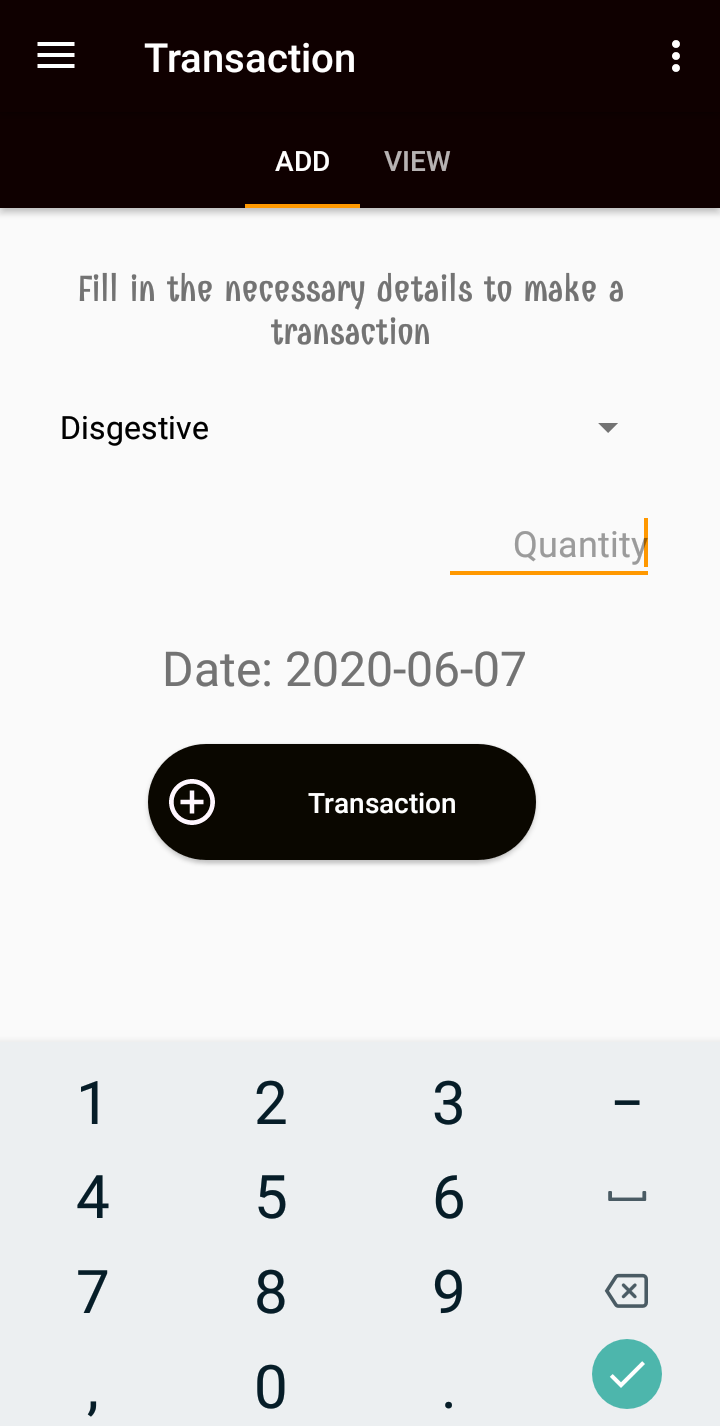
****

Delete Product Button

Update Product Button

Card showing Product Details

**View Product Screen**



Add Transaction Button

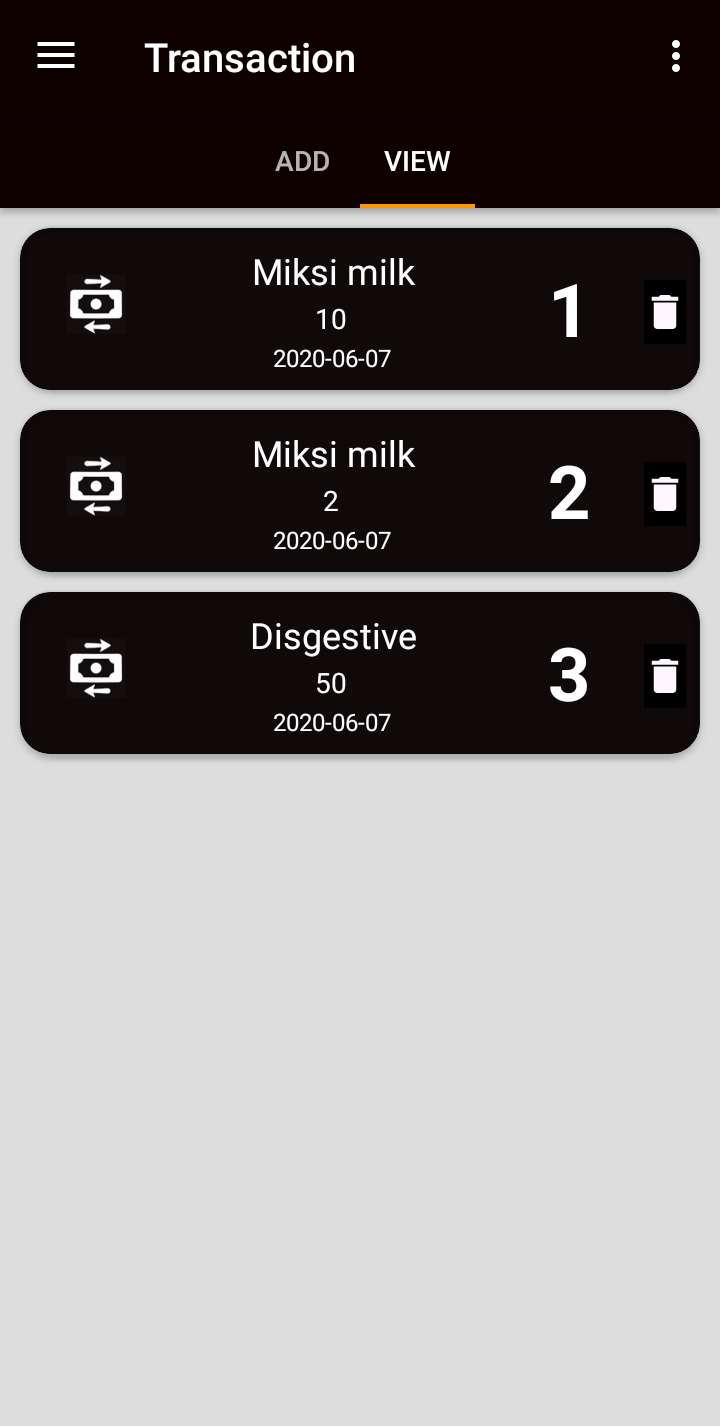
Date of Purchase

Quantity being purchased

List of Products

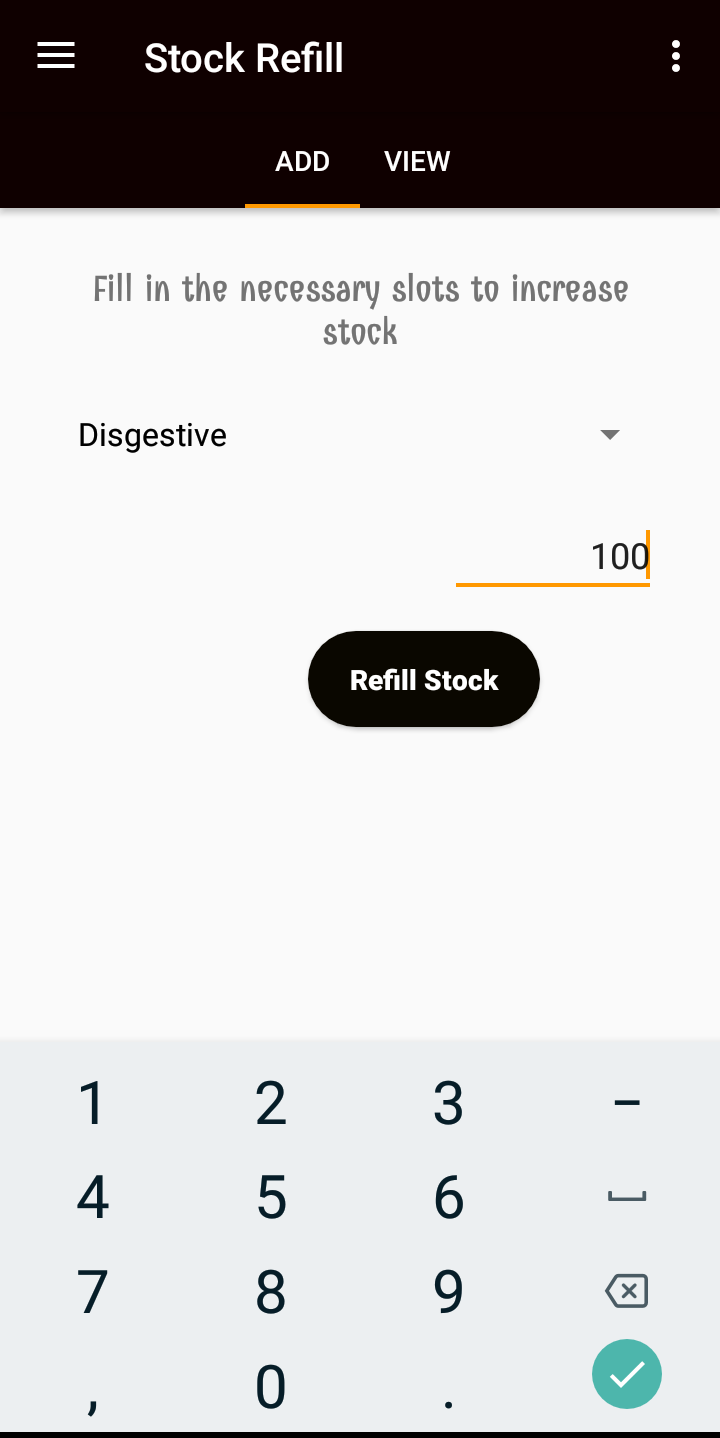
Tab Layout

**Record Transaction Screen**

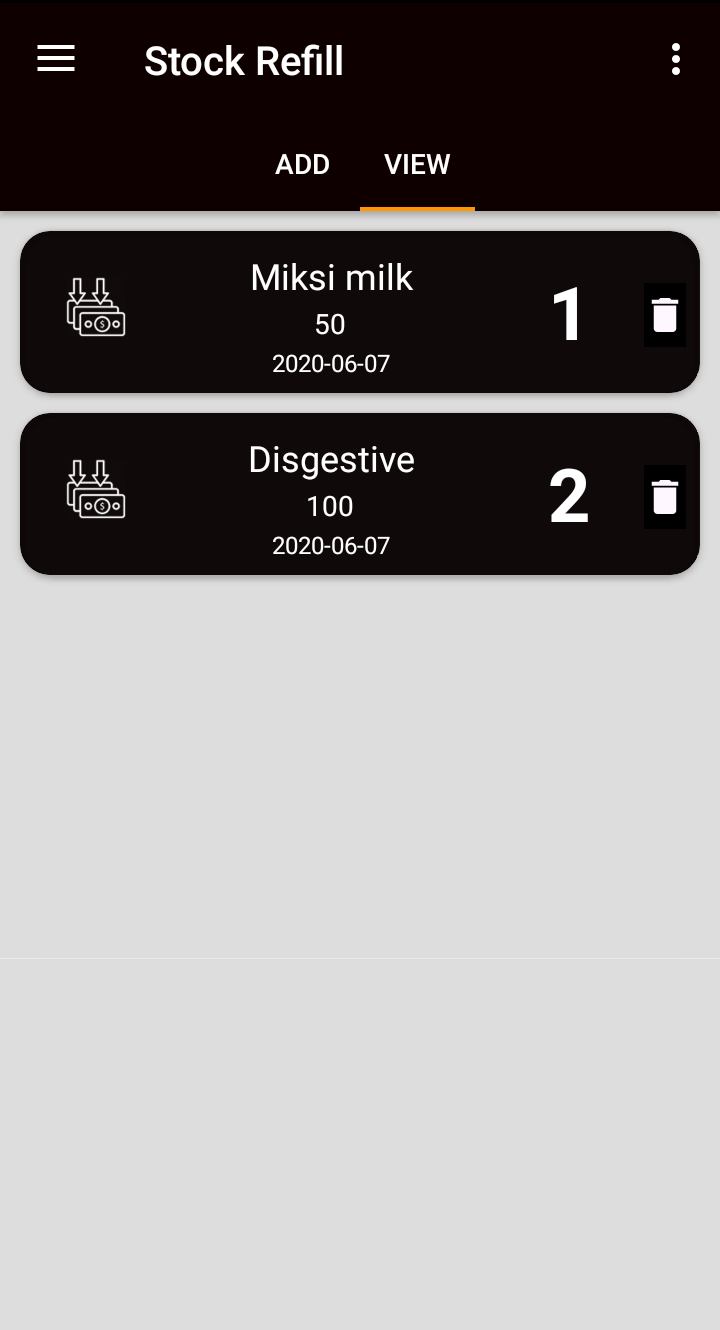
 **Add Stock Screen**

Transaction Id

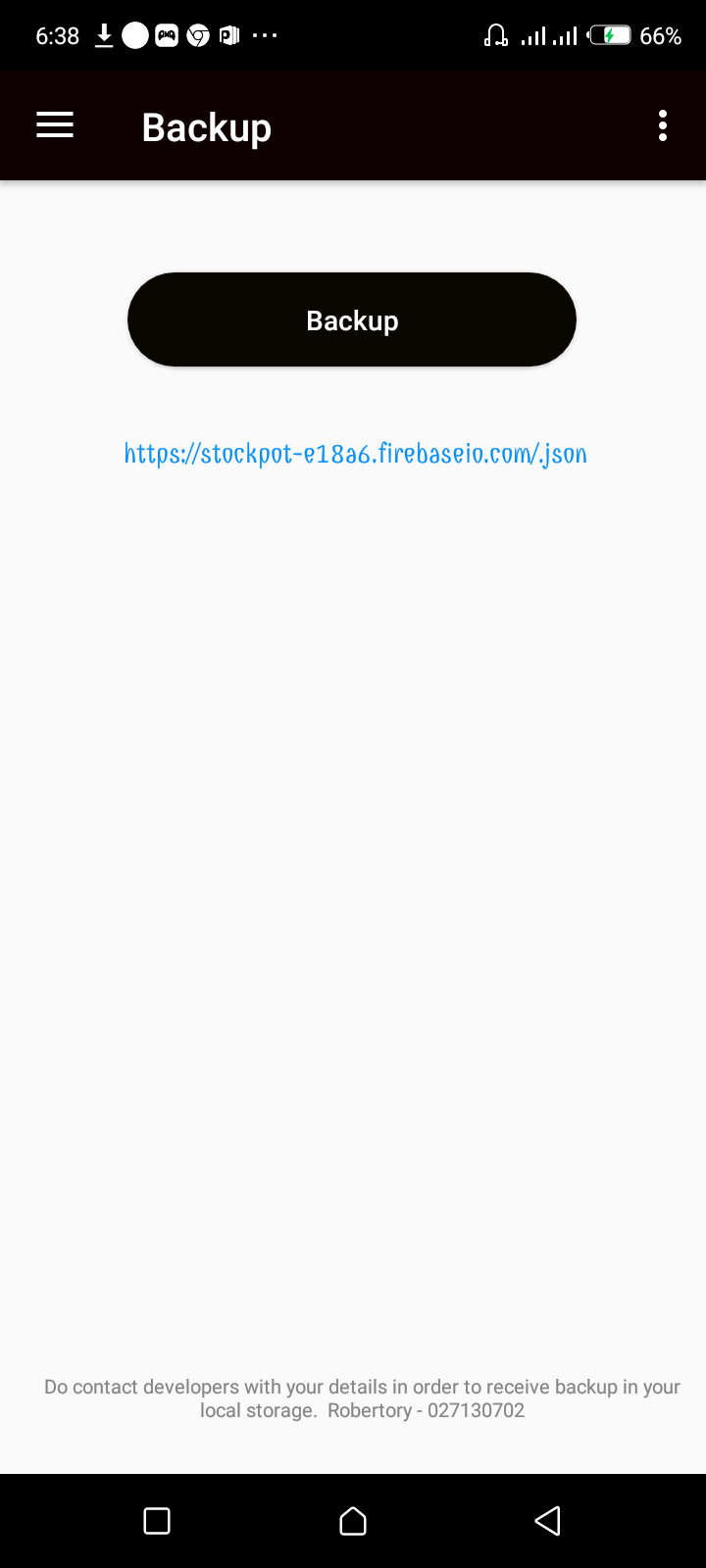
Delete Transaction Button



**View Refill Details**



**View Stock Refill**



* + 1. **DEVELOPMENT TOOLS AND FRAMEWORK**
* **Firebase**



Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure.

Firebase is categorized as a [NoSQL](https://www.educative.io/edpresso/whats-the-difference-betweensql-and-nosql) database program, which stores data in JSON-like documents.

In Firebase, a document is a set of key-value pairs defined by a schema. A group of documents makes up a collection.

**Key Features**

1. Authentication

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

2. Realtime database

Data is synced across all clients in real-time and remains available even when an app goes offline.

3. Hosting

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

4. Test lab

The application is tested on virtual and physical devices located in Google’s data centers.

5. Notifications

Notifications can be sent with firebase with no additional coding.

Users can get started with firebase for free; more details can be found on the [official website](https://firebase.google.com/).

* **Room Database**



Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite. Room is now considered as a better approach for data persistence than SQLite Database. It makes it easier to work with SQLite Database objects in your app, decreasing the amount of boilerplate code and verifying SQL queries at compile time.

# **Why use Room?**

* Compile-time verification of SQL queries. each @Query and @Entity is checked at the compile time, that preserves your app from crash issues at runtime and not only it checks the only syntax, but also missing tables.
* Boilerplate code
* Easily integrated with other Architecture components (like LiveData)

# **Major problems with SQLite usage are**

* There is no compile-time verification of raw SQL queries. For example, if you write a SQL query with a wrong column name that does not exist in real database then it will give exception during run time and you can not capture this issue during compile time.
* As your schema changes, you need to update the affected SQL queries manually. This process can be time-consuming and error-prone.
* You need to use lots of boilerplate code to convert between SQL queries and Java data objects (POJO).

# **Room vs SQLite**

Room is an ORM, Object Relational Mapping library. In other words, Room will map our database objects to Java objects. Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite.

**Difference between SQLite and Room persistence library:-**

* In the case of SQLite, There is no compile-time verification of raw SQLite queries. But in Room, there is SQL validation at compile time.
* You need to use lots of boilerplate code to convert between SQL queries and Java data objects. But, Room maps our database objects to Java Object without boilerplate code.
* As your schema changes, you need to update the affected SQL queries manually. Room solves this problem.
* Room is built to work with LiveData and RxJava for data observation, while SQLite does not.
* **Java**

****

**Java** is a set of [computer software](https://en.wikipedia.org/wiki/Computer_software) and specifications developed by [James Gosling](https://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems), which was later acquired by the [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation), that provides a system for developing [application software](https://en.wikipedia.org/wiki/Application_software) and deploying it in a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) computing environment. Java is used in a wide variety of [computing platforms](https://en.wikipedia.org/wiki/Computing_platform) from [embedded devices](https://en.wikipedia.org/wiki/Embedded_device) and [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone) to [enterprise servers](https://en.wikipedia.org/wiki/Enterprise_server) and [supercomputers](https://en.wikipedia.org/wiki/Supercomputer). [Java applets](https://en.wikipedia.org/wiki/Java_applet), which are less common than standalone Java applications, were commonly run in secure, [sandboxed](https://en.wikipedia.org/wiki/Sandbox_(computer_security)) environments to provide many features of native applications through being embedded in [HTML](https://en.wikipedia.org/wiki/HTML) pages.

Writing in the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)) is the primary way to produce code that will be deployed as [byte code](https://en.wikipedia.org/wiki/Java_byte_code) in a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM); byte code [compilers](https://en.wikipedia.org/wiki/Compiler) are also available for other languages, including [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), and [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)). In addition, several languages have been designed to run natively on the JVM, including [Clojure](https://en.wikipedia.org/wiki/Clojure), [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)), and [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)). [Java syntax](https://en.wikipedia.org/wiki/Java_syntax) borrows heavily from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B), but object-oriented features are modeled after [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk) and [Objective-C](https://en.wikipedia.org/wiki/Objective-C).[[13]](https://en.wikipedia.org/wiki/Java_(software_platform)#cite_note-13) Java eschews certain low-level constructs such as [pointers](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) and has a very simple memory model where objects are [allocated on the heap](https://en.wikipedia.org/wiki/Dynamic_memory_allocation) (while some implementations e.g. all currently supported by Oracle, may use [escape analysis](https://en.wikipedia.org/wiki/Escape_analysis) optimization to allocate on the [stack](https://en.wikipedia.org/wiki/Stack-based_memory_allocation) instead) and all variables of object types are [references](https://en.wikipedia.org/wiki/Reference_(computer_science)). Memory management is handled through integrated automatic [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) performed by the JVM.

* **XML**

Extensible Markup Language (XML) allows you to describe and organize information in ways that are easily understandable by both humans and computers. You can then share that information and its description with others over the Internet, an extranet, network, or in other ways.

XML, like Standard Generalized Markup Language (SGML), is a metalanguage. A metalanguage allows you to define a document markup language and its structure. For example, both XML and Hypertext Markup Language (HTML) are derived from SGML.

You can use XML to create your own markup language that includes a set of rules and tags that describe information suited to your needs, for example, name, title, address, and zip code. You define this markup language in a document type definition (DTD) or XML Schema file that functions as the standard way to describe your information. Using XML to share standardized information means you are no longer required to write programs to focus on proprietary software or convert and translate different data formats.

You and others can use the DTD or XML Schema definition to tag information that you can then use in a variety of ways: printed on an address label, business card, or stationary; displayed in a Web page; or sorted in a list of data with similar attributes. For example, you might want to create an efficient way to share information (such as purchase orders, shipping acknowledgments, order status, and stock status) with your partners and suppliers. You can use XML to share that information by creating and using XML documents that conform to your DTD or XML Schema, in which you specify the standard for the electronic exchange of information.

Although both XML and HTML use tags to describe content, they are also very different:

HTML describes how to format information for display and is meant for computer-to-human interaction.

XML describes what the information is and is meant for computer-to-computer interaction.

There are many other sources of information regarding the advantages and uses of XML that are readily available on the Internet and through commercial publications. For a complete and comprehensive understanding of the capabilities of XML, you should consult these sources.

**Advantages of XML include the following:**

* XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML.
* XML is completely compatible with Java™ and 100% portable. Any application that can process XML can use your information, regardless of platform.
* XML is extendable. Create your own tags, or use tags created by others, that use the natural language of your domain, that have the attributes you need, and that makes sense to you and your users.

**DESIGN PATTERNS EMPLOYED**

* Composite Design Patter
* Adapter Design Pattern
* Command Design Pattern
* Half Async Design Pattern
* Singleton Design Pattern
* Factory Design Pattern
* Builder Design Pattern

**CHAPTER 4**

**TESTING AND EVALUATION**

**4**.**1 FEATURES TO BE TESTED/NOT TO BE TESTED**

**4.1.1 Features to be tested**

Below table lists the features that will be tested during the current test or the subsequent planned tests

|  |  |
| --- | --- |
| **Features to be tested** | **Test Description** |
| **Login to the system** | This tests the login interface of the system. |
| **Adding a Product to database** | This test is conducted to verify if a product is successfully added to the database. This will check if the product object is added to product table and also check if the product details are under respective Ids. |
| **Adding a Transaction to database** | This test checks if new Transaction is added correctly to the database with the specified details. |
| **Adding a Refill to the database** | This test checks if the new Product Refill is correctly added to the database with the specified details. |
| **Checking the threshold levels** | This test is conducted to verify if the Product that are below the threshold levels are listed by the function when called by the retailer. The verification is done by referring to the database. |
| **Updating the sales for the day** | This test is conducted to test the sales update in the database. The test checks if the database is updated with the correct stock values based on the sales data input to the system. |
| **Updating the stock quantity to database** | This test is conducted to test the correct updating of the database after making a transaction from the retailer. |
| **Create Transaction orders** | This test is conducted to check the transaction creation capability of the system. The list of products that is generated for order must comply with the set conditions of threshold levels |
| **Process Orders** | This test is conducted to test if the created orders are processed correctly into a file. |
| **Updating a Product** | This test checks for the correct updating of the selected product. |
| **Deleting a Product** | This test checks for the deletion of the selected product by the retailer. Also, it has to check it corresponding entries in the related tables are deleted correctly. |
| **Manager Interface** | This test will determine if the user is able to navigate through the interface and if the user can access the all the functions of the system navigating via the main manager interface. |
| **Backup Data** | This test data in tables are not null and hence backup for online storage. |

**CHAPTER 5**

**RECOMMENDATION AND CONCLUSION**

The system propose has been passed through a series of systematic tests. Each proposed functionality has been tested to ensure that the application is free from errors and is delivered on its promise. Opinions and recommendation from the supervisor, course mates as well as other tutors of exploring new technology were taken into consideration, which helped to rectify and improve on many aspects of this application.

**CONCLUSION**

The major success of implementing this project and upgrading it to meet more user specifications, will really help retailers improve their sales.

**REFERENCES**

<https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_72/rzamj/rzamjintroadvantages.htm>

<https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_72/rzamj/rzamjintro.htm>

<https://en.wikipedia.org/wiki/Java_(programming_language)>

<https://medium.com/mindorks/using-room-database-android-jetpack-675a89a0e942>

<https://www.educative.io/edpresso/what-is-firebase>

<https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm#:~:text=The%20Waterfall%20Model%20was%20the,no%20overlapping%20in%20the%20phases.>

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/>

<https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/>