

**A MOBILE SMARTINVENTORY – APPLICATION FOR SMALL BUSINESS STALLS**

**DOCUMENTATION**

**BY**

**Karigitho Edwin Joshua K. – ID: 659898**

**A Mid-Semester Project Submitted To School of Science and Technology in Partial Fulfillment of the Requirements for Bachelor of Science in Applied Computer Technology.**

**(SUMMER SEMESTER 2022)**

**UNITED STATES INTERNATIONAL UNIVERSITY-AFRICA**

# **DECLARATION AND APPROVAL**

I, the undersigned, declare that this is my original work and has not been submitted to any other

college, institution, or university other than the United States International University-Africa for academic credit.

Signed:……………………………………………………Date:…………………………………...

**(Karigitho Edwin Joshua 659898)**

This project has been presented for examination with my approval as the appointed supervisor.

Signed:…………………………………………………Date:…………………………………

**(Supervisor’s Name)**

# **COPYRIGHT**

All rights reserved. No part of this dissertation report may be photocopied, recorded, or otherwise

reproduced, stored in a retrieval system, or transmitted in any electronic or mechanical means

without prior permission of USIU-A or the author, (Karigitho Edwin Joshua Kiuma © 2022).

# **ABSTRACT**

*It is a nightmare for a business to come to a realization that an item has been double sold. This is an attribution to lack of proper management of inventories for businesses operating both online and physical.*

*This paper introduces a new mobile application that helps in managing product inventory and producing a monthly report. The mobile inventory application is suitable for small businesses and not large business and companies due to its focus on simplicity. It focuses on additional visuals to the inventory management system making it more modern. Through waterfall methodology, the application was created in a short period while testing the different components. The application was developed in a span of one month with cronoligal features of the application attached in the document.*

**TABLE OF** **CONTENTS**

[**DECLARATION AND APPROVAL** 2](#_Toc110340186)

[**COPYRIGHT** 3](#_Toc110340187)

[**ABSTRACT** 4](#_Toc110340188)

[**CHAPTER 1: INTRODUCTION** 9](#_Toc110340189)

[**Background of the System** 9](#_Toc110340190)

[**Problem Statement** 10](#_Toc110340191)

[**The business stalls now using is a manual system** 10](#_Toc110340192)

[**Time Consuming** 10](#_Toc110340193)

[**Objective** 11](#_Toc110340194)

[**Specific** 11](#_Toc110340195)

[**Preliminary** 11](#_Toc110340196)

[**Assumptions** 11](#_Toc110340197)

[**Scope** 11](#_Toc110340198)

[**Limitation** 12](#_Toc110340199)

[**CHAPTER 3: INTENDED APPROACH/ DEVELOPMENT METHODOLOGY** 13](#_Toc110340200)

[**Methodology** 13](#_Toc110340201)

[**Functional and Non-Functional Requirements** 14](#_Toc110340202)

[**Design tools** 15](#_Toc110340203)

[**Development Tools** 16](#_Toc110340204)

[**Database development tools** 16](#_Toc110340205)

[**Programming Tools** 16](#_Toc110340206)

[**CHAPTER 4: SYSTEM ANALYSIS AND DESIGN** 17](#_Toc110340207)

[**Use case diagrams** 17](#_Toc110340208)

[**Database Schema** 18](#_Toc110340209)

[**Class diagram** 19](#_Toc110340210)

[**Graphical User Interface** 20](#_Toc110340211)

[**CHAPTER 5: CONCLUSION** 28](#_Toc110340212)

[**Future Development** 28](#_Toc110340213)

[**REFERENCES** 29](#_Toc110340214)

[**APPENDIX** 30](#_Toc110340215)

[**Sample budget** 30](#_Toc110340216)

[**Project schedule** 31](#_Toc110340217)

[**Sample Codes** 32](#_Toc110340218)

**List of Tables**

**Table 1:1 (Budget table) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_19**

**Table 1:2 (Schedule table) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_30**

**Table 1:3 (Gantt chart table) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_32**

**List of Figures**

**Fig1:1 (Waterfall diagram) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 14**

**Fig1:1 (Use case diagram) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 17**

**Fig1:2 (Database schema diagram) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 18**

**Fig1:3 (Class diagram) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 19**

**Fig1:4-Fig2:1 (Graphical User diagram) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20**

**Fig1:4 (Scan page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20**

**Fig1:5 (Add items) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20**

**Fig1:6 (Delete page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 21**

**Fig1:7 (View Report) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22**

**Fig1:8 (Search Item page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 23**

**Fig1:9 (Dashboard page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 24**

**Fig1:10 (Splash page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 25**

**Fig2:0 (Login page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 26**

**Fig2:1 (Register page) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 27**

**Fig2:2 (Gantt chart) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 32**

**Abbreviation and Terms of Reference**

**SmartInventory**: Name of the system

**App**: stands for Application

**IDE:**

**GUI:** Graphical User Interface

# 

# **CHAPTER 1: INTRODUCTION**

## **Background of the System**

SMARTINVENTORY is a mobile application designed for small stores that allow users to key in their products corresponding to the amount for easy tracking of the business products. It's a web-based application.

Quality, production times, distribution of times, and final prices will brand you as a successful company, using some key management components can get you to that level. This is what the SmartInventory app is all about, keeping financial and product records for future reference. Each of these is very crucial in materials management and whether the company is going to flourish or fall. The Smartinventory app helps business people to make their inventory work more easy and fast to avoid confusion and losses.

Nowadays, in an era that has advanced technology and a place in the world. Everything can be linked only at your fingertips in the times of rapidly developing sophisticated technology today. Therefore, an inventory system is also not lagging in introducing a method of keeping inventory data systematically and safely. The system plays a very important role in improving the competitiveness of a business. Usually, organizations today face too many challenges to achieve cost, speed, and reliability. An efficient inventory system helps to make sure the store's performance and data record is always in good condition and secured from abusers.

Our target audience is primarily small businesses, and retail or wholesale stores, with a huge amount of products. Built for online sellers, drop-shippers, retailers, distributors, and businesses that deal with products daily basis.

The problem with the recent inventory applications is that users can't scan using their camera or view their history on the products. With the new Mobile App, you can scan the codes for easy intake of the products and also view the history for easy tracking of the products.

## **Problem Statement**

Most small businesses do not have a written report of both or service sales however, they can forecast sales based on historical information and market trends. The problem faced by the businesses is they do not have any systematic system to record and keep their inventory data. It is difficult for the administrator to record the inventory data quickly and safely because they only keep it in logbooks that are not properly organized. This attributed to lack of proper knowledge or simplified technological advancement to aid in proper record keeping. The E-mobile application focuses on enhancing the inventory process through a simplified and user friendly experience.

### **The business stalls now using is a manual system**

The business stalls’ problem is they using a chaos system and it is difficult for the admin to estimate their profit. With the new system developed, the company can manage its inventory data easily, quickly, and more securely. Some of the systems in the Kenyan stall that they use are a Spreadsheet, Microsoft word, or file systems. This even makes it tiresome and challenging whenever you want to search for something and keep track.

### **Time Consuming**

Recording the inventory data will cost time. Admin of the company only one person so he needs to record every stock detail clearly or else it may lead to a lack of information about the data.

## **Objective**

### **Specific**

To develop a mobile application that helps a businessperson to keep inventory records and generate a report.

### **Preliminary**

1. To be able to store
2. To be able to view, delete, update or add products
3. To enable the user to login/signup
4. To be able to make calculations automatically
5. To create a report
6. To be able to tell the user what’s running out
7. To be able to scan the back-codes.

## **Assumptions**

Today, people use the traditional way of recording their business products, either through a notebook, excel sheet and other writing manual materials. The most parts they keep in their records are the product name, the number of products, price and the date the product left the shop. Sometimes data or the files are lost or someone might change the hardware.   
By the help of this application, the user will be able to store data on to the cloud and access the reports anytime and anywhere as long as they have the account details.

## **Scope**

The application shall only cater to warehouses, retailers, and other small business shops but not large enterprises level solution businesses. The programming IDE being used is Android Studio using Java as the programming language. Therefore, the mobile application shall only be able to run on virtual android GUI’s and android mobile devices.

## **Limitation**

The limitations to this application encompass both the user and developer. They include:

**To the user**

1. The user has to be connected to the internet in order to use the application.
2. The user has to have an email account in order to login.
3. The user has to have an android phone with the latest android version.
4. Has to be a 2GB RAM and above.

**To the Developer**

1. The developer has to have android studio, Laptop and must be versant with java as the programming language.
2. Must know how to code database using firebase.

# **CHAPTER 3: INTENDED APPROACH/ DEVELOPMENT METHODOLOGY**

## **Methodology**

The methodology used in this project is the waterfall system development life cycle methodology. In this method, the whole process is divided into different phases that require completion before proceeding to the next phase of the cycle. The results of each phase are what lead to the next phase.

In each case testing is done to make sure it works before going to the next one.

The reason for using this methodology is that:

* It is easier to understand
* It is easier to follow up on progress
* It provides a clear definition of each phase
* Determines the end goal early
* Milestones are easy to pinpoint
* It is easier to document the development process.

Creating a Gantt chart using this methodology is easier than using v-model or agile. This makes it easy to keep track of the timing and system development process.

**WATERFALL SYSTEM DIAGRAM**

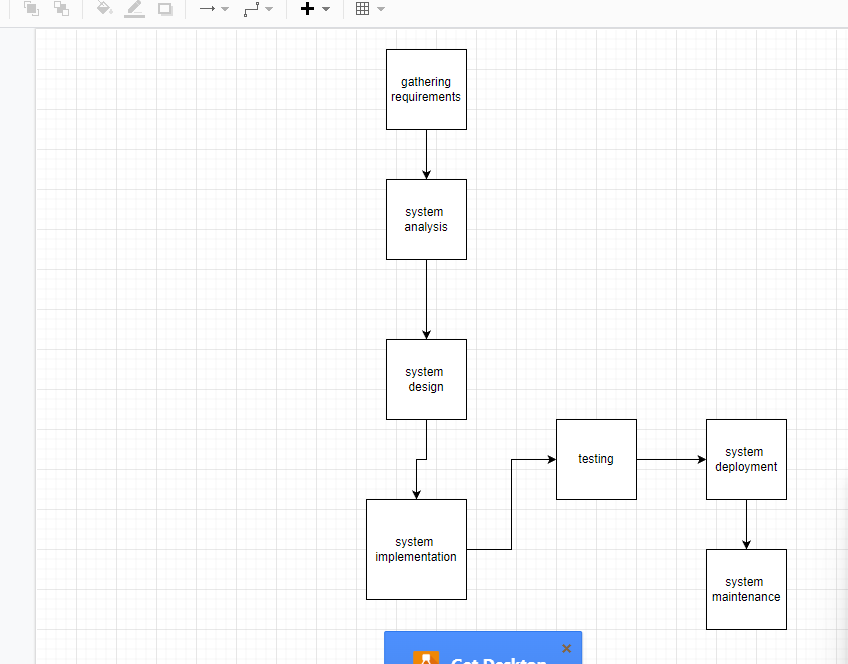


Figure 1

## **Functional and Non-Functional Requirements**

**Functional Requirements**

* The system generates a report
* The user is able to register and log in
* The user’s data will be stored on the database system
* User is requested to either register or log in on opening the application.
* The system shows items from the database
* The system is able to show items in groups
* The system is able to view, delete or add items on the application’s database

**Non-Functional Requirements**

* The system is able to accommodate users.
* The system displays the confirmation message as a toast after making a registration or logging in
* The system retrieves and use data from the database once a user requests to login
* The default background color of the system shall be white after logging in.
* The system shows a toast message after updating, deleting, and adding the items

## **Design tools**

**Visio**

In Visio, it makes it easy to draw a flowchart**.** An exampleis the waterfall diagram

**Diagram.net**

It makes it easy to draw a use case diagram which on this website has a graphical representation. Use case diagrams are graphical representations of how a user interacts with a system. This helps in understanding the functioning of a system and aids in the identification of system requirements.

Entity-relationship diagrams are a graphical representation of the flow of data through different entities in a system. They show the relationships between different attributes of different entities in a system. This helps to ease understanding of the system. I shall therefore use an entity-relationship diagram to display the relationships between various entities and to show the flow of data within the system.

**Mobile App Wireframe**

This brings out the actual image of a mobile phone application and which makes it easy to create one using the options on it.

## **Development Tools**

### **Database development tools**

**Firebase**

Firebase provides detailed documentation and cross-platform SDKs to help you build and ship apps on Android, iOS, the web, C++, and Unity. View all docs. This is a better database management system that can store the inventory on the cloud and can be accessed through the internet.

### **Programming Tools**

**Android Studio**

Android Studio is a mobile application development environment built for the development of mobile applications that run on android operating systems. The reason for choosing this development system environment is because it's well known for creating android applications fast and easily.

**Java programming language**

Java is an object-oriented high-level general-purpose programming language. It's one of the languages used in Android studio which is common. The reason for choosing java is because it can be used for the development of mobile and desktop applications among other things and can work on many platforms such as Linux, Windows, and Mac.

# **CHAPTER 4: SYSTEM ANALYSIS AND DESIGN**

## **Use case diagrams**

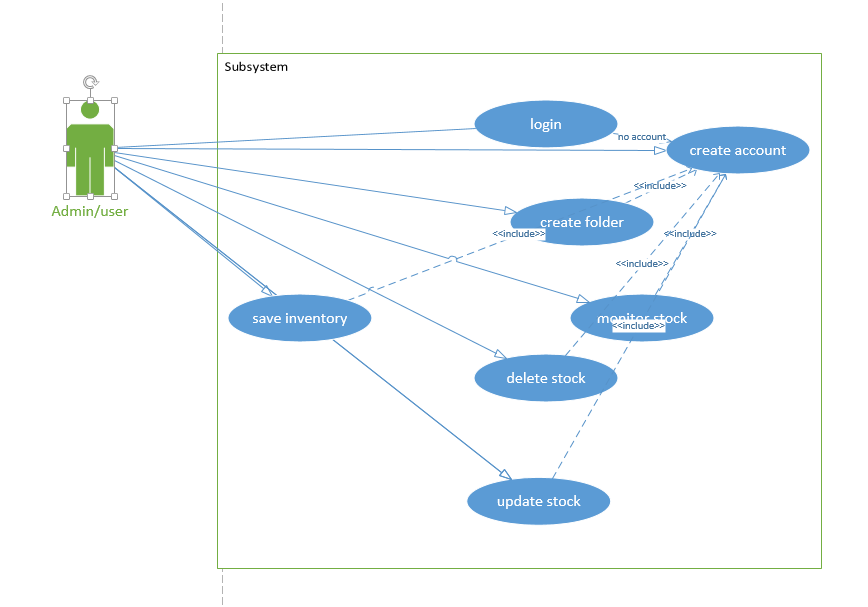


Fig1:1

## **Database Schema**

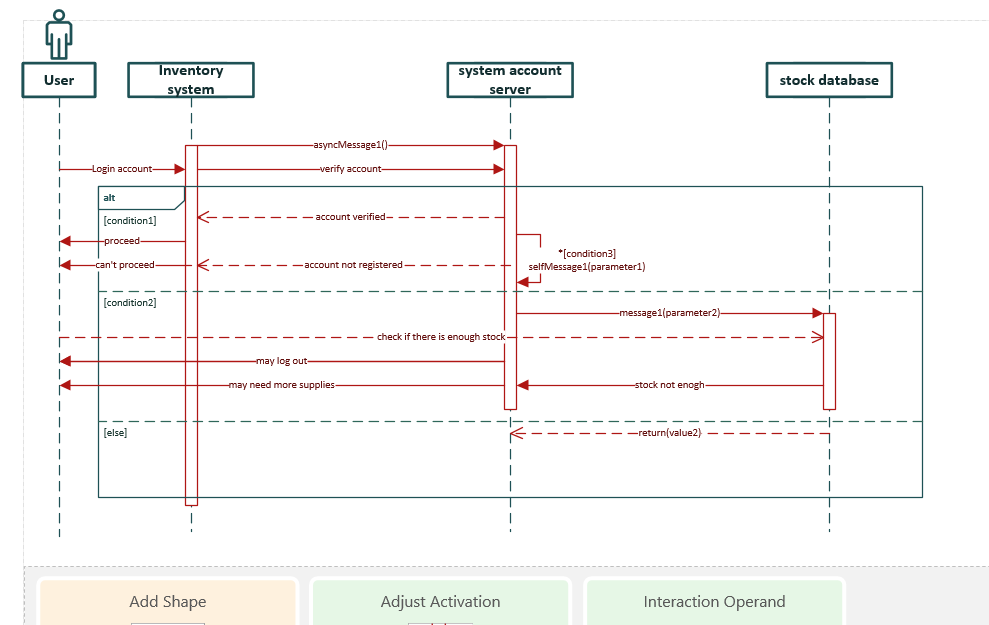
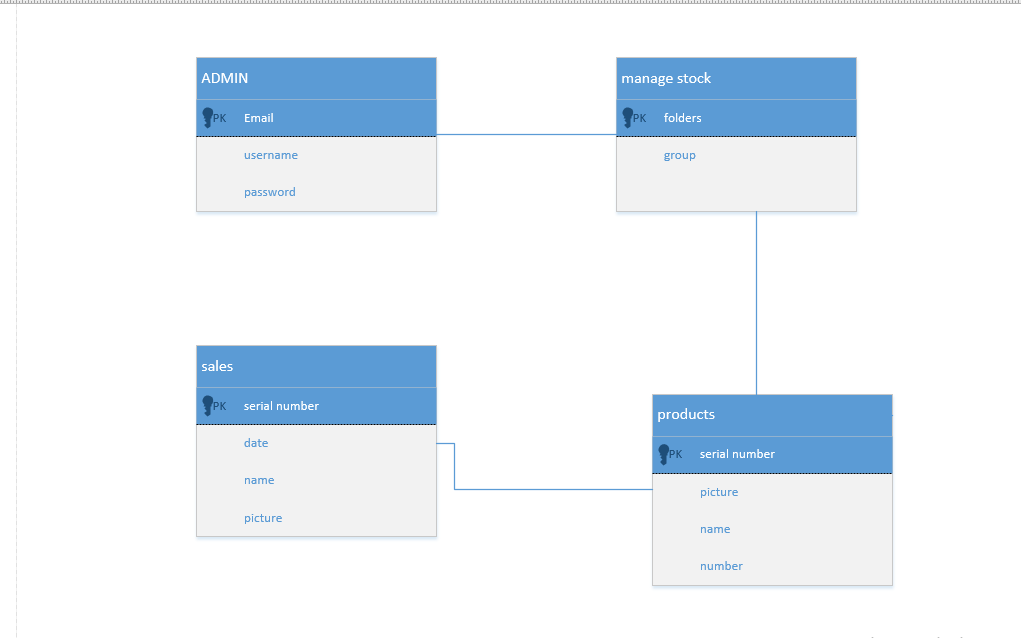


Fig1:2

## **Class diagram**

Fig1:3

## **Graphical User Interface**

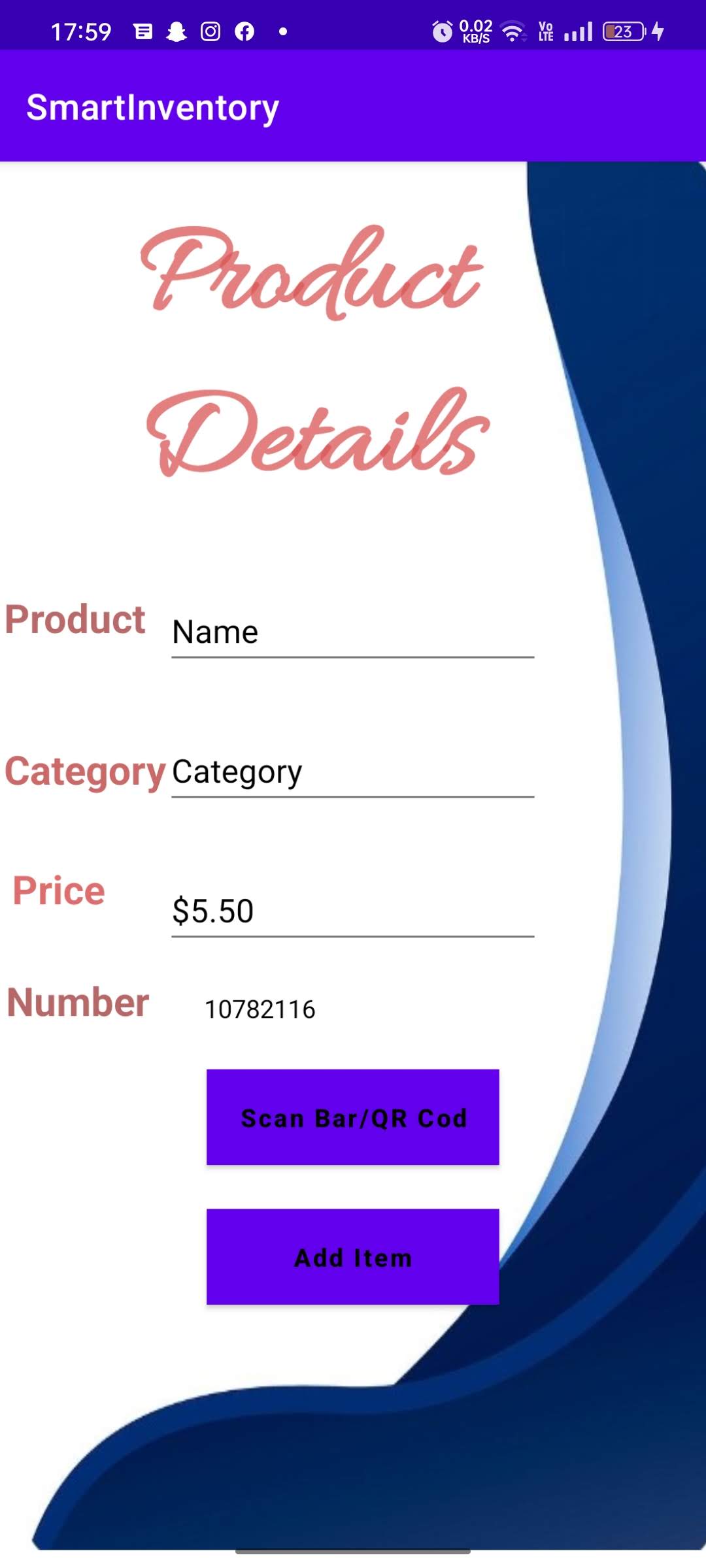
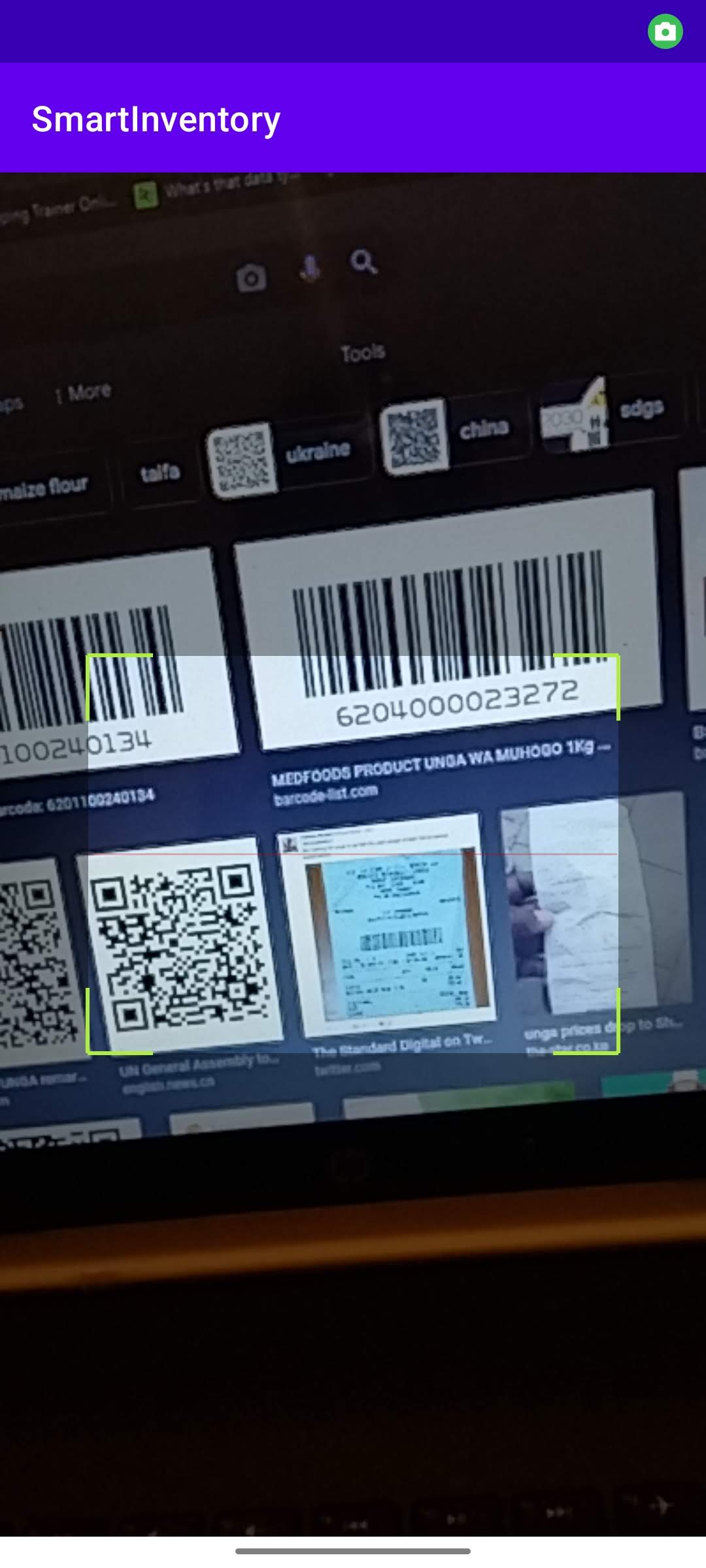


Fig 1:5

This is add in page where the user add the products/items into the application on to the database. It has features like scan barcode to easy the task of the user.

fig1:4  
This is the scanner page where it uses the camera



To scan the barcodes.

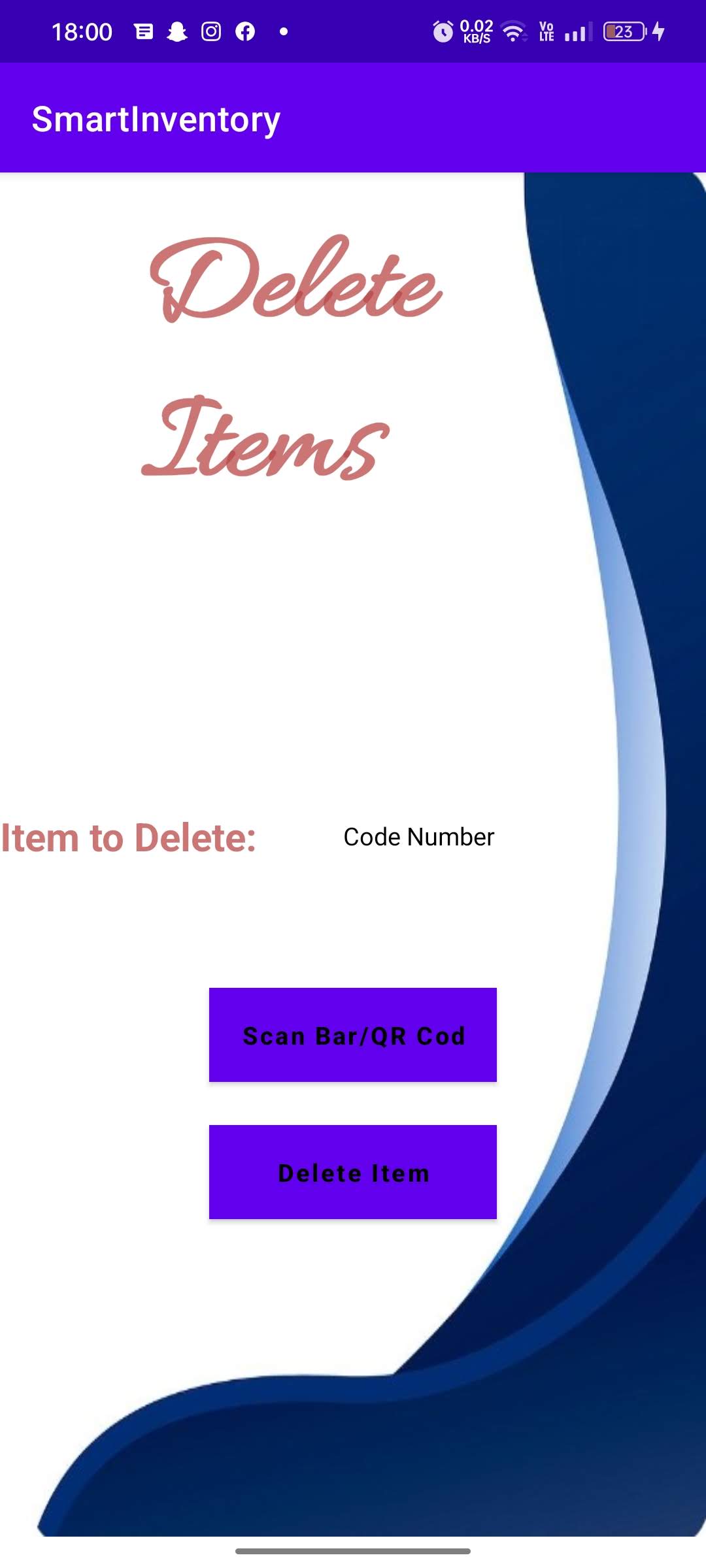


Fig 1:6

This is the delete page there the user deletes the item searching with the barcodes.

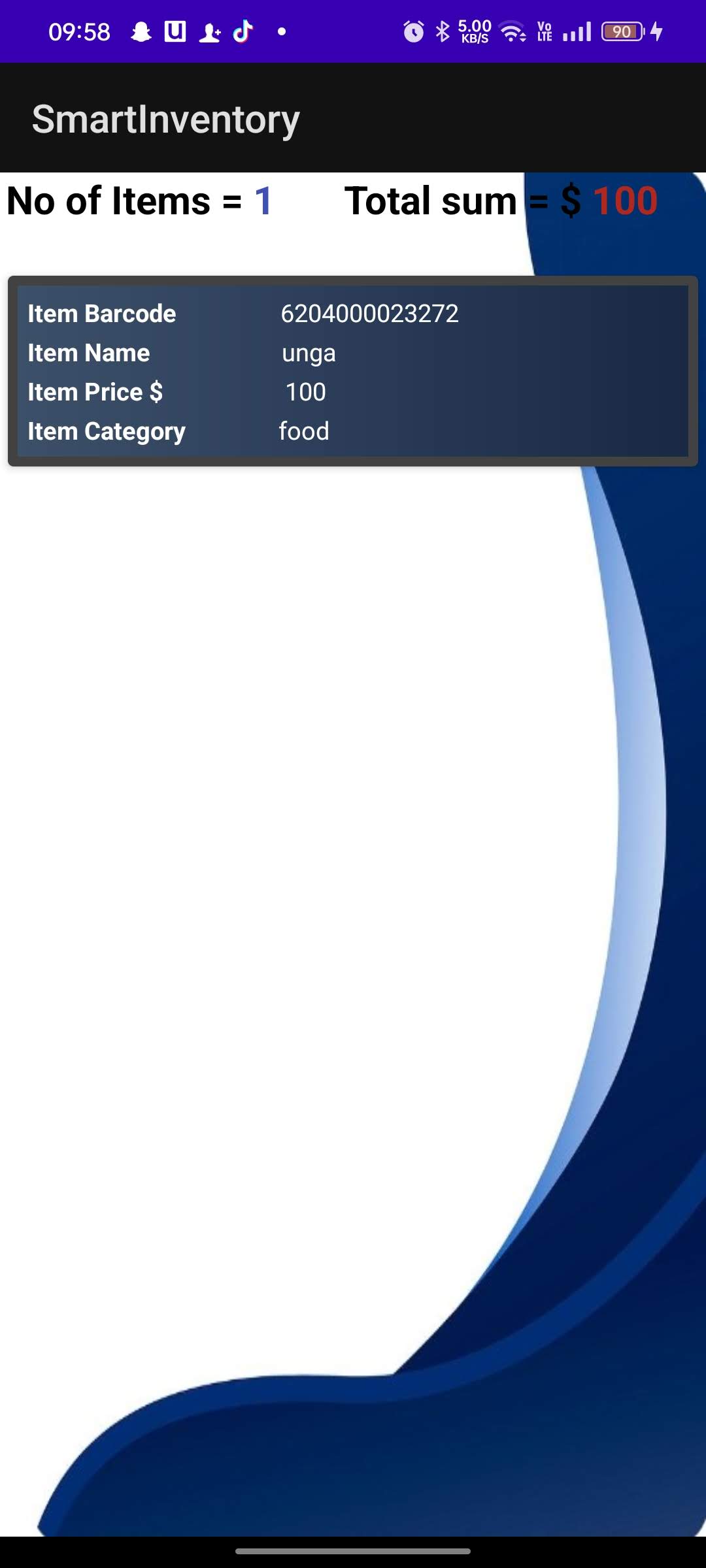


Fig 1:7

This is the report page where the user views the report of their products.

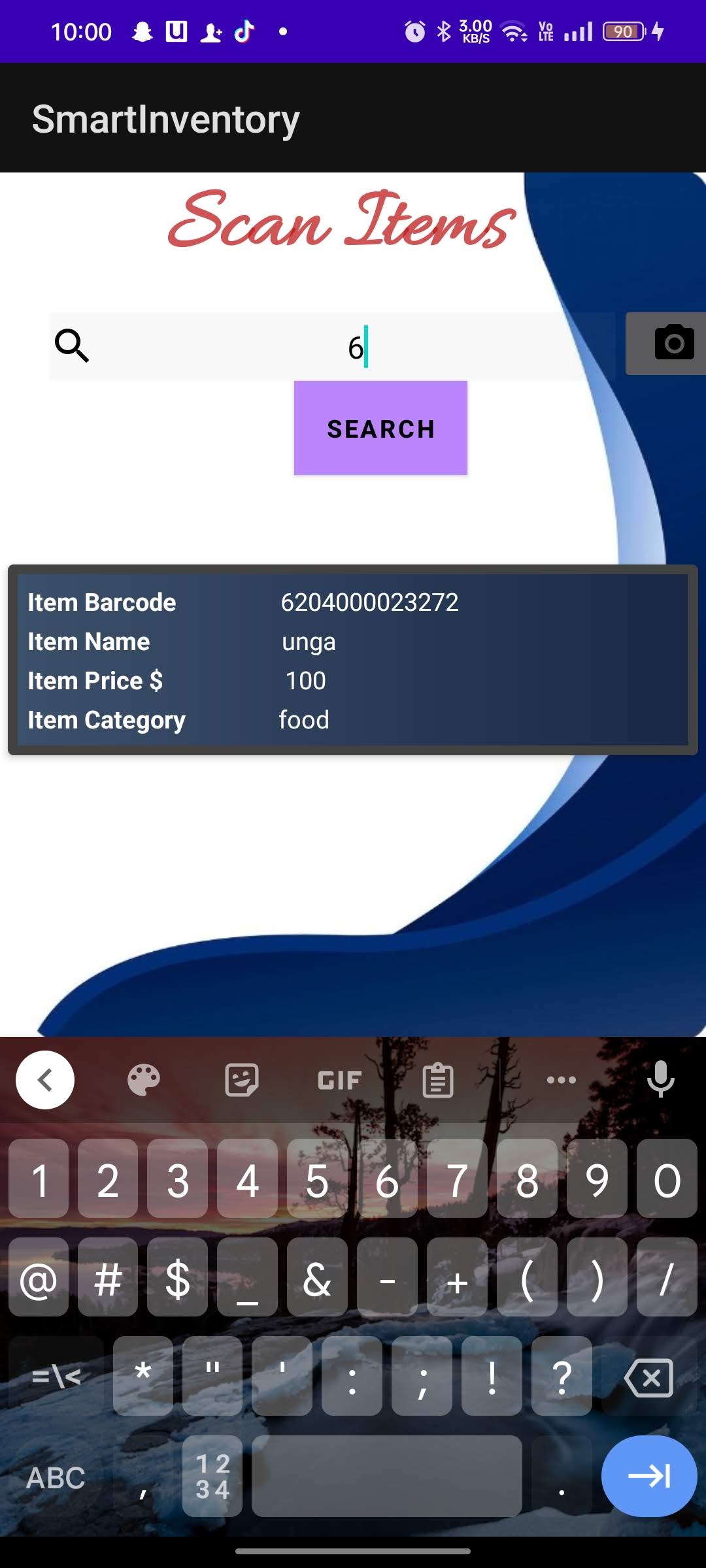


Fig 1:8

This is the search page where the user searches for the items by either search through item name or barcode which can be typed or scanned.

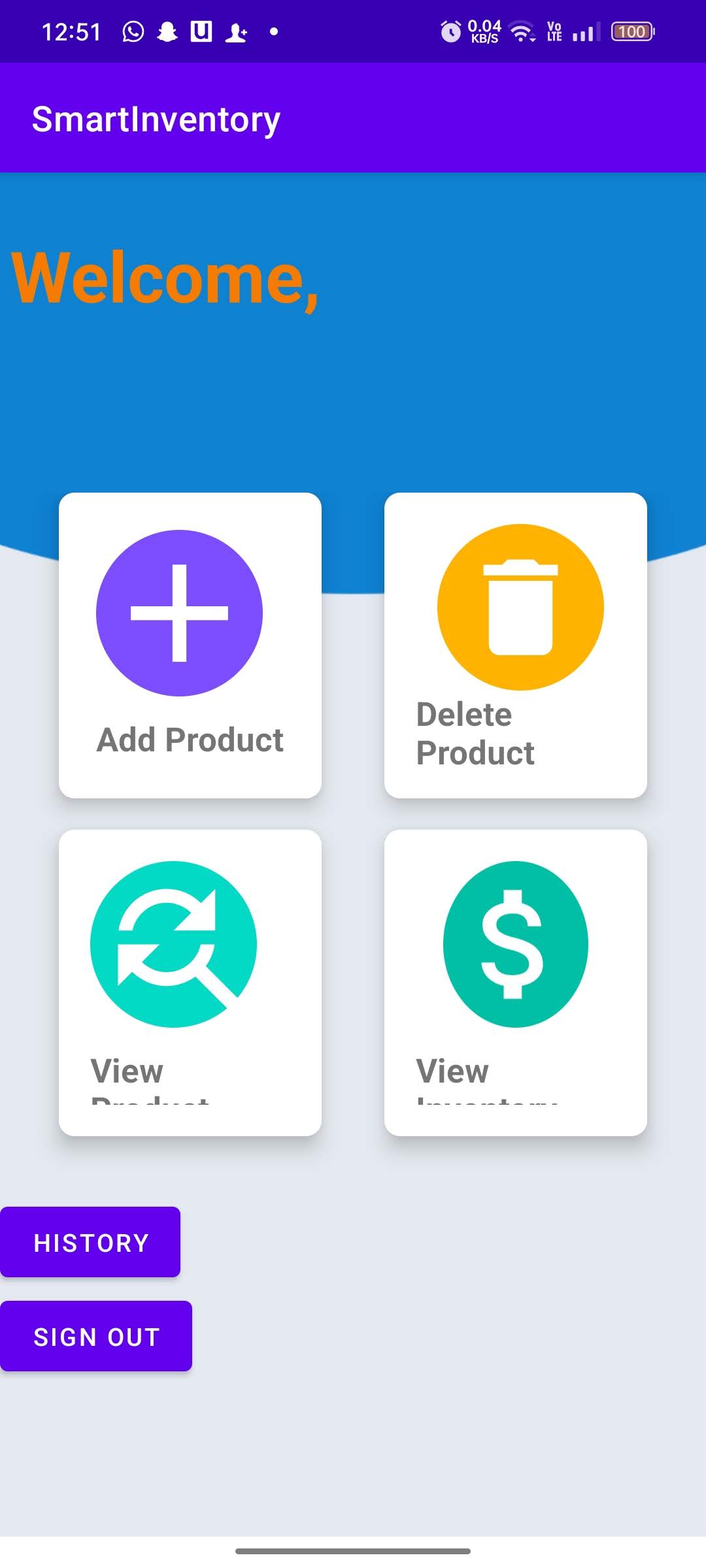


Fig 1:9

This is the dashboard page there it contains all aspects of the application. Sign out, view, delete, search and view the report.



Fig 1:10

This is the splash screen page where it welcomes the user with a form of animation using the logo and the name of the application.

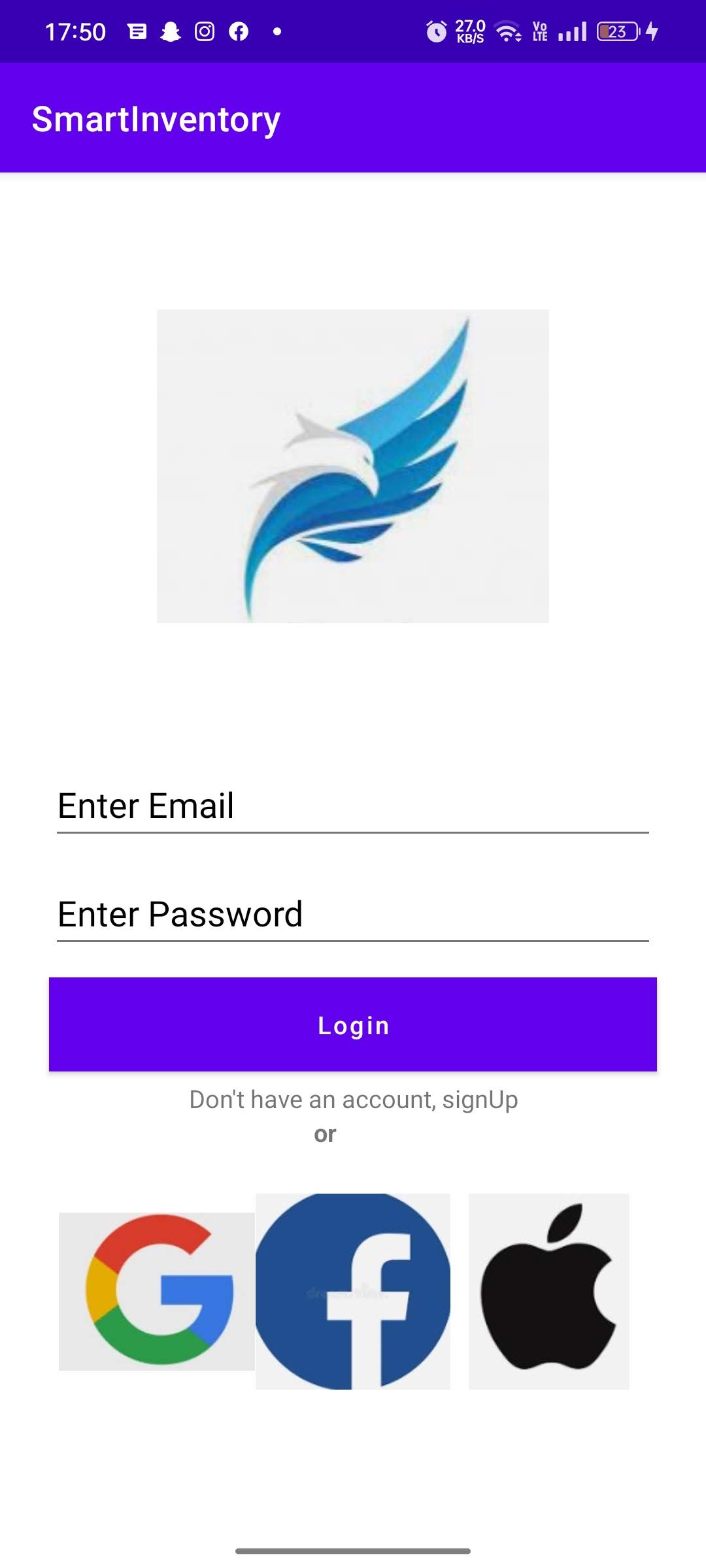


Fig 2:0

This is where the user has to log in the system if the have an account. If no account, the user has to click on the sign up statement to register first.

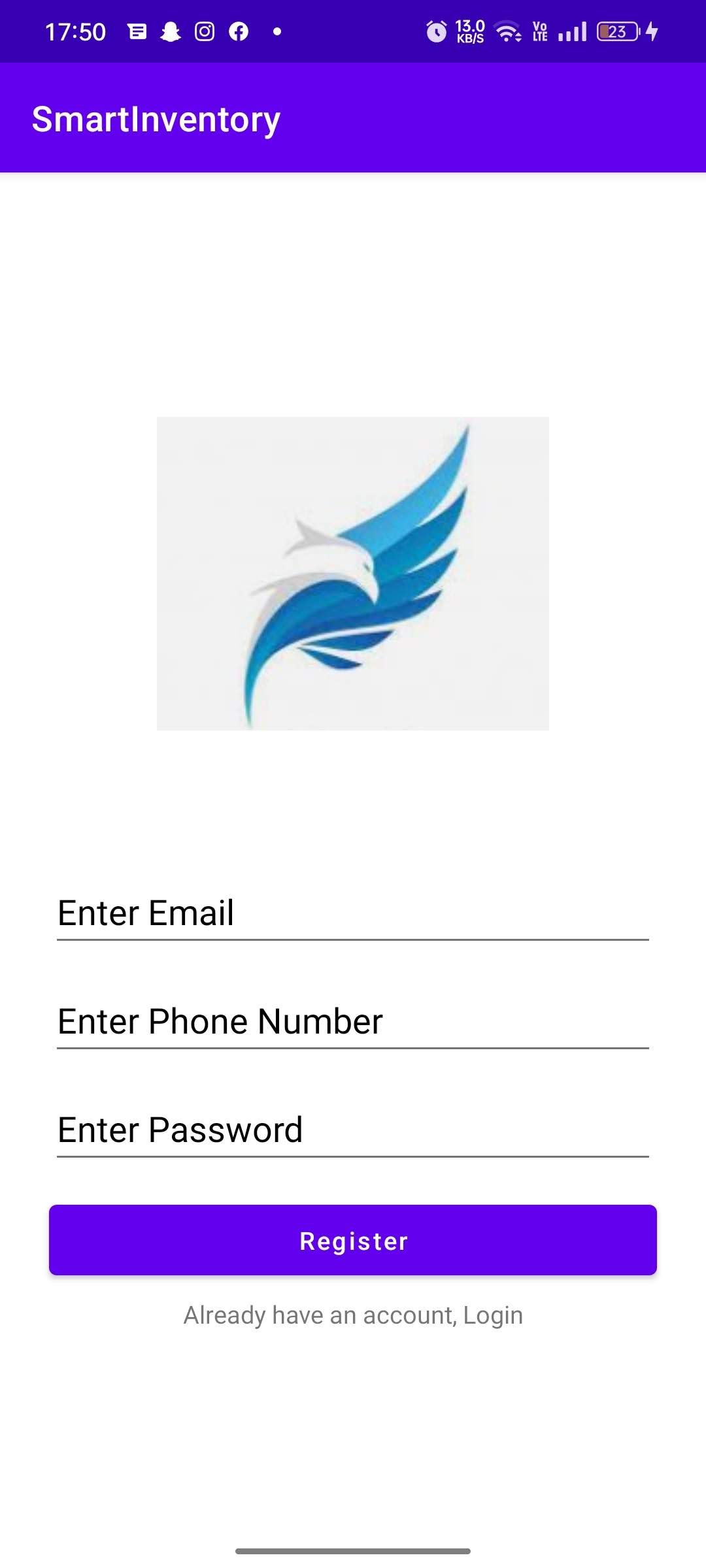


Fig2:1

This is the registration page there the user will have to enter Email, number and password for the first time. This is part of authentication.

# **CHAPTER 5: CONCLUSION**

In conclusion, we have analyzed various benefits of an inventory management system especially when it is done with the help of a handheld device. As the world is revolving around such mobility concepts, adopting these devices to any work environment will let the entrepreneurs reap rewards. It is all about keeping up with the time we live in, and we cannot stay idle when everything around us is moving quickly and smartly.

The time has come to take the business mobile and save the time and effort so that the same can be further utilized in expanding the business. Mobile Enterprise computing is necessary for day-to-day transactions of enterprises or companies. It makes transactions faster and easier because it supports mobility.

## **Future Development**

In the future development of this application, it will be able to accommodate three different users; one it the owner who is to delete, edit, view, and update data. The other is to view and delete and the last one is to view only.

The other update is to be able to view history of the products flow. This will have features like real dates and time on them. This will help users to keep track of their products. The other update is to enable users to add pictures on the products for easy identification.

The last part is adding the settings page, which has features like dark mode, notification of the products needed to be added in stock.

# **REFERENCES**

*Creating a Mobile Application site*. (n.d.). Retrieved from https://app.moqups.com: https://app.moqups.com/eG3GBLfpZHPIqSiclGfBjuWLKc99eimp/edit/page/ae8fe8eb0

*sortly*. (n.d.). Retrieved from https://www.sortly.com/

Space, S. (Director). (2015). *Moving from Traditional to Modern Inventory Systems* [Motion Picture].

# **APPENDIX**

This should generally address all the envisaged resources that will enable the development of the system to succeed. The key items are hardware (smartphone, laptop), software (android studio), human (Admin, developer), and any other costs that will be incurred.

## **Sample budget**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantity** | **Price Per Item (Kshs)** | **Total** |
| Laptop | 1 | 60,000 | 60,000 |
| Android mobile phone | 1 | 30,000 | 30,000 |
| USB –B cable | 1 | 1500 | 1,500 |
| Software*(Android studio)* |  | free | 0 |
| Software*(Microsoft Project)* |  | free | 0 |
| Firebase |  | free | 0 |
| Visio |  | free | 0 |
| Diagram.net |  | free | 0 |
| Mobile App |  | free | 0 |
| Wireframe |  | free | 0 |
| **Total** |  |  | **100,500** |
|  |  |  |  |

**Table 1:1**

## **Project schedule**

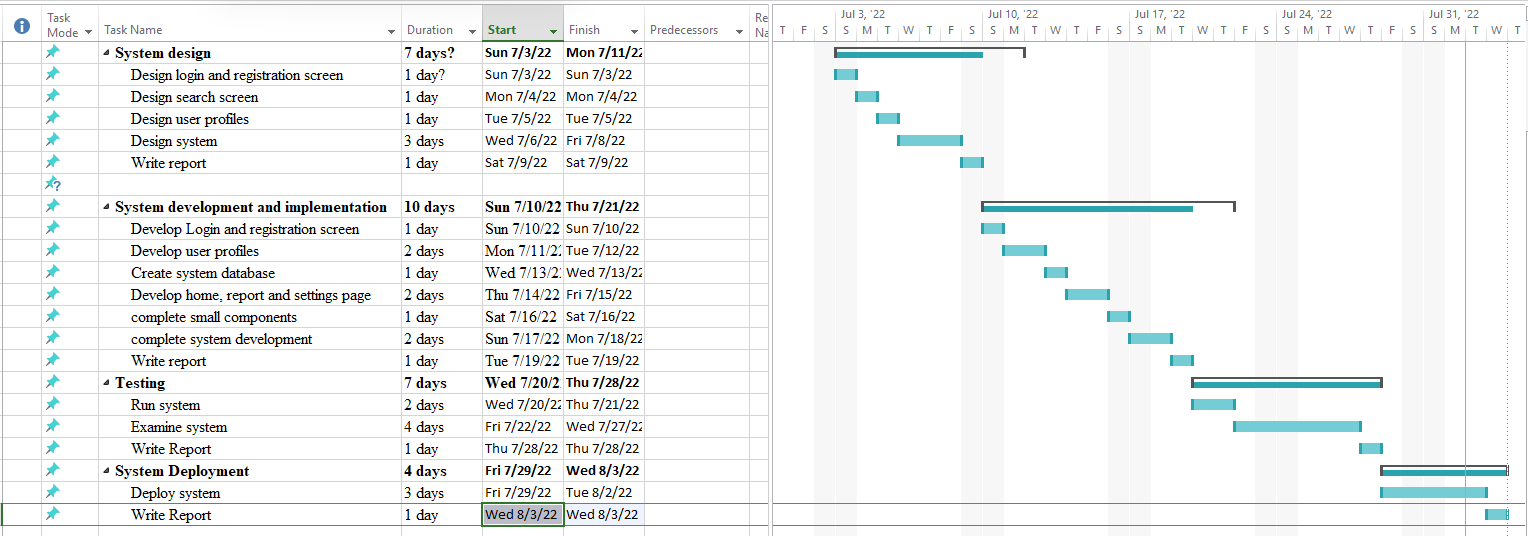
This is a breakdown of all the tasks that will be carried out in the course of carrying out the project. An estimate of the duration of each task in hours should be done. A work breakdown schedule should be formulated and a task schedule/Gantt chart prepared (Preferably using Microsoft Project)

Task Schedule

|  |
| --- |
|  |
| Task Mode | Task Name | Duration | Start | Finish | Predecessors |
|  |  |  |  |  |  |
| **Manually Scheduled** | **System design** | **7 days?** | **Sun 7/3/22** | **Mon 7/11/22** |  |
| Manually Scheduled | Design login and registration screen | 1 day? | Sun 7/3/22 | Sun 7/3/22 |  |
| Manually Scheduled | Design search screen | 1 day | Mon 7/4/22 | Mon 7/4/22 |  |
| Manually Scheduled | Design user profiles | 1 day | Tue 7/5/22 | Tue 7/5/22 |  |
| Manually Scheduled | Design system | 3 days | Wed 7/6/22 | Fri 7/8/22 |  |
| Manually Scheduled | Write report | 1 day | Sat 7/9/22 | Sat 7/9/22 |  |
|  |  |  |  |  |  |
| **Manually Scheduled** | **System development and implementation** | **10 days** | **Sun 7/10/22** | **Thu 7/21/22** |  |
| Manually Scheduled | Develop Login and registration screen | 1 day | Sun 7/10/22 | Sun 7/10/22 |  |
| Manually Scheduled | Develop user profiles | 2 days | Mon 7/11/22 | Tue 7/12/22 |  |
| Manually Scheduled | Create system database | 1 day | Wed 7/13/22 | Wed 7/13/22 |  |
| Manually Scheduled | Develop home, report and settings page | 2 days | Thu 7/14/22 | Fri 7/15/22 |  |
| Manually Scheduled | complete small components | 1 day | Sat 7/16/22 | Sat 7/16/22 |  |
| Manually Scheduled | complete system development | 2 days | Sun 7/17/22 | Mon 7/18/22 |  |
| Manually Scheduled | Write report | 1 day | Tue 7/19/22 | Tue 7/19/22 |  |
| **Manually Scheduled** | **Testing** | **7 days** | **Wed 7/20/22** | **Thu 7/28/22** |  |
| Manually Scheduled | Run system | 2 days | Wed 7/20/22 | Thu 7/21/22 |  |
| Manually Scheduled | Examine system | 4 days | Fri 7/22/22 | Wed 7/27/22 |  |
| Manually Scheduled | Write Report | 1 day | Thu 7/28/22 | Thu 7/28/22 |  |
| **Manually Scheduled** | **System Deployment** | **4 days** | **Fri 7/29/22** | **Wed 8/3/22** |  |
| Manually Scheduled | Deploy system | 3 days | Fri 7/29/22 | Tue 8/2/22 |  |
| Manually Scheduled | Write Report | 1 day | Wed 8/3/22 | Wed 8/3/22 |  |

**Table 1:2**

GANTT CHART.



**Table 2:2**

## **Sample Codes**

**ADD PAGE CODES**

// addding item to databse  
public void additem() {  
 String itemnameValue = itemname.getText().toString();  
 String itemcategoryValue = itemcategory.getText().toString();  
 String itempriceValue = itemprice.getText().toString();  
 String itembarcodeValue = itembarcode.getText().toString();  
 final FirebaseUser users = mAuth.getCurrentUser();  
 String finaluser = users.getEmail();  
 String resultemail = finaluser.replace(".", "");  
 if (itembarcodeValue.isEmpty()) {  
 itembarcode.setError("It's Empty");  
 itembarcode.requestFocus();  
 return;  
 }  
 if(!TextUtils.*isEmpty*(itemnameValue)&&!TextUtils.*isEmpty*(itemcategoryValue)&&!TextUtils.*isEmpty*(itempriceValue)){  
  
 Items items = new Items(itemnameValue,itemcategoryValue,itempriceValue,itembarcodeValue);  
 databaseReference.child(resultemail).child("Items").child(itembarcodeValue).setValue(items);  
 databaseReferencecat.child(resultemail).child("ItemByCategory").child(itemcategoryValue).child(itembarcodeValue).setValue(items);  
 itemname.setText("");  
 itembarcode.setText("");  
 itemprice.setText("");  
 itembarcode.setText("");  
 Toast.*makeText*(addItemsActivity.this,itemnameValue+" Added",Toast.*LENGTH\_SHORT*).show();  
 }  
 else {  
 Toast.*makeText*(addItemsActivity.this,"Please Fill all the fields",Toast.*LENGTH\_SHORT*).show();

Add page

**DASHBOARD**

analyticsBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(Dashboard.this, Analytics.class);  
 startActivity(intent);  
 }  
 });  
 addItems = (CardView)findViewById(R.id.*addItems*);  
 deleteItems = (CardView) findViewById(R.id.*deleteItems*);  
 scanItems = (CardView) findViewById(R.id.*scanItems*);  
 viewInventory = (CardView) findViewById(R.id.*viewInventory*);  
  
 addItems.setOnClickListener(this);  
 deleteItems.setOnClickListener(this);  
 scanItems.setOnClickListener(this);  
 viewInventory.setOnClickListener(this);  
  
}  
@Override  
public void onClick(View view) {  
 Intent i;  
  
 switch (view.getId()){  
 case R.id.*addItems* : i = new Intent(this,addItemsActivity.class); startActivity(i); break;  
 case R.id.*deleteItems* : i = new Intent(this,deleteItemsActivity.class);startActivity(i); break;  
 case R.id.*scanItems* : i = new Intent(this,scanItemsActivity.class);startActivity(i); break;  
 case R.id.*viewInventory* : i = new Intent(this,viewInventoryActivity.class);startActivity(i); break;  
 default: break;  
 }

**DELETE PAGE**

public void deletefrmdatabase()  
{  
 String deletebarcodevalue = *resultdeleteview*.getText().toString();  
 final FirebaseUser users = mAuth.getCurrentUser();  
 String finaluser=users.getEmail();  
 String resultemail = finaluser.replace(".","");  
 if(!TextUtils.*isEmpty*(deletebarcodevalue)){  
 databaseReference.child(resultemail).child("Items").child(deletebarcodevalue).removeValue();  
 Toast.*makeText*(deleteItemsActivity.this,"Item is Deleted",Toast.*LENGTH\_SHORT*).show();  
 }  
 else{  
 Toast.*makeText*(deleteItemsActivity.this,"Please scan Barcode",Toast.*LENGTH\_SHORT*).show();

**ITEM CLASS**

public class Items {  
 private String itemname;  
 private String itemcategory;  
 private String itemprice;  
 private String itembarcode;  
  
  
 public Items() {  
  
 }  
  
 public Items(String itemname,String itemcategory,String itemprice,String itembarcode){  
  
 this.itemname=itemname;  
 this.itemcategory=itemcategory;  
 this.itemprice=itemprice;  
 this.itembarcode= itembarcode;  
 }  
  
 public String getItemname() {  
 return itemname;  
 }  
  
 public String getItemcategory() {  
 return itemcategory;  
 }  
  
 public String getItemprice() {  
 return itemprice;  
 }  
  
 public String getItembarcode() {  
 return itembarcode;  
 }  
}

**LOGIN PAGE**

@Override  
public void onClick(View v) {  
 String emailString = email.getText().toString();  
 String passwordString = password.getText().toString();  
  
 if (TextUtils.*isEmpty*(emailString)){  
 email.setError("Email is required");  
 }  
 if (TextUtils.*isEmpty*(passwordString)){  
 password.setError("Password is required");  
 }  
  
 else{  
 progressDialog.setMessage("login in progress");  
 progressDialog.setCanceledOnTouchOutside(false);  
 progressDialog.show();  
  
 mAuth.signInWithEmailAndPassword(emailString, passwordString).addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if(task.isSuccessful()){  
 Intent intent = new Intent(login.this, Dashboard.class);  
 startActivity(intent);  
  
  
 progressDialog.dismiss();  
 }else{  
 Toast.*makeText*(login.this, task.getException().toString(), Toast.*LENGTH\_SHORT*).show();  
 progressDialog.dismiss();  
 }  
 }

**SPLASH PAGE**

@Override  
 public void run() {  
 OnBoarding = getSharedPreferences("OnBoarding", *MODE\_PRIVATE*);  
 boolean isFirstTime = OnBoarding.getBoolean("firstTime", true);  
 if (isFirstTime){  
 SharedPreferences.Editor editor = OnBoarding.edit();  
 editor.putBoolean("firstTime", false);  
 editor.commit();  
  
 Intent intent = new Intent(MainActivity.this, login.class);  
 startActivity(intent);  
 finish();  
 }else {  
 Intent intent = new Intent(MainActivity.this, login.class);  
 startActivity(intent);  
 finish();  
 }  
  
 }  
},*SPLASH*);

**REGISTER PAGE**

registerBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String emailString = email.getText().toString();  
 String phoneString = phone\_no.getText().toString();  
 String passwordString = password.getText().toString();  
  
 if (TextUtils.*isEmpty*(emailString)){  
 email.setError("Email is required");  
 }  
 if (TextUtils.*isEmpty*(phoneString)){  
 phone\_no.setError("Phone Number is required");  
 }  
 if (TextUtils.*isEmpty*(passwordString)){  
 password.setError("Password is required");  
 }  
  
 else{  
 progressDialog.setMessage("Registration in progress");  
 progressDialog.setCanceledOnTouchOutside(false);  
 progressDialog.show();  
  
 mAuth.createUserWithEmailAndPassword(emailString, passwordString).addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if(task.isSuccessful()){  
 Intent intent = new Intent(RegistrationActivity.this,login.class);  
 startActivity(intent);  
  
 progressDialog.dismiss();  
 }else{  
 Toast.*makeText*(RegistrationActivity.this, task.getException().toString(), Toast.*LENGTH\_SHORT*).show();  
 progressDialog.dismiss();  
 }  
 }  
 });

**SCANNER PAGE**

@Override  
protected void onPostResume() {  
 super.onPostResume();  
 if (ContextCompat.*checkSelfPermission*(getApplicationContext(), Manifest.permission.*CAMERA*)  
 != PackageManager.*PERMISSION\_GRANTED*) {  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*CAMERA*},  
 MY\_PERMISSIONS\_REQUEST\_CAMERA);  
 }  
 scannerView.setResultHandler(this);  
 scannerView.startCamera();

**DISPLAY SEARCHED ITEMS PAGE**

public static class UsersViewHolder extends RecyclerView.ViewHolder{  
 View mView;  
 public UsersViewHolder(View itemView){  
 super(itemView);  
 mView =itemView;  
 }  
  
 public void setDetails(Context ctx,String itembarcode, String itemcategory, String itemname, String itemprice){  
 TextView item\_barcode = (TextView) mView.findViewById(R.id.*viewitembarcode*);  
 TextView item\_name = (TextView) mView.findViewById(R.id.*viewitemname*);  
 TextView item\_category = (TextView) mView.findViewById(R.id.*viewitemcategory*);  
 TextView item\_price = (TextView) mView.findViewById(R.id.*viewitemprice*);  
  
 item\_barcode.setText(itembarcode);  
 item\_category.setText(itemcategory);  
 item\_name.setText(itemname);  
 item\_price.setText(itemprice);  
 }

**VIEW INVENTORY**

mdatabaseReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 int sum = 0;  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 Map<String, Object> map = (Map<String, Object>) ds.getValue();  
 Object price = map.get("itemprice");  
 int pValue = Integer.*parseInt*(String.*valueOf*(price));  
 sum += pValue;  
 totalnoofsum.setText(String.*valueOf*(sum));  
  
 }  
 }

protected void onStart() {  
 super.onStart();  
  
 FirebaseRecyclerAdapter<Items, scanItemsActivity.UsersViewHolder> firebaseRecyclerAdapter = new FirebaseRecyclerAdapter<Items, scanItemsActivity.UsersViewHolder>  
 (Items.class,  
 R.layout.*list\_layout*,  
 scanItemsActivity.UsersViewHolder.class,  
 mdatabaseReference) {  
 @Override  
 protected void populateViewHolder(scanItemsActivity.UsersViewHolder viewHolder, Items model, int position) {  
  
 viewHolder.setDetails(getApplicationContext(), model.getItembarcode(), model.getItemcategory(), model.getItemname(), model.getItemprice());  
 }  
 };  
  
 mrecyclerview.setAdapter(firebaseRecyclerAdapter);  
}