**A**

**MAJOR PROJECT REPORT**

**ON**

**STORE POS AND INVENTORY MANAGEMENT APP**

**Submitted in partial fulfillment of the requirements**

**For the award of Degree of**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

**M. ARYA SRI GANESH 245318733155**

**G. VISHAL 245318733139**

**MVS PRANAVI 245318733153**

**Under the guidance**

**Of**

**Dr. K. VINUTHNA**

**ASSOCIATE PROFESSOR**



**Department of Computer Science and Engineering**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

Kachavanisingaram Village, Hyderabad, Telangana 500058.

**June 2022**

 **NEIL GOGTE INSTITUTE OF TECHNOLOGY**

**A Unit of Keshav Memorial Technical Education (KMTES)**

**Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad**

**2018-2022**

**CERTIFICATE**

*This is to certify that the project work entitled* **“STORE POS AND INVENTORY MANAGEMENT APP***” is a bonafide work carried out by* **M. ARYA SRI GANESH (245318733155), G. VISHAL (245318733139), MVS PRANAVI (245318733153)** of IV-year VIII semester **Bachelor of Engineering** *in* **COMPUTER SCIENCE AND ENGINEERING** *by Osmania University, Hyderabad during the academic year* **2018-2022** *is a record of bonafide work carried out by them*. *The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.*

**Internal Guide Head of Department**

Dr. K. VINUTHNA Dr. K. V. Ranga Rao

**External**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

**A Unit of Keshav Memorial Technical Education (KMTES)**

**Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad**

**DECLARATION**

We hereby declare that the Major Project Report entitled, “**STORE POS AND INVENTORY MANAGEMENT APP”** submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

**Date:**

**M. ARYA SRI GANESH (245318733155)**

**G. VISHAL (245318733139)**

**MVS PRANAVI (245318733153)**

**NEIL GOGTE INSTITUTE OF TECHNOLOGY**

**A Unit of Keshav Memorial Technical Education (KMTES)**

**Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad**

**ACKNOWLEDGEMENT**

We are happy to express our deep sense of gratitude to the principal of the college **Dr. D. Jaya Prakash,** Professor, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**,** **Dr. K.V. Ranga Rao**, **Head of the Department,** Computer Science and Engineering, Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide **Dr. K. Vinuthna, Associate Professor** for her Technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

### CONTENT

### 

### 

### DESCRIPTION PAGE NO.

### ABSTRACT i

### LIST OF FIGURES ii

### CHAPTERS 1

### INTRODUCTION 2

### 1.1. Purpose of the Project 2

### 1.2. Problem with the Existing System 2

### 1.3. Proposed System 3

### 1.4. Scope of the Project 3

### 2. SOFTWARE REQUIREMENTS 5

### SPECIFICATIONS

### 2.1. What is SRS 5

### 2.2. Role Of SRS 5

### 2.3. Requirements Specification Document 5

### 2.4. Functional Requirements 6

### 2.5. Non-Functional Requirements 7

### 2.6. Software Requirements 7

### 2.7. Hardware Requirements 7

### 2.8. Technologies Used 8

### 3. LITERATURE SURVEY 16

### 4. SYSTEM DESIGN 18

### 4.1. Introduction to UML 18

### 4.2. Use Case Diagrams 19

### 4.3. Sequence diagram 21

### 4.4. Class diagram 23

### 4.5. Activity diagram 24

### 5. IMPLEMENTATION 26

### 5.1. Login Page 26

### 5.2. Home Page 29

### 5.3. Goods Fragment 31

### 5.4. Billing Fragment 37

### 5.5. Add Product 41

### 5.6. Database Functions 62

### 6. TEST CASES 67

### 7. SCREENSHOTS 69

### 7.1. Login Page 69

### 7.2. List Of Goods 70

### 7.3. Billing Page 71

### 7.4. Adding a New Product 72

### 7.5. Product Added 72

### 8.FURTHER ENHANCEMENTS 74

### 9.CONCLUSION 76

### 10.REFERENCES 78

## ABSTRACT

STORE POS AND INVENTORY MANAGEMENT APP is a project developed to automate the processes to reduce the clerical labour of the staff working in both technical and as well as accounts departments in a store. This system uses the latest technologies and cost-effective tools there-by providing the better control to the stores by preventing manual errors. In the existing system, all the details of the records are maintained manually. Details of the items are not maintained properly. It consumes a lot of time and is less secured. Records are not maintained in a proper order. Loss of data may occur. This proposed system maintains the information in the database. It is used as an intranet application. This system provides high-security.

This system provides following features:

 Ability to track inventory status

 Set up alerts as per the sales history

 Keep track of sale/purchase orders to plan out purchases

 Ability to generate bill.

 Reduced cost price for hardware

## LIST OF FIGURES

### FIG NO LIST OF FIGURES PAGE NO

4.2 Use Case Diagrams 20

4.3 Sequence Diagram 22

4.4 Class Diagram 23

4.5 Activity Diagram 24

6.1 Login Page 69

6.2 List of Good 70

6.3 Billing Page 71

6.4 Adding a new Product 72

6.5 Product Added 72

**CHAPTER - 1**

## INTRODUCTION

### 1.1 Purpose of the project

* Competition is rapidly growing in business requiring entrepreneurs to create product excellence and to study the behaviour of consumers.
* Technological developments enable employers in small, medium and large companies to do business by using Internet.
* Now the development of Internet can be integrated with mobile devices such as mobile phones to make the flow of information more widely
* This application facilitates employee of stores in organizing, processing, monitoring the movement of inventory items that occurred in the stores.

### 1.2 Problem with the Existing System

**Lack of Security**: The paper document is less secure compared to an electronic system. Misplaced of documents can easily get into wrong hands. The organization Secret or classified information is unsafe.

**Time Consuming:** Manually managing is a very tough and time-consuming process. Manually checking and tracking each and every product in the job is not easy

**Difficulty in a modification of data**: When working with paper documents, it is much harder to make changes. If you want to make any change you will have to make a copy, so you don’t destroy the original with any edits or comments you might add. This means the editing process is more time consuming and costly than if you were working with digital copies.

**Increases Cost:** One of the biggest drawbacks of manually management system is the associated costs. These costs quickly add up can become a significant expense in larger organizations leads to decrease in the organizational profit.

### Proposed System

It is totally mobile-based application which makes use of internet and a database to provide following services to the management of the stores:

* Maintaining stock
* Monitoring the movement of stock
* Identifying the products which are nearing they expiry date
* Identifying the products which are getting low in number
* Calculating annual statistics like annual sales & annual profit

### Scope of the Project

Store manager is an android application which stores, controls stock related information in the stores and also allows management to generate statistical data like annual sales, annual profit along with generation of bill to the customer.

As this is a generic application, it can be used by a wide range variety of outlets such as Retailers and Wholesalers to automate the process of manually maintaining the records. So that organization can manage their sore in efficient and organized form.

**CHAPTER - 2**

## 2.SYSTEM RERUIREMENT SPECIFICATIONS

**2.1 What is SRS?**

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase)

The SRS phase consists of two basic activities:

**Problem/Requirement Analysis:**

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

**Requirement Specification:**

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

### 2.2 Role of SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

### 2.3 Requirements Specification Document

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and non functional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behaviour of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS. This section introduces the requirement specification document for Word Building Game using Alexa which enlists functional as well as non-functional requirements.

### 2.4 Functional Requirements

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data. Functional requirements define specific behaviour or function of the application. Following are the functional requirements:

FR1) Login with Google

FR2) Storing Stock Details to the Database

FR3) Displaying Stock Details

FR4) Generating Bill

FR5) Calculating Annual Statistics

FR6) Storing Transaction Details

FR7) Managing Creditors Account

### 2.5 Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviour. Especially these are the constraints the system must work within. Following are the non-functional requirements:

NFR 1) Accurate Results

NFR 2) Availability

NFR 3) User - Friendly Interface

NFR 4) Easy to Maintain

NFR 5) Bug – Free

**2.6 Software Requirements**

Programming Language: Java

Platform: Android Studio IDE

### 2.7 Hardware Requirements

#### System Configuration to Develop Application

Minimum Requirements:

* OS: Windows or Linux or Mac
* Processor: 3.0 GHz
* RAM: 8 GB
* Graphics Card: 2 GB (Optional)
* Basic: Keyboard, Mouse, Internet

#### Mobile Configuration to run the Application

Minimum Requirements:

* OS: Android OS
* Android version: 4.0.0 (Ice Cream Sandwich)
* RAM: 2 GB
* Camera: 5 MP
* SD Card: 2 GB

**2.8 Technologies Used:**

##### 2.8.1 Android Studio IDE

Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps.

Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0.

Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages are supported by Android Studio.

Features of Android Studio

* It has a flexible Gradle-based build system.
* It has a fast and feature-rich emulator for app testing.
* Android Studio has a consolidated environment where we can develop for all Android devices.
* Apply changes to the resource code of our running app without restarting the app.
* Android Studio provides extensive testing tools and frameworks.
* It supports C++ and NDK.
* It provides build-in supports for Google Cloud Platform. It makes it easy to integrate Google Cloud Messaging and App Engine.

Android Studio Project Structure:

The Android Studio project contains one or more modules with resource files and source code files. These include different types of modules-

Hello Java Program for Beginners

* Android app modules
* Library modules
* Google App Engine modules

By default, Android Studio displays our project files in the Android project view, as shown in the above image. This view is formed by modules to provide quick access to our project's key source files.

These build files are visible to the top-

level under Gradle Scripts. And the app module contains

Manifests:

It has the

“

AndroidManifest.xml

”

file.

J

ava:

It contains the source code of Java files, including the

“Junit”

test code.

R

es:

It contains all non

-

code resources, UI strings, XML layouts, and bitmap images.

Gradle build system:

Gradle build is the foundation of the build system in Android Studio. It uses more Android specific capabilities provided by the Android plug-in for Gradle. This build system runs independently from the command line and integrated tool from the Android Studio menu. We can use build features for the following purpose:

* Configure, customize, and extend the build process.
* We can create multiple APKs from our app, with different features using the same project and modules.
* Reuse resource and code across source sets.

##### 2.8.2 XML

XML (Extensible Markup Language) is a mark-up language. XML is designed to store and transport data. Xml was released in late 90’s. It was created to provide an easy to use and store self-describing data. XML became a W3C Recommendation on February 10, 1998. XML is not a replacement for HTML. XML is designed to be self-descriptive. XML is designed to carry data, not to display data. XML tags are not predefined. You must define your own tags. XML is platform independent and language independent. A markup language is a modern system for high-light or under-line a document.

Students often underline or highlight a passage to revise easily, same in the sense of modern markup language highlighting or underlining is replaced by tags.

Platform Independent and Language Independent: The main benefit of xml is that you can use it to take data from a program like Microsoft SQL, convert it into XML then share that XML with other programs and platforms. You can communicate between two platforms which are generally very difficult.

The main thing which makes XML truly powerful is its international acceptance. Many corporations use XML interfaces for databases, programming, office application mobile phones and more. It is due to its platform independent feature.

##### 2.8.3 Java

Our core Java programming tutorial is designed for students and working professionals. Java is an [object-oriented,](https://www.javatpoint.com/java-oops-concepts) class-based, concurrent, secured and general-purpose computer programming language. It is a widely used robust technology.

What is Java?

Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995. James Gosling is known as the father of Java. Before Java, its name was Oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

**Types of Java Applications**

There are mainly 4 types of applications that can be created using Java programming:

#### Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

#### Web Application

An application that runs on the server side and creates a dynamic page is called a web application. Currently, [Servlet,](https://www.javatpoint.com/servlet-tutorial) [JSP,](https://www.javatpoint.com/jsp-tutorial) [Struts,](https://www.javatpoint.com/struts-2-tutorial) [Spring,](https://www.javatpoint.com/spring-tutorial) [Hibernate,](https://www.javatpoint.com/hibernate-tutorial) [JSF,](https://www.javatpoint.com/jsf-tutorial) etc. technologies are used for creating web applications in Java.

#### Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, [EJB](https://www.javatpoint.com/ejb-tutorial) is used for creating enterprise applications.

#### Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

##### 2.8.4 Firebase Fire-store

[Google Fire-store o](https://blog.back4app.com/what-is-firestore/)r Cloud Fire-store is a part of the Google Firebase app development platform. It is a cloud-hosted NoSQL database option for the storage and synchronization of data. Users can directly access Fire-store from their web and mobile applications with native SDKs.

Users can use it with programming languages such as Java, Unity, Node.js, Go, and C++ SDKs, and there is also support for RPC and REST APIs. Using the Fire-store database facilitates better performance, automatic scaling, peak reliability, and considerable ease-of-use.

Fire-store enables the syncing of data across various client applications through real-time listeners. It makes use of Access Management, and Cloud Identity features from Google for authentication purposes. Fire-store enables the storage of data as documents, and these documents are stored within collections.

**Fire-store Capabilities**

**Real-Time Updates** – Cloud Fire-store utilizes synchronization for updating data across connected devices. This capability also enables users to implement one-time fetch queries efficiently.

**Offline Synchronization** – Cloud Fire-store performs caching of data being used by an app to let it read, write, query, and listen to data even with an offline device.

**Data Structure** – Cloud Fire-store’s data model offers support for hierarchical data structures. Users can store their data as documents that are stored in collections. Documents contain complex nested objects and sub-collections.

**Expressive Querying** – Cloud Fire-store enables the use of queries for fetching specific individual documents or retrieving documents matching query parameters from a collection.

Queries include multiple chained filters and support sorting and filtering.

**Scalability** – Cloud Fire-store offers the scalability you expect from the Google Cloud Platform. Some of its capabilities include multi-regional replication, atomic batch operations, assurance of consistency, and support for real transactions.

##### 2.8.5 Firebase Storage

[Firebase](https://firebase.google.com/docs?gclid=EAIaIQobChMIooGMk52p6gIVTcayCh2nPgBBEAAYASACEgLYZ_D_BwE) is a platform for web and mobile application development. It was created by Firebase,

Inc in 2011 and acquired in 2014 by Google. Firebase Storage is storage incorporated into the Firebase platform. You can use it to store and manage media generated by web and mobile app users.

The [Firebase SDK](https://firebase.google.com/docs/firestore/client/libraries) lets developers to work with Firebase programmatically, including cloud storage functionality. It enables them to download and upload files from and to clients directly. It also includes features that enable transfers to be resumed or retried in cases of poor connectivity.

**How Does Cloud Storage Fit into the Firebase Architecture?**

The Firebase Architecture is made of several components, which work together to provide a framework and infrastructure for developing and hosting web and mobile applications.

* Firebase Cloud Storage—also called Firebase File Storage, is an [object storage](https://cloudian.com/blog/object-storage-care/) service offered on Google Cloud Platform. When [Google Cloud Storage](https://cloud.netapp.com/blog/object-storage-block-and-shared-file-storage-in-google-cloud) is incorporated into Firebase apps, you gain access to Google security measures and the ability to secure any uploads or downloads in your app. Through the SDK you can also manage your media and access it directly from your storage account. Integration is supported for Android, C++, iOS, Unity, and Web apps.
* Cloud Fire-store—a scalable, flexible database service for server, web, and mobile development. It serves as a NoSQL document database. You can use it to store, query, and sync your app data. It is supported for Android, iOS, and web applications.
* Authentication—included in the SDK in the form of UI libraries. You can use it as a backend service to authenticate app users via password, phone numbers, or federated ID. Federated options include Google, Twitter, and Facebook. It is supported for Android, iOS, and web applications.
* Hosting—provides production-grade hosting for your web content. With this component you can deploy apps and serve dynamic or static content via a global content delivery network (CDN). You can also combine the hosting component with Cloud Functions to develop and host micro-services. Only web applications are supported.

**CHAPTER - 3**

## 3. LITERATURE SURVEY

The first conducted by Athoillah and Irawan conducted on the design of Android-based mobile information systems to control inventory in the warehouse. With this, the need to store the record of goods manually is removed.

Rubianto et al. discussed the system of libraries by using traditional waterfall method in SMAN 8 Garut. This application described the modelling of library information systems by using traditional waterfall method. It mainly focussed on identifying problems, opportunities, and interest; determining requirements; analyzing system requirements; designing the system; developing and documenting the software; and using flow map or flowchart, Data Flow Diagrams (DFD), and Entity Relationship Diagram (ERD) for the model.

Hastanti et al. (2015) investigated the sales system based on the web (e-commerce) in Tata Distro in Pacitan. The results of the research were the creation of promoting media and sales-based online website, design, and manufacture to facilitate management, sales, promotions, and buyers.

The research focuses on designing procurement applications by using Android platform to facilitate the control of the procurement of goods on the markets. To design the application, researcher uses several stages by using waterfall method.

The method has several stages:

* System analysis
* Design
* Coding
* Implementation.

**CHAPTER - 4**

## SYSTEM DESIGN

### 4.1 Introduction to UML

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows:

1. User Model View

This view represents the system from the users’ perspective. The analysis representation describes a usage scenario from the end-users’ perspective.

1. Structural Model View

In this model, the data and functionality are arrived from inside the system. This model view models the static structures.

1. Behavioural Model View

It represents the dynamic of behavioural as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.

1. Implementation Model View

In this view, the structural and behavioural as parts of the system are represented as they are to be built.

1. Environmental Model View

In this view, the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

#### 4.2 Use Case Diagrams

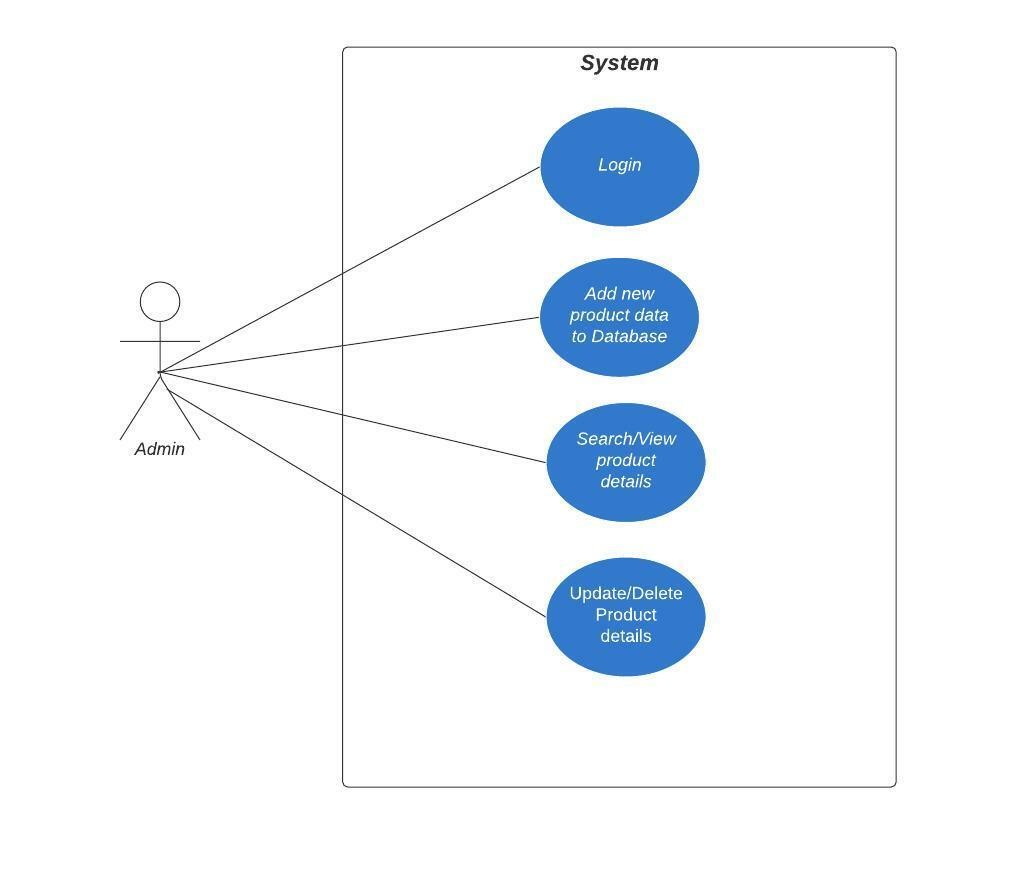
To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running or operating.

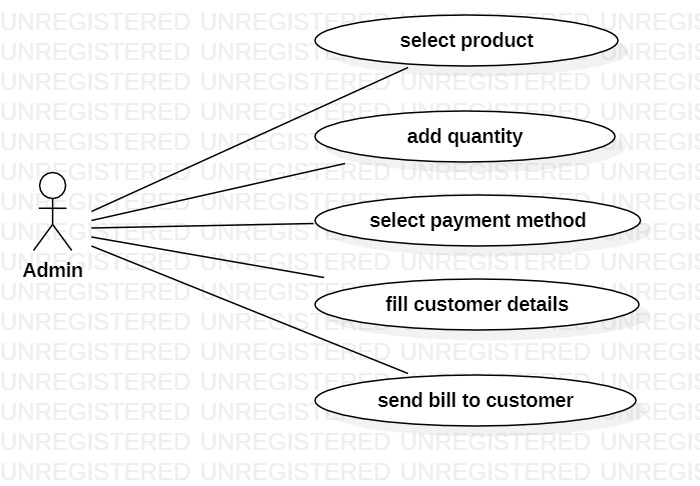
So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

1. Used to gather requirements of a system.
2. Used to get an outside view of a system.
3. Identify external and internal factors influencing the system.
4. Show the interacting among the requirements are actors.





**Fig 4.2 - Use Case Diagrams**

#### 4.3 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per se. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

**Basic Sequence Diagram Notations**

#### Class Roles or Participants

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

#### Activation or Execution Occurrence

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin grey rectangle placed vertically on its lifeline.

#### Messages

Messages are arrows that represent communication between objects. Use half arrowed lines to represent asynchronous messages.

Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

**Lifelines**

Lifelines are vertical dashed lines that indicate the object's presence over time.

#### Destroying Objects

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

#### Loops

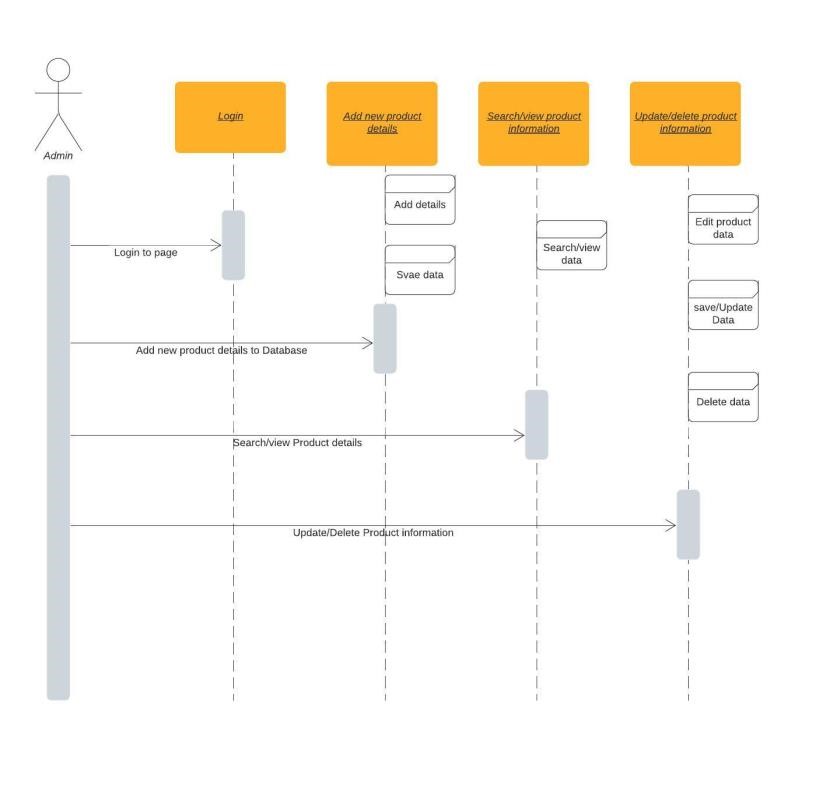
A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [].

Guards.

When modelling object interactions, there will be times when a condition must be

met for a message

e to be sent to an object.



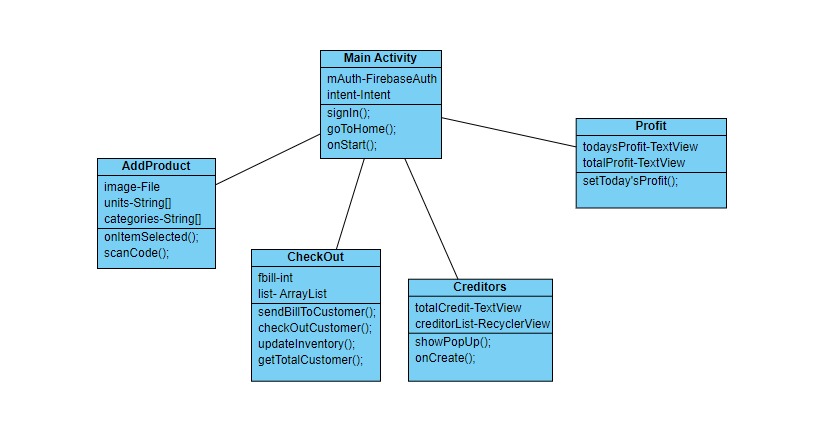
#### Fig 4.3– Sequence Diagram

##### 4.4 Class Diagram

Class diagrams are the main building blocks of every object-oriented method. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship in between them.

The main purpose to use class diagrams are:

1. This is the only UML which can appropriately depict various aspects of OOPs concept.
2. Proper design and analysis of application can be faster and efficient.



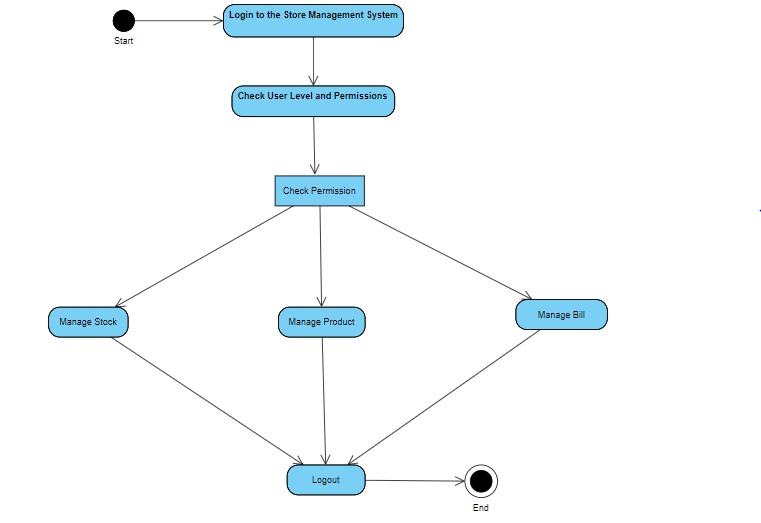
#### Fig 4.4 – Class Diagram

**4.5 Activity Diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

****

#### Fig 4.5 – Activity Diagram

**CHAPTER - 5**

## 5. IMPLEMENTATION

### 5.1 Login Module

package com.example.stockmanager.View;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Build;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.view.Window;

import android.view.WindowManager;

import android.widget.Toast;

import com.example.stockmanager.R;

import com.google.android.gms.auth.api.signin.GoogleSignIn;

import com.google.android.gms.auth.api.signin.GoogleSignInAccount;

import com.google.android.gms.auth.api.signin.GoogleSignInClient;

import com.google.android.gms.auth.api.signin.GoogleSignInOptions;

import com.google.android.gms.common.api.ApiException;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthCredential;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.auth.GoogleAuthProvider;

public class MainActivity extends AppCompatActivity implements ILogin {

com.google.android.gms.common.SignInButtonsignInButton;

GoogleSignInClientgoogleSignInClient;

FirebaseAuthmAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

signInButton=findViewById(R.id.sign\_in\_button);

if (Build.VERSION.SDK\_INT>= 21) {

Window window = this.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG\_DRAWS\_SYSTEM\_BAR\_BAC

KGROUNDS);

window.clearFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);

window.setStatusBarColor(this.getResources().getColor(R.color.black));

}

GoogleSignInOptionsgso = new GoogleSignInOptions

.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)

.requestIdToken("774151898490-

gfgtb4a961qoo5vqgeg33t2cj1ekf6kl.apps.googleusercontent.com")

.requestEmail()

.build();

mAuth = FirebaseAuth.getInstance();

googleSignInClient= GoogleSignIn.getClient(this, gso);

signInButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

SignIn();

}

});

}

@Override

public void SignIn() {

Intent signInIntent = googleSignInClient.getSignInIntent();

startActivityForResult(signInIntent, 1);

}

@Override

public void GoToHome(FirebaseUser user) {

if(user!=null){

Intent i= new Intent(MainActivity.this, Home.class);

startActivity(i);

}else{

Toast.makeText(getApplicationContext(),"No Login

account",Toast.LENGTH\_LONG).show();

}}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == 1) {

Task<GoogleSignInAccount> task =

GoogleSignIn.getSignedInAccountFromIntent(data);

try {

GoogleSignInAccount account = (GoogleSignInAccount) ((Task<?>)

task).getResult(ApiException.class);

Log.d("Login", "firebaseAuthWithGoogle:" + account.getId());

firebaseAuthWithGoogle(account.getIdToken());

} catch (ApiException e) {

Log.w("Login", "Google sign in failed", e);

} } }

private void firebaseAuthWithGoogle(String idToken) {

AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);

mAuth.signInWithCredential(credential)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

// Sign in success, update UI with the signed-in user's information

Log.d("Login", "signInWithCredential:success");

FirebaseUser user = mAuth.getCurrentUser();

GoToHome(user);

} else {

// If sign in fails, display a message to the user.

Log.w("Login", "signInWithCredential:failure", task.getException());

FirebaseUser user = mAuth.getCurrentUser();

GoToHome(user);

}}

});

}

@Override

protected void onStart() {

super.onStart();

if(FirebaseAuth.getInstance().getCurrentUser()!=null){

GoToHome(FirebaseAuth.getInstance().getCurrentUser());

finish();

} } }

**5.2 Home Page**

package com.example.stockmanager.View;

import androidx.annotation.NonNull;

import androidx.annotation.RequiresApi;

import androidx.appcompat.app.AppCompatActivity;

import androidx.fragment.app.Fragment;

import androidx.fragment.app.FragmentManager;

import android.os.Build;

import android.os.Bundle;

import android.view.MenuItem;

import android.view.Window;

import android.view.WindowManager;

import android.widget.FrameLayout;

import android.widget.Toast;

import com.example.stockmanager.Controller.FirebaseUtils;

import com.example.stockmanager.R;

import com.google.android.material.bottomnavigation.BottomNavigationView;

public class Home extends AppCompatActivityimplements

BottomNavigationView.OnNavigationItemSelectedListener {

BottomNavigationViewbottomNavigationView;

Fragment itemFragment;

Fragment billingFragment;

final FragmentManagerfm = getSupportFragmentManager();

Fragment active ;

FrameLayout frame;

@RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_home);

if (Build.VERSION.SDK\_INT>= 21) {

Window window = this.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG\_DRAWS\_SYSTEM\_BAR\_BAC

KGROUNDS);

window.clearFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);

window.setStatusBarColor(this.getResources().getColor(R.color.black));

}

itemFragment = new GoodsFragment(this);

billingFragment = new BillingFragment(Home.this);

bottomNavigationView = (BottomNavigationView) findViewById(R.id.bottom\_navigation);

fm.beginTransaction().add(R.id.frame, billingFragment,

"2").hide(billingFragment).commit();

fm.beginTransaction().add(R.id.frame, itemFragment, "1").commit();

active=itemFragment;

bottomNavigationView.setOnNavigationItemSelectedListener(Home.this);

frame = findViewById(R.id.frame);

Toast.makeText(getApplicationContext(), new

FirebaseUtils().getCurrentUser().getDisplayName(), Toast.LENGTH\_LONG).show();

}

@Override

public booleanonNavigationItemSelected(@NonNull MenuItemmenuItem) {

switch (menuItem.getItemId()) {

case R.id.goods:

fm.beginTransaction().hide(active).show(itemFragment).commit();

active = itemFragment;

return true;

case R.id.billing:

fm.beginTransaction().hide(active).show(billingFragment).commit();

active = billingFragment;

return true;

}

return false;

}

}

**5.3 Goods Fragment**

package com.example.stockmanager.View; import android.annotation.SuppressLint; import android.content.Context;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.widget.SearchView;

import androidx.fragment.app.Fragment;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import com.example.stockmanager.Controller.FirebaseUtils;

import com.example.stockmanager.GroupViewAdapter;

import com.example.stockmanager.Model.Product;

import com.example.stockmanager.R;

import com.google.android.material.bottomsheet.BottomSheetDialog;

import com.google.android.material.floatingactionbutton.ExtendedFloatingActionButton; import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import java.util.ArrayList;

import java.util.Map;

import java.util.TreeMap;

public class GoodsFragment extends Fragment implements ItemFragmentInterface {

ImageViewprofile,menu;

FirebaseUser user;

RecyclerViewrecyclerView;

Context context;

View view;

Spinner filter;

GroupViewAdaptergroupViewAdapter;

ArrayList<Product>list;

SearchViewsearchView;

ExtendedFloatingActionButton add;

String categories[];

private static final String ARG\_COLUMN\_COUNT = "column-count"; private int mColumnCount = 1;

public GoodsFragment(){}

public GoodsFragment(Context context) {

this.context=context;

}

@Override

public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

if (getArguments() != null) {

mColumnCount = getArguments().getInt(ARG\_COLUMN\_COUNT);

}

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

user = FirebaseAuth.getInstance().getCurrentUser();

view = inflater.inflate(R.layout.goods\_fragment, container, false); add=view.findViewById(R.id.addnew); filter=view.findViewById(R.id.filter); profile=view.findViewById(R.id.profile); searchView=view.findViewById(R.id.searchView); menu=view.findViewById(R.id.menu);

categories=context.getResources().getStringArray(R.array.cat\_array); Glide.with(view.getContext()).load(user.getPhotoUrl()).into(profile); recyclerView=view.findViewById(R.id.list);

recyclerView.setLayoutManager(newLinearLayoutManager(context)); menu.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) { showBottomSheetDialog();

}

});

filter.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> parent, View view, int position, long id)

{

Toast.makeText(getContext(),categories[position],Toast.LENGTH\_LONG).show();

new

FirebaseUtils(GoodsFragment.this,categories).getFilteredCategories(categories[position]);

}

@Override

public void onNothingSelected(AdapterView<?> parent) {

}

});

final ArrayAdapter<String>spinnerArrayAdapter = new

ArrayAdapter<String>(this.getContext(),R.layout.spinner\_item,categories){

@Override

public View getDropDownView(int position, View convertView,

ViewGroup parent) {

View view = super.getDropDownView(position, convertView, parent);

TextView tv = (TextView) view;

if(position%2 == 1) {

tv.setBackgroundColor(Color.parseColor("#FFF9A600"));

tv.setTextColor(Color.parseColor("#f8f6f2"));

}

else {

tv.setBackgroundColor(Color.parseColor("#FFE49200"));

tv.setTextColor(Color.parseColor("#f8f6f2"));

}

return view;

}

};

filter.setPrompt("Select an item");

spinnerArrayAdapter.setDropDownViewResource(R.layout.spinner\_item);

filter.setAdapter(spinnerArrayAdapter);

list=new ArrayList<>();

add.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent i=new Intent(GoodsFragment.this.getContext(),AddProduct.class);

startActivity(i);

}

});

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

@Override

public booleanonQueryTextSubmit(String query) {

filter(query);

return false;

}

@Override

public booleanonQueryTextChange(String newText) {

filter(newText);

return false;

}

});

return view;

}

private void filter(String newText) {

TreeMap<String,ArrayList<Product>>map=new TreeMap<>();

TreeMap<Integer,ArrayList<Product>>inventory=new TreeMap<>();

int i=0;

for (Product product:

list) {

if(product.getPRODUCT\_NAME().toLowerCase().contains(newText))

{ if(!map.containsKey(product.getCATEGORY()))map.put(product.getCATEGORY(),new

ArrayList<Product>());

map.get(product.getCATEGORY()).add(product);

}

}

if(map.size()==0){

Toast.makeText(getContext(),"No Data Found",Toast.LENGTH\_LONG).show();

}else{

for (Map.Entry<String,ArrayList<Product>> e:

map.entrySet()) {

if(e.getValue().size()>0)

inventory.put(i++,e.getValue());

}

groupViewAdapter.filterList(inventory);

}

}

@Override

public void setRecyclerView(TreeMap<Integer, ArrayList<Product>> list) {

if(list.size()==0){

Toast.makeText(getContext(),"No Products Avalable",Toast.LENGTH\_LONG).show();

}else{

this.list.clear();

groupViewAdapter=new GroupViewAdapter(list,view.getContext());

recyclerView.setAdapter(groupViewAdapter);

for (int i = 0; i<list.size(); i++) {

this.list.addAll(list.get(i));

}

}

}

@Override

public void noDataAvailableUnderCategory(String category) {

Toast.makeText(context,"No products avaialable in "+category,

Toast.LENGTH\_LONG).show();

}

private void showBottomSheetDialog() {

LinearLayout logout;

final BottomSheetDialogbottomSheetDialog = new

BottomSheetDialog(GoodsFragment.this.getContext());

bottomSheetDialog.setContentView(R.layout.mainmenu);

logout=bottomSheetDialog.findViewById(R.id.logout\_layout);

logout.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

FirebaseAuth.getInstance().signOut();

Toast.makeText(context,"Logged Out",Toast.LENGTH\_LONG).show();

Intent i = new Intent(context,MainActivity.class);

startActivity(i);

getActivity().finish();

}

});

bottomSheetDialog.show();

}

}

**5.4 Billing Fragment**

package com.example.stockmanager.View;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.constraintlayout.widget.ConstratLayout;

import androidx.fragment.app.Fragment;

importandroidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import com.example.stockmanager.Controller.FirebaseUtils;

import com.example.stockmanager.R;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.mikhaellopez.circularimageview.CircularImageView;

public class BillingFragment extends Fragment {

private static final String ARG\_PARAM1 = "param1";

Context context;

private static final String ARG\_PARAM2 = "param2";

CircularImageView profile;

ImageViewtransactionnext;

FirebaseUser user;

RecyclerViewtransactionlist;

TextViewtransactiontext, more, salestoday, totalsales, see\_profits;

ConstraintLayout billing, expandable, transaction, creditors;

public BillingFragment() {

}

public BillingFragment(Context context) {

this.context = context;

}

public static BillingFragmentnewInstance(String param1, String param2) {

BillingFragment fragment = new BillingFragment();

return fragment;

}

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

user= FirebaseAuth.getInstance().getCurrentUser();

View view = inflater.inflate(R.layout.fragment\_billing\_fragment, container, false);

profile = view.findViewById(R.id.profile);

billing = view.findViewById(R.id.billinglayout);

transactiontext = view.findViewById(R.id.ttext);

transaction = view.findViewById(R.id.transactionslayout);

expandable = view.findViewById(R.id.expandable);

creditors = view.findViewById(R.id.creditlayout);

salestoday = view.findViewById(R.id.salestoday);

totalsales = view.findViewById(R.id.totalsales);

see\_profits = view.findViewById(R.id.seeprofits);

more = view.findViewById(R.id.more);

Glide.with(context).load(user.getPhotoUrl()).into(profile);

transactiontext.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(context,"Under development",Toast.LENGTH\_LONG).show();

}

});

see\_profits.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(context,"Under development",Toast.LENGTH\_LONG).show();

}

});

more.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(context,"Under development",Toast.LENGTH\_LONG).show();

}

});

creditors.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(context,"Under development",Toast.LENGTH\_LONG).show();

}

});

billing.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(context,"Under development",Toast.LENGTH\_LONG).show();

}

});

return view;

}

}

**5.5 Add Product**

package com.example.stockmanager.View;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.annotation.RequiresApi;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.content.FileProvider;

import android.Manifest;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.app.DatePickerDialog;

import android.app.ProgressDialog;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import android.os.Environment;

import android.provider.MediaStore;

import android.view.View;

import android.view.Window;

import android.view.WindowManager;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;

import com.example.stockmanager.Model.Product;

import com.example.stockmanager.R;

import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.android.material.bottomsheet.BottomSheetDialog;

import com.google.android.material.floatingactionbutton.ExtendedFloatingActionButton;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.storage.FirebaseStorage;

import com.google.firebase.storage.StorageReference;

import com.google.firebase.storage.UploadTask;

import com.google.zxing.integration.android.IntentIntegratr;

importcom.google.zxing.integration.android.IntentRl;

import java.io.File;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

public class AddProduct extends AppCompatActivity implements

AdapterView.OnItemSelectedListener {

private static final int RESULT\_LOAD\_IMAGE =2;

Spinner spin,cat;

ImageViewback,calendar,scanner,img;

TextViewmfd,code;

File image;

Uri selectedImage;

FirebaseFirestoredb;

ExtendedFloatingActionButtonaddtostore;

String unit,categorycheck;

android.content.res.Resources res;

EditTextproduct,stock,sp,cp,months;

private static final int MY\_CAMERA\_PERMISSION\_CODE = 1;

private static final int CAMERA\_REQUEST = 1;

private static final int REQUEST\_IMAGE\_CAPTURE = 1;

String currentPhotoPath="none";

String[] units ;

String[] categories ;

String[] spincategories ;

@SuppressLint("WrongConstant")

@RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// getWindow().setEnterTransition(new Slide(Gravity.BOTTOM));

setContentView(R.layout.activity\_add\_product);

Window window = this.getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG\_DRAWS\_SYSTEM\_BAR\_BAC

KGROUNDS);

window.clearFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);

window.setStatusBarColor(this.getResources().getColor(R.color.black));

res=getResources();

units = res.getStringArray(R.array.units\_array);

categories =res.getStringArray(R.array.cat\_array);

spincategories=res.getStringArray(R.array.cat\_array);

spincategories[0]="Select Category";

scanner=findViewById(R.id.bcr);

img=findViewById(R.id.circularImageView);

mfd=findViewById(R.id.mfd);

calendar=findViewById(R.id.calendar);

spin = findViewById(R.id.units);

cat=findViewById(R.id.categories);

back=findViewById(R.id.back);

code=findViewById(R.id.code);

product=findViewById(R.id.productName);

cp=findViewById(R.id.cp);

sp=findViewById(R.id.sp);

stock=findViewById(R.id.stock);

addtostore=findViewById(R.id.addtostore);

months=findViewById(R.id.months);

scanner.setOnClickListener(view ->scanCode());

addtostore.setOnClickListener(new View.OnClickListener() {

@RequiresApi(api = Build.VERSION\_CODES.O)

@Override

public void onClick(View v) {

try {

adddItemToInventory();

} catch (ParseException e) {

e.printStackTrace();

}

}

});

calendar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Calendar myCalendar = Calendar.getInstance();

DatePickerDialog.OnDateSetListener date = new DatePickerDialog.OnDateSetListener() {

@RequiresApi(api = Build.VERSION\_CODES.N)

@Override

public void onDateSet(DatePicker view, int year, int monthOfYear,

int dayOfMonth) {

// TODO Auto-generated method stub

myCalendar.set(Calendar.YEAR, year);

myCalendar.set(Calendar.MONTH, monthOfYear);

myCalendar.set(Calendar.DAY\_OF\_MONTH, dayOfMonth);

String myFormat = "yyyy-MM-dd"; //In which you need put here

SimpleDateFormatsdf = new SimpleDateFormat(myFormat, Locale.US);

mfd.setText(sdf.format(myCalendar.getTime())+" ");

}

};

new DatePickerDialog(AddProduct.this, date, myCalendar

.get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),

myCalendar.get(Calendar.DAY\_OF\_MONTH)).show();

}

});

back.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

finish();

}

});

cat.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> parent, View view, int position, long id)

{

//

Toast.makeText(getApplicationContext(),categories[position],Toast.LENGTH\_LONG).show

();

categorycheck=categories[position];

}

@Override

public void onNothingSelected(AdapterView<?> parent) {

}

});

img.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showBottomSheetDialog();

}

});

spin.setOnItemSelectedListener(AddProduct.this);

ArrayAdapter ad= new ArrayAdapter(this,android.R.layout.simple\_spinner\_item,units);

ArrayAdapter ad1= new

ArrayAdapter(this,android.R.layout.simple\_spinner\_item,spincategories);

ad.setDropDownViewResource( android.R.layout.simple\_spinner\_dropdown\_item);

ad1.setDropDownViewResource( android.R.layout.simple\_spinner\_dropdown\_item);

cat.setAdapter(ad1);

spin.setAdapter(ad);

}

@Override

public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

//

Toast.makeText(getApplicationContext(),units[position],Toast.LENGTH\_LONG).show();

unit=units[position];

}

@Override

public void onNothingSelected(AdapterView<?> parent) {

Toast.makeText(getApplicationContext(),"No Change",Toast.LENGTH\_LONG).show();

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

//super.onActivityResult(requestCode, resultCode, data);

if (requestCode == RESULT\_LOAD\_IMAGE &&resultCode == RESULT\_OK &&null

!= data) {

selectedImage = data.getData(); currentPhotoPath="not\_none";

img.setImageURI(selectedImage);

}else

if (requestCode == CAMERA\_REQUEST &&resultCode == Activity.RESULT\_OK)

{

image= new File(currentPhotoPath);

img.setImageURI(Uri.fromFile(image));

selectedImage=Uri.fromFile(image);

} else{

IntentResult result= IntentIntegrator.parseActivityResult(requestCode, resultCode, data);

if(result!=null){

if(result.getContents()!=null){

code.setText(result.getContents());

}else{

Toast.makeText(this,"No Result",Toast.LENGTH\_LONG).show();

}

}else{

super.onActivityResult(requestCode, resultCode, data);

}

}

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,

@NonNull int[] grantResults)

{

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == MY\_CAMERA\_PERMISSION\_CODE)

{

if (grantResults[0] == PackageManager.PERMISSION\_GRANTED)

{

Toast.makeText(this, "camera permission granted", Toast.LENGTH\_LONG).show();

Intent cameraIntent = new

Intent(android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE);

startActivityForResult(cameraIntent, CAMERA\_REQUEST);

}

else

{

Toast.makeText(this, "camera permission denied", Toast.LENGTH\_LONG).show();

}

}

}

private File createImageFile() throws IOException {

// Create an image file name

String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss").format(new

Date());

String imageFileName = "JPEG\_" + timeStamp + "\_";

File storageDir = getExternalFilesDir(Environment.DIRECTORY\_PICTURES);

File image = File.createTempFile(

imageFileName, /\* prefix \*/

".jpg", /\* suffix \*/

storageDir /\* directory \*/

);

// Save a file: path for use with ACTION\_VIEW intents

currentPhotoPath = image.getAbsolutePath();

return image;

}

private void dispatchTakePictureIntent() {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

// Ensure that there's a camera activity to handle the intent

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

// Create the File where the photo should go

File photoFile = null;

try {

photoFile = createImageFile();

} catch (IOException ex) {

// Error occurred while creating the File

}

// Continue only if the File was successfully created

if (photoFile != null) {

Uri photoURI = FileProvider.getUriForFile(this,

"com.example.stockmanager.provider",

photoFile);

takePictureIntent.putExtra(MediaStore.EXTRA\_OUTPUT, photoURI);

startActivityForResult(takePictureIntent, REQUEST\_IMAGE\_CAPTURE);

}

}

}

private void scanCode() {

IntentIntegrator intent= new IntentIntegrator(this);

intent.setCaptureActivity(capture.class);

intent.setOrientationLocked(false);

intent.setDesiredBarcodeFormats(IntentIntegrator.ALL\_CODE\_TYPES);

intent.setPrompt("Cover Entire Barcode while Scanning Code");

intent.initiateScan();

}

private void showBottomSheetDialog() {

final BottomSheetDialogbottomSheetDialog = new BottomSheetDialog(this);

bottomSheetDialog.setContentView(R.layout.bottom\_sheet\_layout);

LinearLayout photo = bottomSheetDialog.findViewById(R.id.takephoto);

LinearLayout gallery = bottomSheetDialog.findViewById(R.id.gallery);

photo.setOnClickListener(new View.OnClickListener() {

@RequiresApi(api = Build.VERSION\_CODES.M)

@Override

public void onClick(View v) {

if (checkSelfPermission(Manifest.permission.CAMERA) !=

PackageManager.PERMISSION\_GRANTED

||checkSelfPermission(Manifest.permission.WRITE\_EXTERNAL\_STORAGE)!=PackageMa

nager.PERMISSION\_GRANTED)

{

requestPermissions(new

String[]{Manifest.permission.CAMERA,Manifest.permission.WRITE\_EXTERNAL\_STOR

AGE}, MY\_CAMERA\_PERMISSION\_CODE);

}

else

{

dispatchTakePictureIntent();

bottomSheetDialog.cancel();

}

}

});

gallery.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent i = new Intent(

Intent.ACTION\_PICK,

android.provider.MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(i, RESULT\_LOAD\_IMAGE);

bottomSheetDialog.cancel();

}

});

bottomSheetDialog.show();

}

@RequiresApi(api = Build.VERSION\_CODES.O)

private void adddItemToInventory() throws ParseException {

ProgressDialogprogressDialog=new ProgressDialog(this);

progressDialog.setTitle(R.string.progressTitle);

progressDialog.setCancelable(false);

progressDialog.show();

final String Product\_Name=product.getText().toString();

final String Cost\_Price=cp.getText().toString();

final String Selling\_Price=sp.getText().toString();

final String Barcode=code.getText().toString();

final String Quantity=stock.getText().toString();

String Manufacturing\_date=mfd.getText().toString().trim();

String BestBefore=months.getText().toString();

final Product[] p = new Product[1];

final StorageReferencestorageReference= FirebaseStorage.getInstance().getReference();

if(Product\_Name.equals("")){

Toast.makeText(this,"Enter Product name",Toast.LENGTH\_LONG).show();

product.requestFocus();

progressDialog.cancel();

}else if(Cost\_Price.equals("")){

Toast.makeText(this,"Enter Cost Price",Toast.LENGTH\_LONG).show();

cp.requestFocus();

progressDialog.cancel();

}else if(Selling\_Price.equals("")){

Toast.makeText(this,"Enter Selling Price",Toast.LENGTH\_LONG).show();

sp.requestFocus();

progressDialog.cancel();

}else if(BestBefore.equals("")) {

Toast.makeText(this, "Enter Best Before", Toast.LENGTH\_LONG).show();

months.requestFocus();

progressDialog.cancel();

}else if(Quantity.equals("")){

Toast.makeText(this,"Enter Stock value",Toast.LENGTH\_LONG).show();

stock.requestFocus();

progressDialog.cancel();

}else if(Manufacturing\_date.equals("Manufacturing Date : (YYYY/MM/DD)")){

Toast.makeText(this,"Select Manufacturing Date",Toast.LENGTH\_LONG).show();

mfd.requestFocus();

progressDialog.cancel();

}else if(Barcode.equals("")){

Toast.makeText(this,"Enter or Scan Barcode",Toast.LENGTH\_LONG).show();

code.requestFocus();

progressDialog.cancel();

}else if(unit.equals("Select Units")){

Toast.makeText(this,"Select Valid Units",Toast.LENGTH\_LONG).show();

spin.requestFocus();

progressDialog.cancel();

}else if(categorycheck.equals("Select Category")){

Toast.makeText(this,"Select Valid Category",Toast.LENGTH\_LONG).show();

cat.requestFocus();

progressDialog.cancel();

}else if(currentPhotoPath.equals("none")){

Toast.makeText(this,"Select Product Image",Toast.LENGTH\_LONG).show();

progressDialog.cancel();

img.requestFocus();

}

else {

progressDialog.setMessage("Uploading Image");

LocalDate date= LocalDate.parse(Manufacturing\_date);

LocalDatereturnvalue = date.plusMonths(Integer.parseInt(BestBefore));

String Expiry=returnvalue.toString();

storageReference.child(Barcode).putFile(selectedImage).addOnSuccessListener(new

OnSuccessListener<UploadTask.TaskSnapshot>() {

@Override

public void onSuccess(UploadTask.TaskSnapshottaskSnapshot) {

progressDialog.setMessage("Uploading Product Data");

Toast.makeText(AddProduct.this, "Image Uploaded", Toast.LENGTH\_SHORT).show();

storageReference.child(Barcode).getDownloadUrl().addOnSuccessListener(new

OnSuccessListener<Uri>() {

@Override

public void onSuccess(Uri uri) {

// Got the download URL for 'users/me/profile.png'

p[0] =new

Product(Product\_Name,Cost\_Price,Selling\_Price,Integer.parseInt(Quantity),Barcode,

String.valueOf(uri),unit,Manufacturing\_date,Expiry,categorycheck);

db= FirebaseFirestore.getInstance();

db.collection("Inventory")

.document(Barcode)

.set(p[0])

.addOnSuccessListener(new OnSuccessListener<Void>() {

@Override

public void onSuccess(Void aVoid) {

Toast.makeText(AddProduct.this,Product\_Name+" Added

Succcessfully",Toast.LENGTH\_LONG).show();

progressDialog.cancel();

startActivity(getIntent());

finish();

}

})

.addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(AddProduct.this,"Failed to add new

product",Toast.LENGTH\_LONG).show();

progressDialog.cancel();

}

});

}

});

}

}).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

System.out.println("Error Due to: "+e.getMessage());

Toast.makeText(AddProduct.this, "Cancelled", Toast.LENGTH\_SHORT).show();

}

});

Toast.makeText(this,"All details are Filled",Toast.LENGTH\_LONG).show();

}

}

}

**5.6**

**Database Functions**

package com.example.stockmanager.Controller;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import com.example.stockmanager.View.ItemFragmentInterface; import com.example.stockmanager.Model.Product; import com.google.android.gms.tasks.OnCompleteListener; import com.google.android.gms.tasks.Task; import com.google.firebase.auth.FirebaseAuth; import com.google.firebase.auth.FirebaseUser; import com.google.firebase.firestore.DocumentSnapshot; import com.google.firebase.firestore.EventListener; import com.google.firebase.firestore.FirebaseFirestore; import com.google.firebase.firestore.FirebaseFirestoreException; import com.google.firebase.firestore.Query; import com.google.firebase.firestore.QueryDocumentSnapshot; import com.google.firebase.firestore.QuerySnapshot; import org.jetbrains.annotations.NotNull;

import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date; import java.util.Map; import java.util.TreeMap;

public class FirebaseUtils implements FirebaseOperations {

FirebaseFirestoredb;

TreeMap<Integer,ArrayList<Product>>ProductList=new TreeMap<>();

TreeMap<String,ArrayList<Product>>Cat=new TreeMap<>();

String collectionref="Inventory";

String[] Categories;

ItemFragmentInterfaceitemFragmentInterface;

public FirebaseUtils(){};

public FirebaseUtils(ItemFragmentInterfaceitemFragmentInterface,String []Categories){

this.itemFragmentInterface=itemFragmentInterface;

this.Categories=Categories;

}

@Override

public FirebaseUsergetCurrentUser() {

return FirebaseAuth.getInstance().getCurrentUser();

}

int i=0;

@Override

public ArrayList<Product>getProducts() {

ArrayList<Product>list=new ArrayList<>();

db=FirebaseFirestore.getInstance();

db.collection(collectionref).addSnapshotListener(new EventListener<QuerySnapshot>() {

@Override

public void onEvent(@Nullable @org.jetbrains.annotations.Nullable QuerySnapshot

value, @Nullable @org.jetbrains.annotations.Nullable FirebaseFirestoreException error) {

ProductList.clear();

Cat.clear();

for (String cat:

Categories) {

Cat.put(cat,newArrayList<Product>());

}

db.collection(collectionref)

.get()

.addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {

@Override

public void onComplete(@NonNull @NotNull Task<QuerySnapshot> task)

{

int i=0;

for (QueryDocumentSnapshot document:

task.getResult()) {

Product p=document.toObject(Product.class);

System.out.println(p);

Cat.get(p.getCATEGORY()).add(p);

}

System.out.println(Cat);

for (Map.Entry<String,ArrayList<Product>> e:

Cat.entrySet()) {

if(e.getValue().size()>0)

ProductList.put(i++,e.getValue());

}

itemFragmentInterface.setRecyclerView(ProductList);

}

});

}

});

return list;

}

@Override

public void getFilteredCategories(String category) {

if(category.equalsIgnoreCase("All"))

{

getProducts();

return;

}

ProductList.clear();

ArrayList<Product>list= new ArrayList<>();

db=FirebaseFirestore.getInstance();

String finalCategory = category;

db.collection(collectionref)

.whereEqualTo("category",category)

.get()

.addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {

@Override

public void onComplete(@NonNull @NotNull Task<QuerySnapshot> task) {

int i=0;

for (QueryDocumentSnapshot document:

task.getResult()) {

list.add(document.toObject(Product.class));

}

if(list.size()!=0){

ProductList.put(0,list);

itemFragmentInterface.setRecyclerView(ProductList);

}

else

itemFragmentInterface.noDataAvailableUnderCategory(finalCategory);

}});}}

**CHAPTER - 6**

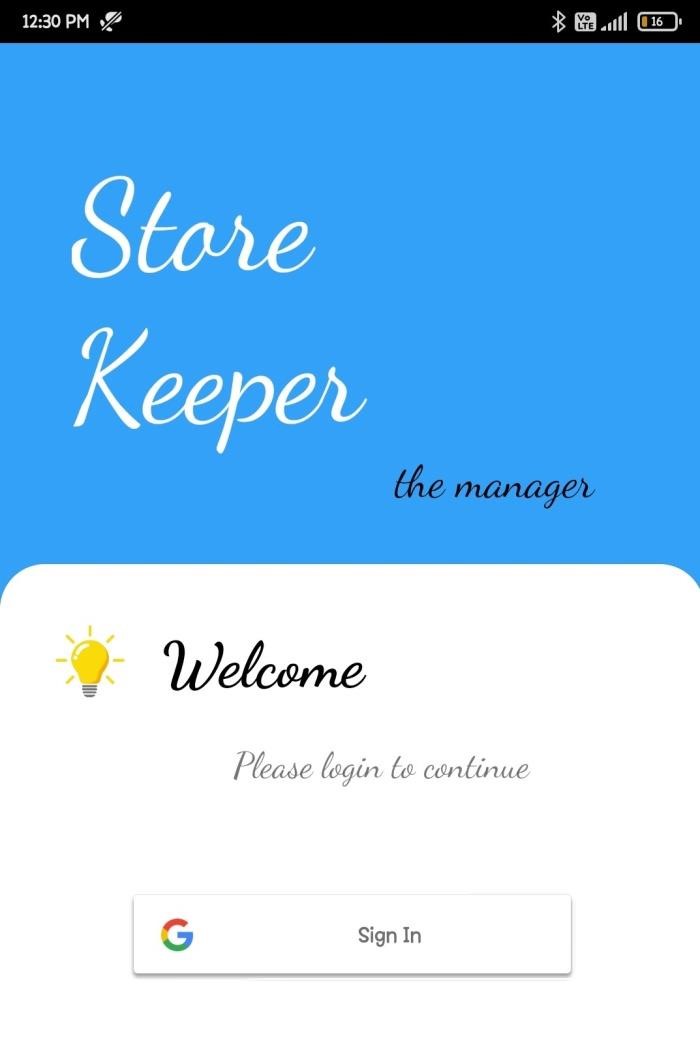
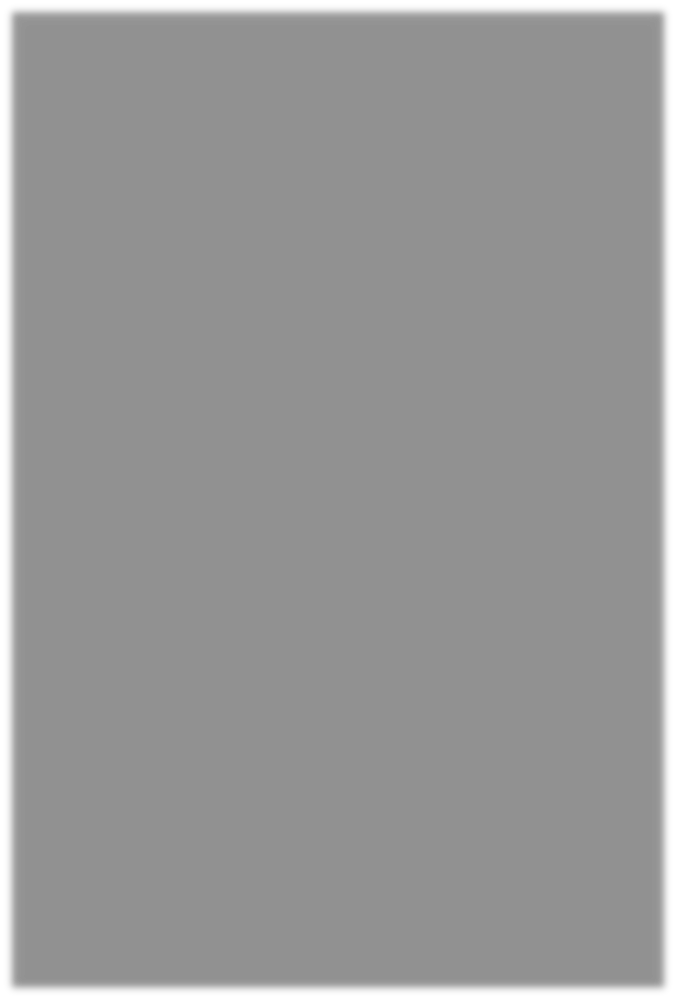
## 6. TEST CASES

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | INPUT | IF AVAILABLE | IF NOT AVAILABLE |
| 1 | Login | Admin can login with gmail id and password | There is no process |
| 2 | Add Product | Admin can add the product with its actual description and picture | There is no process |
| 3 | Add to Cart | Admin must add at least one product | There is no process |
| 4 | Checkout | Admin must select any one of the payment methods | There is no process |
| 5 | Billing | Admin must add valid details of the customer | There is no process |

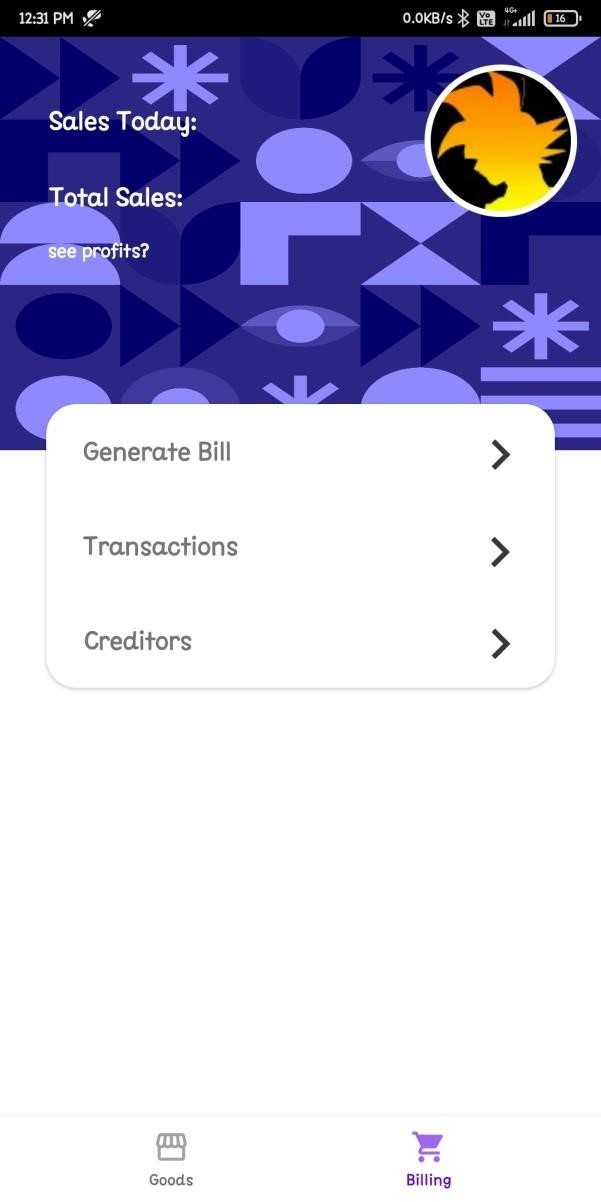
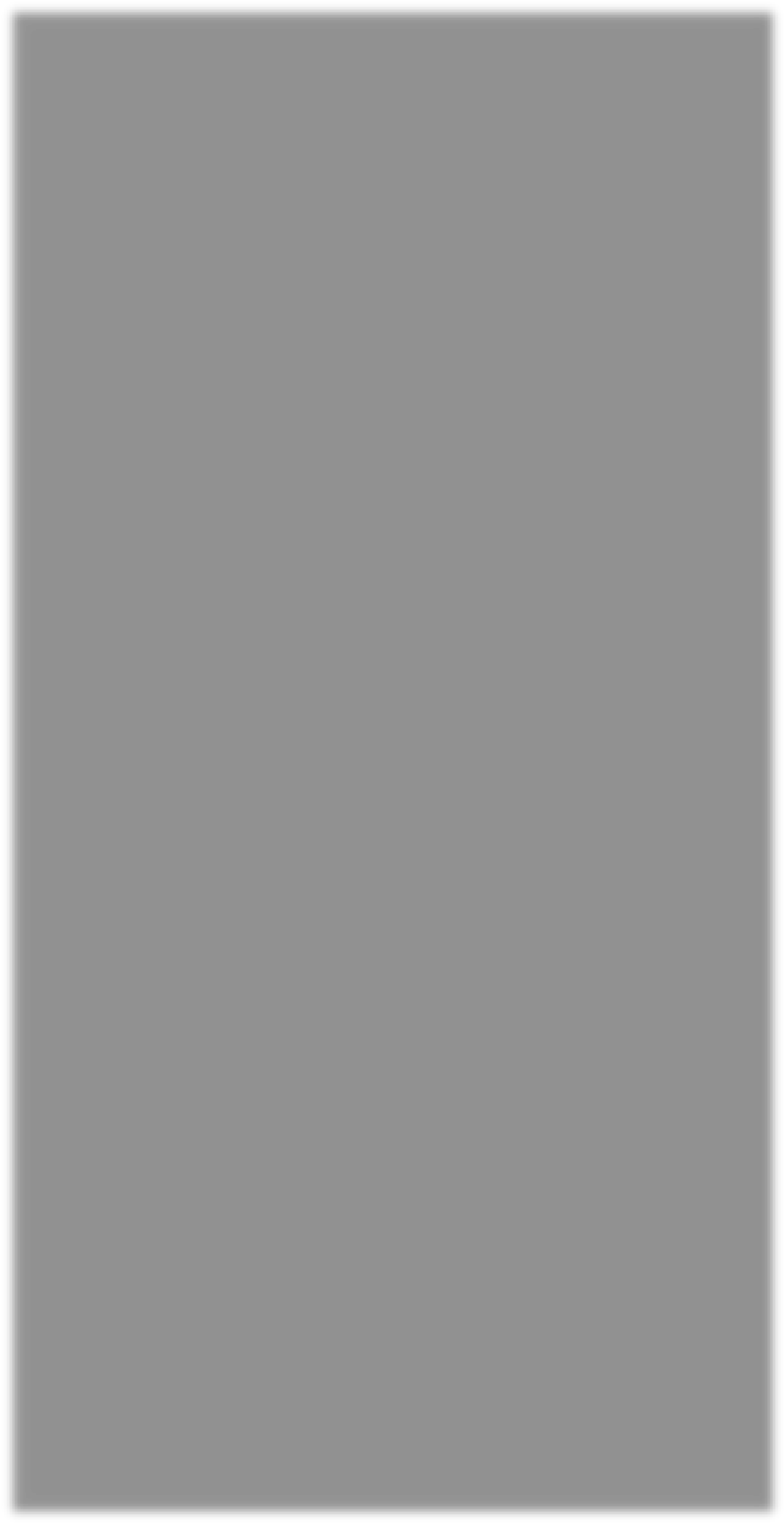
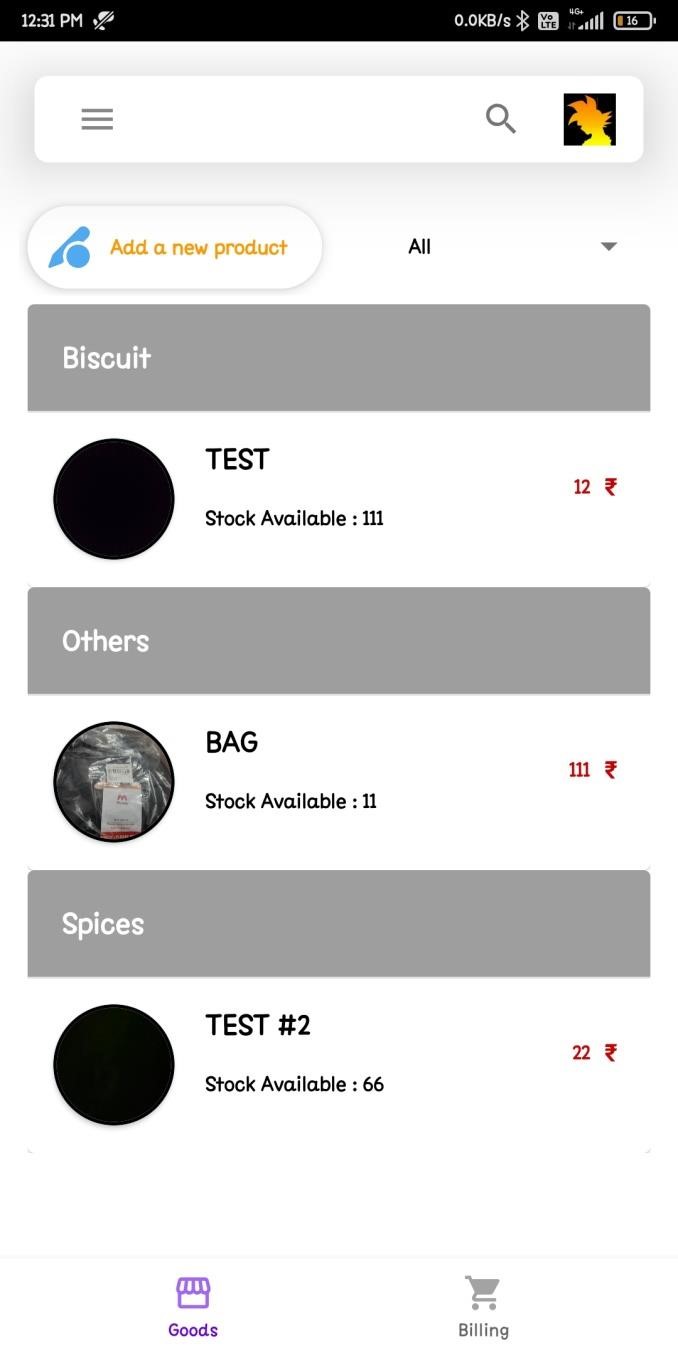
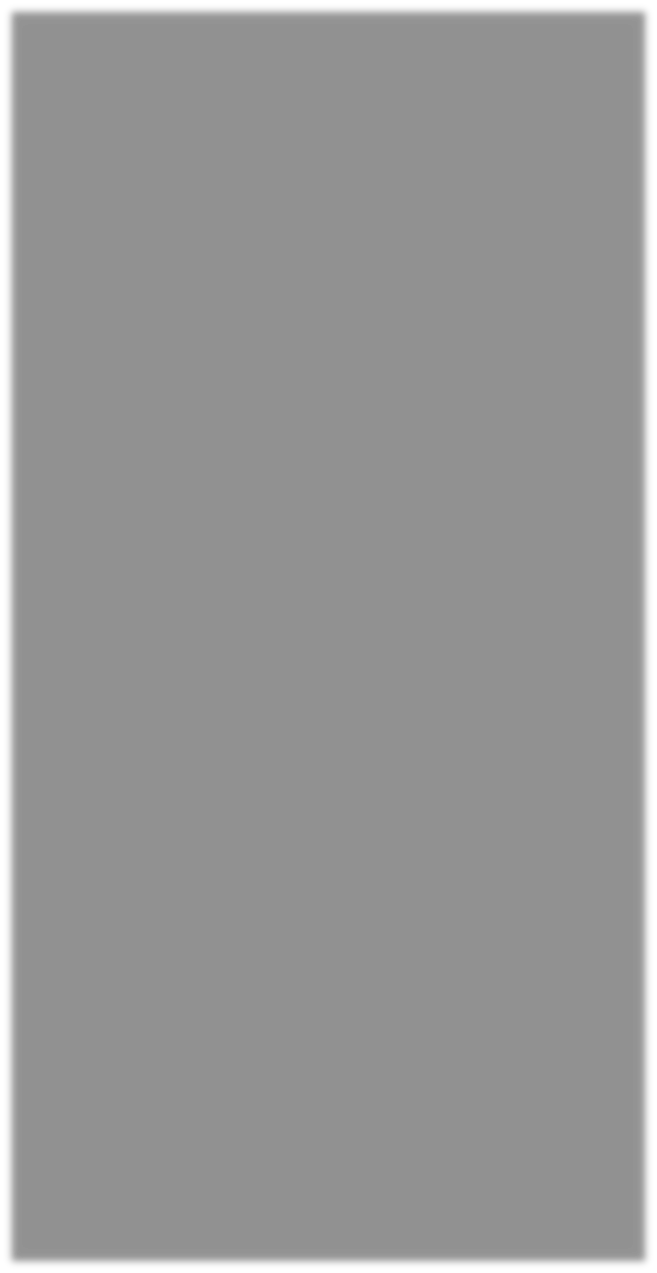
## CHAPTER - 7

## 7. SCREENSHOTS

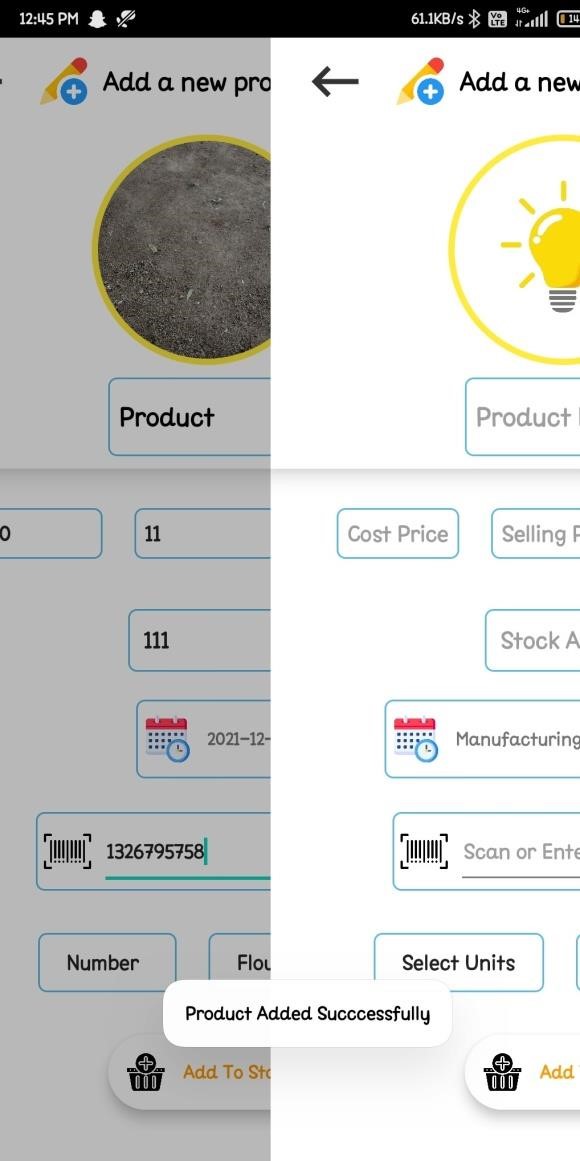
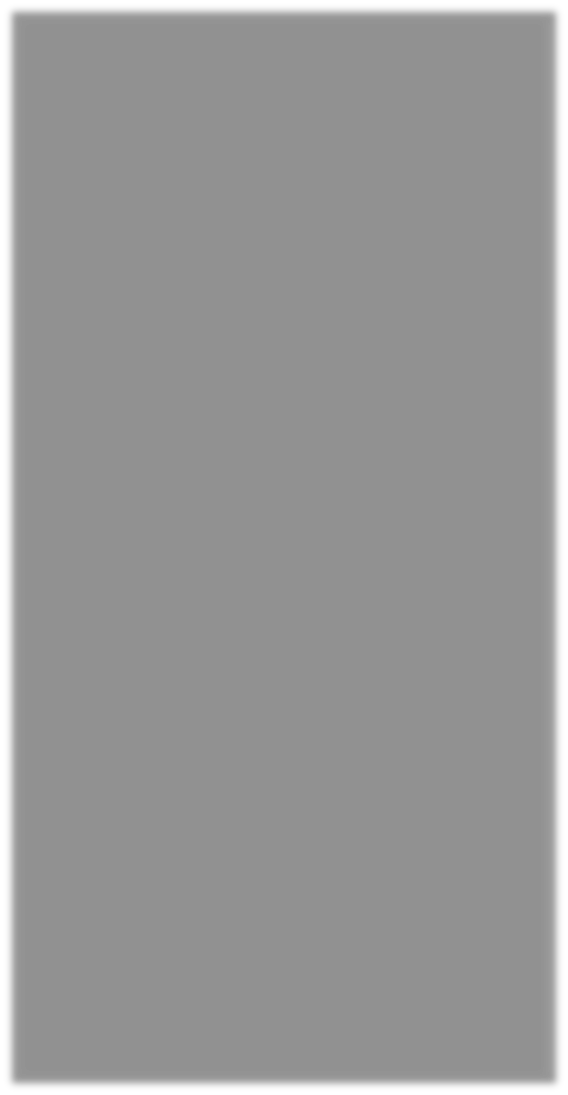
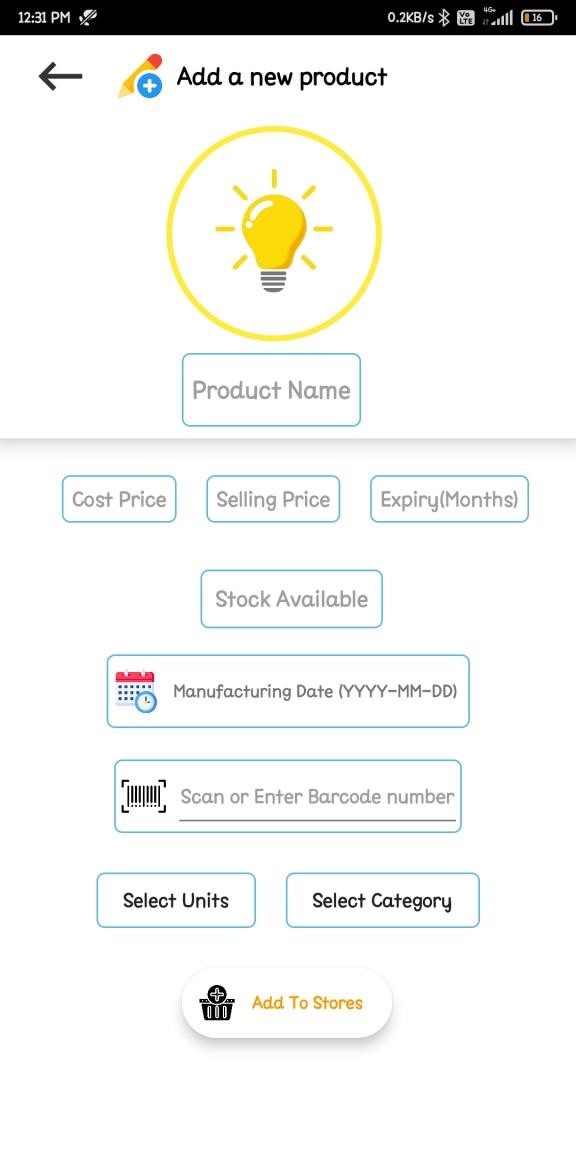
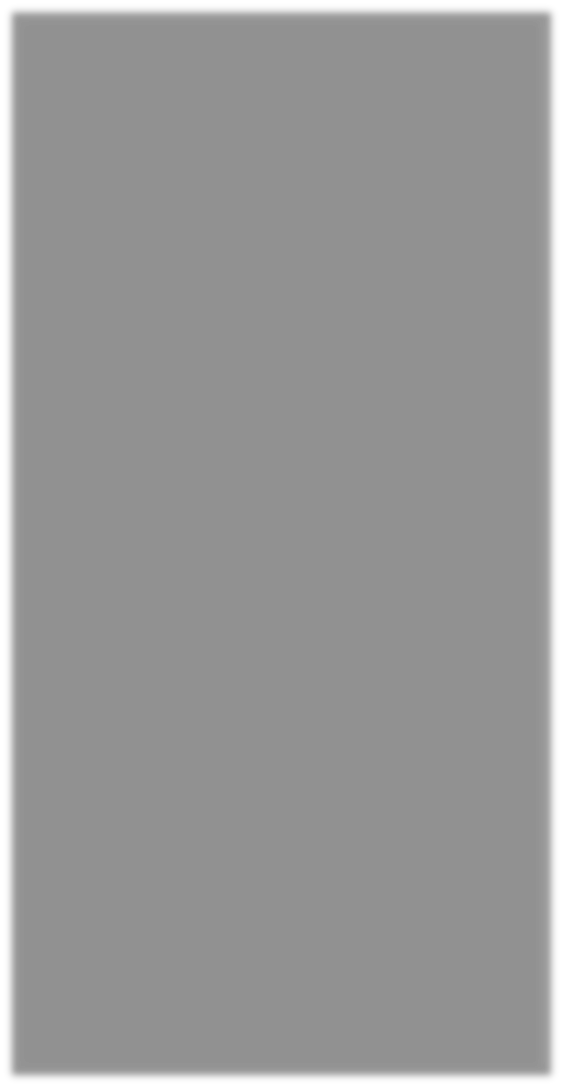
**Fig 7.1 Login Page**



**Fig 7.2 List of Goods**



**Fig 7.3 Billing Page**



### Fig 7.4 Adding a new Product Fig 7.5 Product added

**CHAPTER - 8**

## 8.FUTURE ENHANCEMENTS

In the future we wish to mainly focus on improving and withstanding the efficiency of the application. We also wish to develop an interface for the statistics that occur between the wholesale buyers and whole sale sellers. We look forward to enhance the categorization in the application for the users.

We wish to provide more details about the products registered in the application and also provide more insights about the demand of a product in a store.

We also focus to provide further information about each category and the products lying under a category such as product/s that is in demand in a category etc.

**CHAPTER-9**

## 9.CONCLUSION

With the growing demand and supply chain, this application is an effort to manage the supply chain market and also prevent the loss of data.

This also makes the management easy, simple and can be operated from any part of the world any time.

In conclusion, Store Management App has to do with making appropriate effort to stop the rising problem to all manual supermarket operation in order to enhance the operation of such supermarket. This application can be used to aid all supermarkets that are still operating manually have been successfully developed. The software can be implementing in all types of supermarkets.

**CHAPTER-10**

## 10.REFERENCES

* Astuti, Y., & Nugroho, A. (2014). Sistem inventarisasi aset tetap (Studi kasus SDN Sidomukti, Ambal, Kebumen). Jurnal DASI, 15(1), 68.
* Athoillah, M., & Irawan, M. (2013). Perancangan sistem informasi mobile berbasis Android untuk kontrol persediaan barang di gudang. Surabaya Jurnal Sains dan Seni Pomits, 1(1), 1-6.
* DetikInet. (2013). Survei: Pengguna Android lebih puas ketimbang iPhone. Retrieved November 20th, 2015 from https://inet.detik.com/consumer/d2168531/ survei-pengguna-Android-lebih-puas-ketimbangiphone.
* Felker, D. (2011). Android tablet application development for dummies. Hoboken, NJ: John Wiley & Sons.
* Hastanti, R. P., Eka, B., Indah, P., & Wardati, U. (2015). Sistem penjualan berbasis web (e-commerce) pada Tata Distro Kabupaten Pacitan. Bianglala Informatika, 3(2), 1-9.
* Pressman, R. S. (2010). Software engineering: A practitioner’s approach. New York: McGraw-Hill Education.
* Raf, M. (2012). Pengaruh faktor-faktor memotivasi konsumen berbelanja terhadap keputusan konsumen berbelanja di pasar modern Kota Jambi. Digest Marketing, 1(1), 63-75.
* Rubianto, I. M., Nurwandi, L., & Gunadhi, E. (2012). Pemodelan sistem informasi perpustakaan mengunakan metode pengembangan traditional waterfall (Studi kasus di SMAN 8 Garut). Jurnal Algoritma, 9(35), 1-6.
* Sukamto, R. A., & Shalahuddin, M. (2014). Rekayasa perangkat lunak terstruktur dan berorientasi objek. Bandung: Informatika.