

1. STUDY ABOUT ANDROID

1.1 Introduction

Android is a Linux-based mobile phone operating system developed by Google. Android is unique because Google is actively developing the platform but giving it away for free to hardware manufacturers and phone carriers who want to use Android on their devices.

Google's Android operating system is an open-source platform that's currently available on a wide variety of smartphones. Android has its advantages -- it's highly customizable, for one -- but it's also somewhat geeky software that can seem intimidating to smartphone newbies.

Android is available on a variety of handsets, including Google's Nexus One (which is manufactured by HTC) and Verizon's Motorola Droid. The open nature of the Android platform allows handset manufacturers to customize the software for use on their handsets. As a result, the Android software can look and feel very different on different handsets.

1.2 Customizable Interface

All Android smartphones are touch-screen devices; some -- but not all -- have hardware keyboards, too. All come with a desktop that is made up of a certain number of screens (some Android phones have 3, others have 5, while still others have 7) that you can customize to your liking. You can populate screens with shortcuts to apps or widgets that display news headlines, search boxes, or more. The customization is certainly a bonus; no other smartphone platform offers as much flexibility in setting up your desktop screens to your liking.

In addition to using shortcuts on your various screens for accessing apps and files, Android also offers a comprehensive menu. You access the menu in different ways on different phones, but none of them make it difficult to find. From the menu, you can click on the small but neatly organized icons to access apps and features like the Android Market.

The Android interface will vary slightly from phone to phone, but, in general, the software itself has become more polished looking over time. The first version, which I reviewed on the T-Mobile G1 more than a year ago, was somewhat rough around the edges, appearance wise. The latest version, 2.1, which I tested on the new Nexus One, is far sleeker looking.

But even in its latest version, the Android interface lacks some of the polish and pizzazz found in two of its key rivals: Apple's iPhone OS and Palm's webOS. Both of these platforms

look more elegant than Android. The iPhone OS, in particular, is a bit more intuitive to use; getting comfortable with Android can take more time and practice.

1.3 Beyond the Phone

A modified version of Android is used in the Google TV, the Barnes & Noble Nook eReader, the Samsung Galaxy Tab, and countless other devices. Parrot makes both a digital photo frame and a car stereo system powered by modified versions of Android.

1.3.1 Google Android

Android's open nature means that almost anyone can create an application to run on it. And you will find a growing selection of titles available in the Android Market, the platform's answer to Apple's App Store. Android supports multi-tasking, too, so you can run multiple apps at once. This means you can open a Web page, for example, and as it loads, check for incoming e-mail. It's handy.

Android also has the benefit of being closely tied to Google; the company offers lots of excellent mobile apps. Some, like Google Maps, are available on different mobile platforms, but others, like the excellent Google Maps Navigation (beta), are only available on Android phones.

1.3.2 Open Handset Alliance

Google formed a group of hardware, software, and telecommunication companies called the Open Handset Alliance with the goal of contributing to Android development. Most members also have the goal of making money from Android, either by selling phones, phone service, or mobile applications

1.3.3 Android Market

Anyone can download the SDK (software development kit) and write applications for Android phones. Google doesn't screen developers or applications.

These apps can be downloaded from the Android Market. If the app costs money, you pay for it using Google Checkout. T-Mobile also has an agreement to allow their phone customers to purchase some apps and have the fee added to their monthly phone bill.

Some devices do not include support for the Android Market and may use an alternative market.

1.4 Service Providers

The iPhone has been very popular for AT&T, but unless you void your warranty to unlock it, you can *only* use an iPhone with AT&T. Android is an open platform, so many carriers can

potentially offer Android-powered phones. That doesn't mean individual carriers won't lock the specific phone they sell you to their service, but every major carrier in the US offers an Android phone. Android is currently the fastest growing phone platform in the world.

1.4.1 Data Plan

To use Android on a phone, you need a data plan on top of your voice plan. You can't even activate an Android phone without a data plan enabled. Some networks in the US offered tiered data service, and some offer unlimited plans.

1.4.2 Google Services

Because Google developed Android, it comes with a lot of Google services installed right out of the box. Gmail, Google Calendar, Google Maps, and Google Web search are all pre-installed, and Google is also the default Web page for the Web browser. However, because Android can be modified, carriers can choose to change this. Verizon Wireless, for instance, has modified some Android phones to use Bing as the default search engine

.

1.5 Touchscreen

Android supports a touch screen and is difficult to use without one. You can use a trackball for some navigation, but nearly everything is done through touch. Android also supports multi-touch gestures such as pinch-to-zoom in versions 2.1 (Eclair) and above.

1.6 Keyboard

The initial release of Android required a separate keyboard. However, "Cupcake," (Android 1.5) and later editions have all supported an on-screen keyboard. That means you can use models like the Verizon Droid 2 that include a slide-out physical keyboard or the HTC EVO that rely entirely on the touch screen.

1.7 Features of Android

- i. Handset layouts.
- ii. Storage.
- iii. Connectivity.
- iv. Messaging.
- v. Web Browser
- vi. Java Virtual Machine
- vii. Media Support

1.8 Causes for Confusion

But not all applications run on all versions of Android -- and there are plenty of versions of the software out there, which can cause some confusion. The Motorola Droid, for example, was the first Android phone to feature version 2.0 of the OS. At the time of its launch, the Droid was the only phone that could run Google Maps Navigation (beta). Now, the Nexus One features the most recent version of Android (2.1, at the time of this writing), and is the only phone that can run the new Google Earth app for Android. And newer phones don't always run the newest versions of Android; some new handsets end up shipping with older versions.

Adding to the confusion is the fact that the different versions of Android offer different features, and that manufacturers can decide whether or not to enable certain features. For example, multi-touch -- which allows a phone's touch screen to register more than one touch at a time so you can do things like pinch and spread a screen to zoom in and out -- is available on some Android phones but not others.

The Android OS lacks the elegance of its chief rivals, Apple's iPhone OS and Palm's webOS, and the fact that it's available in so many versions can be very confusing. But it has the benefit of being available on a variety of handsets, and offers customization its rivals can't touch. If you're willing to put in the time to learn all about Android and how to use it, you're likely to find that this mobile platform is powerful.

1.9 The Bottom Line

Android is an exciting platform for consumers and developers. It is the philosophical opposite of the iPhone in many ways. Where the iPhone tries to create the best user experience by restricting hardware and software standards, Android tries to insure it by opening up as much of the operating system as possible.

This is both good and bad. Fragmented versions of Android may provide a unique user experience, but they also mean fewer users per variation. That means it's harder to support for app developers, accessory makers, and technology writers. Because each Android upgrade must be modified for the specific hardware and user interface upgrades of each device, that also means it takes longer for modified Android phones to receive updates.

Fragmentation issues aside, Android is a robust platform that boasts some of the fastest and most amazing phones and tablets on the market.

2. ECLIPSE

2.1 Introduction

The software chosen for the development of the project 'Biz Retail' is Eclipse. The ADT (Android Developer Tools) Plugin can be used along with Eclipse to provide an efficient development environment.

2.2 About Eclipse

Eclipse is a Java-based open source platform that allows a software developer to create a customized development environment (IDE) from plug-in components built by Eclipse members. Eclipse got its start in 2001 when IBM donated three million lines of code from its Java tools. The original goal of Eclipse was to create and foster an open source IDE community that would complement the community that surrounds Apache. Rumour has it that a secondary goal was to "eclipse Microsoft Visual Studio" which is how the platform got its name. The major advantage to an open source development platform is that it allows an IT department to mix and match development tools rather than being committed to a single vendor's suite of development products. Although the Eclipse Platform is written in Java, it supports plug-ins that allow developers to develop and test code written in other languages.

2.3 About Android Developer Tools (ADT)

ADT (Android Developer Tools) is a plugin for Eclipse that provides a suite of tools that are integrated with the Eclipse IDE. It offers you access to many features that help you develop Android applications quickly. ADT provides GUI access to many of the command line SDK tools as well as a UI design tool for rapid prototyping, designing, and building of your application's user interface. Because ADT is a plugin for Eclipse, you get the functionality of a well-established IDE, along with Android-specific features that are bundled with ADT. The following describes important features of Eclipse and ADT:

2.3.1 Integrated Android Project

ADT integrates many development workflow tasks into Eclipse, making it easy for you to rapidly develop and test your Android applications.

2.3.2 SDK Tools integration

Many of the SDK tools are integrated into Eclipse's menus, perspectives, or as a part of background processes ran by ADT.

2.3.3 Java and XML editors

The Java programming language editor contains common IDE features such as compile time syntax checking, auto-completion, and integrated documentation for the Android framework APIs. ADT also provides custom XML editors that let you edit Android-specific XML files in a form-based UI. A graphical layout editor lets you design user interfaces with a drag and drop interface.

2.3.4 Graphical Layout Editor

Edit and design your XML layout files with a drag and drop interface. The layout editor renders your interface as well, offering you a preview as you design your layouts. This editor is invoked when you open an XML file with a view declared (usually declared in res/layout.

2.3.5 XML Resources Editor

Edit XML resources with a simple graphical interface. This editor is invoked when you open an XML file.

2.3.6 Graphical Layout Editor

ADT provides many features to allow you to design and build your application's user interface. Many of these features are in the graphical layout editor, which you can access by opening one of your application's XML layout files in Eclipse.

2.3.7 Canvas

In the middle of the editor is the canvas. It provides the rendered view of your layout and supports dragging and dropping of UI widgets directly from the palette. You can select the platform version used to render the items in the canvas. Each platform version has its own look and feel, which might be the similar to or radically different from another platform version. The canvas also provides context-sensitive actions in the layout actions bar, such as adjusting layout margins and orientation. The layout actions bar displays available actions depending on the selected UI element in the canvas.

2.3.8 Palette

On the left side of the editor is the palette. It provides a set of widgets that you can drag onto the canvas. The palette shows rendered previews of the widgets for easy lookup of desired UI widgets.

2.4 Configuration Chooser

At the top of the editor is the configuration chooser. It provides options to change a layout's rendering mode or screen type.

2.4.1 Canvas and Outline view

The canvas is the area where you can drag and drop UI widgets from the palette to design your layout. The canvas offers a rendered preview of your layout depending on factors such as the selected platform version, screen orientation, and currently selected theme that you specify in the configuration chooser. You can also drag and drop items into the outline view, which displays your layout in a hierarchical list. The outline view exposes much of the same functionality as the canvas but offers another method of organization that is beneficial for ordering and quickly selecting items. When you right-click a specific item in the canvas or outline view, you can access a context-sensitive menu that lets you modify the following attributes of the layout or view:

2.4.2 View and Layout properties

When you right-click a view or layout in the canvas or outline view, it brings up a context-sensitive menu that lets you set things such as

- i. ID of the view or layout
- ii. Text of the view
- iii. Layout width
- iv. Layout height
- v. Properties such as alpha or clickable

2.4.3 Animation

If your layout or view is animated, you can preview the animation directly in the canvas. Right click an item in the canvas and select Play Animation. If animation is not associated with item, an option is available in the menu to create one.

2.5 Web Platform

The Eclipse Web Tools Platform (WTP) project is an extension of the Eclipse platform with tools for developing Web and Java EE applications. It includes source and graphical editors for a variety of languages, wizards and built-in applications to simplify development, and tools and APIs to support deploying, running, and testing apps

2.6 Modelling Platform

The Modelling project contains all the official projects of the Eclipse Foundation focusing on model-based development technologies. They are all compatible with the Eclipse Modeling Framework created by IBM. Those projects are separated in several categories: Model Transformation, Model Development Tools, Concrete Syntax Development, Abstract Syntax Development, Technology and Research, and Amalgam.

2.7 Advantages of Eclipse

The biggest advantages in eclipse are as follows

- i. Code Completion
- ii. Refactoring
- iii. Syntax Checking
- iv. Debugging
- v. Visual Editor
- vi. Alignment
- vii. Compilation

2.8 Drawbacks of Eclipse

- i. It is big and quite complex.
- ii. Some things hidden by processing have to be handled manually and some things must be changed from standard sketches
 - a. You have to do the basic imports yourself; now Eclipse makes that dead simple, so that is not a major issue
 - b. You have to remember to replace color type by int, add f at the end of float literals, don't use int() but int(), and so on.

2.9 Justification

Thus Eclipse, which is the most efficient and easy-to-use software has been selected for the development of the project 'Biz Retail'.

3. DATA COLLECTION

Data is collected based on the four different modules such as Purchase, Inventory, Sales, Revenue and Profit.

For the Purchases module, data is collected based on the type of wear such as men, women and kids, subtype of wear such as formals, casuals and ethnic, fabric such as cotton, silk and polyester, price of each type of fabric.

For the Inventory module, the available stock is maintained for each type of wear with their relevant subtype and the fabric.

For the Sales module, data is collected based on the type of wear such as men, women and kids, subtype of wear such as formals, casuals and ethnic, fabric such as cotton, silk and polyester, price of each type of fabric.

For the Revenue and Profit module, revenue is calculated based on the purchase and profit is calculated based on the remaining items in the inventory, sales made by the customer.

3.1 Table Specifications

3.1.1 Purchase Data

Column Name	Data Type	Description
Type	Text	It describes the type of wear such as men, women and kids.
Subtype	Text	It describes the subtype of wear such as formal, casual and ethnic.
Fabric	Text	It describes the type of fabric such as cotton, silk and polyester.
Quantity	Integer	Quantity of the item
Price	Integer	Price of an item
Total Price	Integer	Total price of the required quantity

3.1.2 Inventory Data

3.1.2.1 Type

Column Name	Data Type	Description
Men	Text	It describes the men wear
Women	Text	It describes the women wear
Kids	Text	It describes the kids wear

3.1.2.2 Men Subtype

inventory_men

Column Name	Data Type	Description
Formals	Integer	It describes the available quantity of the men formals
Casuals	Integer	It describes the available quantity of the men casuals
Ethnic	Integer	It describes the available quantity of the men ethnic

3.1.2.3 Women Subtype

inventory_women

Column Name	Data Type	Description
Formals	Integer	It describes the available quantity of the women formals
Casuals	Integer	It describes the available quantity of the women casuals
Ethnic	Integer	It describes the available quantity of the women ethnic

3.1.2.4 Kids Subtype

inventory_kids

Column Name	Data Type	Description
Formals	Integer	It describes the available quantity of the kids formals
Casuals	Integer	It describes the available quantity of the kids casuals

Ethnic	Integer	It describes the available quantity of the kids ethnic
--------	---------	--

3.1.2.5 Men Fabric

men_formals

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

men_caasuals

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

men_ethnic

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

3.1.2.6 Women Fabric

women_formals

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

women_casuals

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

women_ethnic

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

3.1.2.7 Kids Fabric

kids_formals

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

kids_casuals

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton
Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

kids_ethnic

Column Name	Data Type	Description
Cotton	Integer	It describes the available quantity of cotton

Silk	Integer	It describes the available quantity of silk
Polyester	Integer	It describes the available quantity of polyester

3.1.3 Sales Data

sales

Column Name	Data Type	Description
Type	Text	It describes the type of wear such as men, women and kids.
Subtype	Text	It describes the subtype of wear such as formal, casual and ethnic.
Fabric	Text	It describes the type of fabric such as cotton, silk and polyester
Quantity	Integer	Quantity of the item
Price	Integer	Price of an item
Total Price	Integer	Total price of the required quantity

3.1.4 Revenue and Profit Data

revenue

Column Name	Data Type	Description
Revenue	Integer	It describes the total purchase of the purchased items
Sales	Integer	It describes the total sales of the sold items
Tax	Integer	It describes the tax
Profit	Integer	It describes the profit

4. PROJECT DESCRIPTION

4.1 Introduction

The project “Biz Retail” aims to produce a useful retail application which can be used in the textile stores.

4.2 Purpose

The purpose of this project is to outline the Retail Management System, “Biz Retail for Android”. It also outlines the external behaviour of the project including non-functional requirements, design constraints and other miscellaneous specifications.

4.3 Scope

The scope of the project “Biz Retail” is to ensure portability and compatibility. It also allows for maintenance, upgrades and periodic backups by the developed and authorized personal.

4.4 Context

The area of interest of this project will be a small business management. The end user will be the customer who buys the items. The documents or bills produced will be read by the customers, suppliers, auditors as well as tax officers

4.5 Existing System

All retail management systems are required to keep track of the stocks available, ordering and sellind details. Many software and non-software are also available. The major approaches were manual system, spreadsheet and commercial softwares. The manual system will be error prone and inflexible. Spreadsheet may not suit for large applications. Commercial Softwares require expensive licenses.

4.6 Proposed System

The project Biz Retail aims to build a new retail management system that has much the same functionality as the existing commercial softwares but with open source license.

4.7 Constraints

The project Biz Retail has the constraint of working on android which includes limited power and resources.

4.8 System Requirements

4.8.1 Hardware Requirements

Screen 7 inch touch screen

Memory 256 MB RAM or more

Processor 256 MHz or more

Disk Space 10 MB or more

4.8.2 Software Requirements

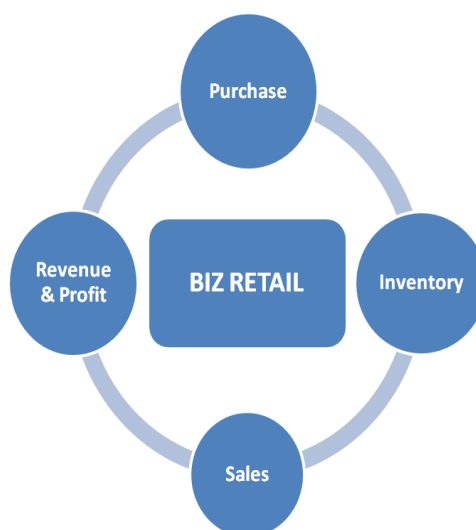
Android OS 2.1 and up

4.9 Module Description

4.9.1 Block Diagram

The project “Biz Retail” is used to keep track of purchase from suppliers, the stock available, sales and revenue. It consists of four modules namely

- a. Purchase
- b. Inventory
- c. Sales
- d. Revenue & Profit



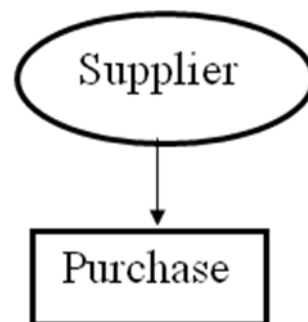
The Biz Retail is a project designed for managing, ordering and selling items. The supplier has the sole responsibility to supply the necessary items as requested by the purchase module. The purchase module checks the availability of items in the inventory. If there are available items in the inventory then the items will be sold.

If the items in the inventory is less or there are no items in the inventory then the purchase module makes purchase from the supplier. The supplier receives orders from the Purchase module. Based on the order received, the supplier will supply the items.

The supplier then makes payment. The payment may be settled either by ready cash or check or credit with tax. The items are sold directly to the customer. The customer buys the items and receives bill for the purchased items. The amount is received through any mode of payment with relevant taxes.

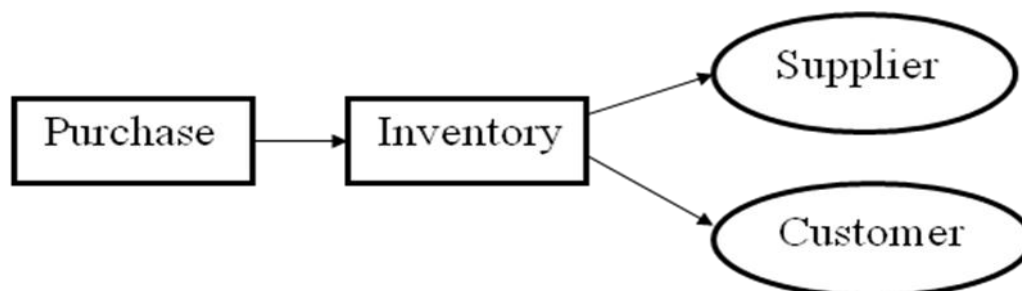
4.9.2 Purchase

The Purchase module is used to purchase items from suppliers.



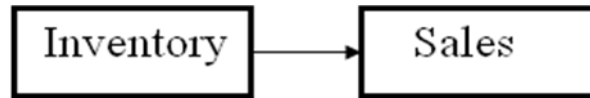
4.9.3 Inventory

If the inventory is empty or low on stock, the Purchase module will be used to purchase items from the suppliers. Only if an item is in the Inventory, it can be put up for sale. The size of the Inventory will be increased if items are purchased from suppliers. The size of the Inventory will be decreased if customers purchase items.



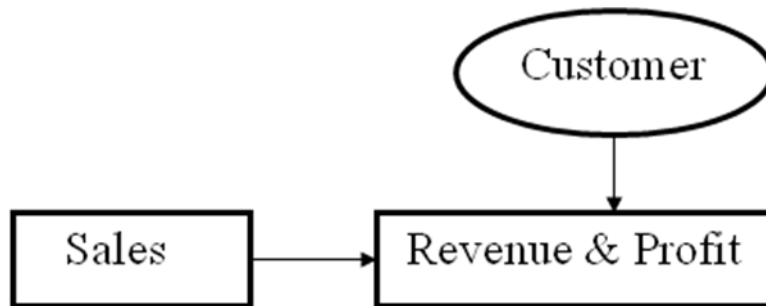
4.9.4 Sales

When a customer purchases an item from the Inventory, the sales module will be updated and the Inventory module will be decremented. The Inventory is also checked from time to time for availability of items. When an item is sold, the Revenue and Profit module will be updated.



4.9.5 Revenue and Profit

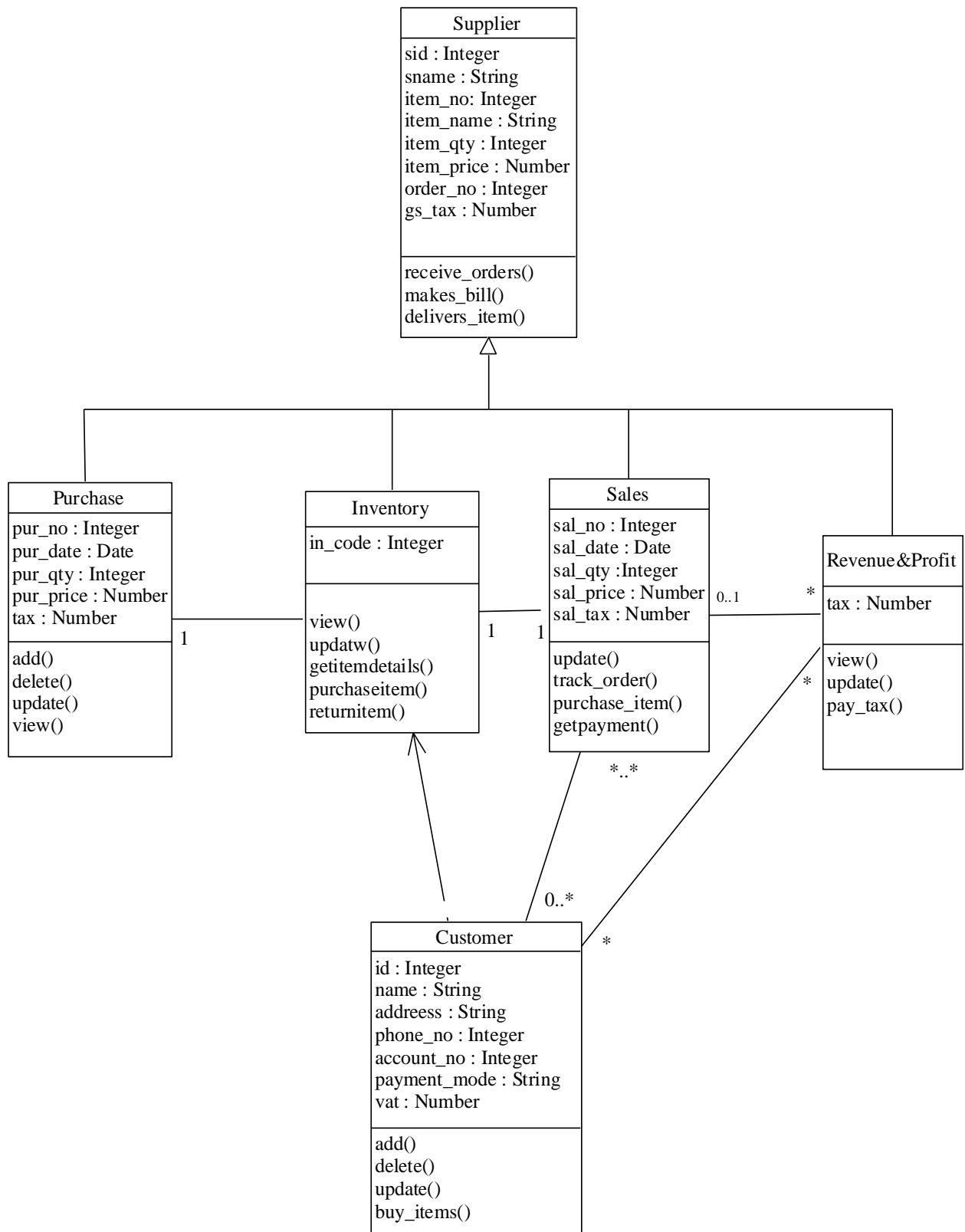
The Revenue and Profit module is used to keep track of the revenue and profit made by the retail store. When an item is purchased by a customer, the revenue and profit module will be updated.



4.10 Module Design

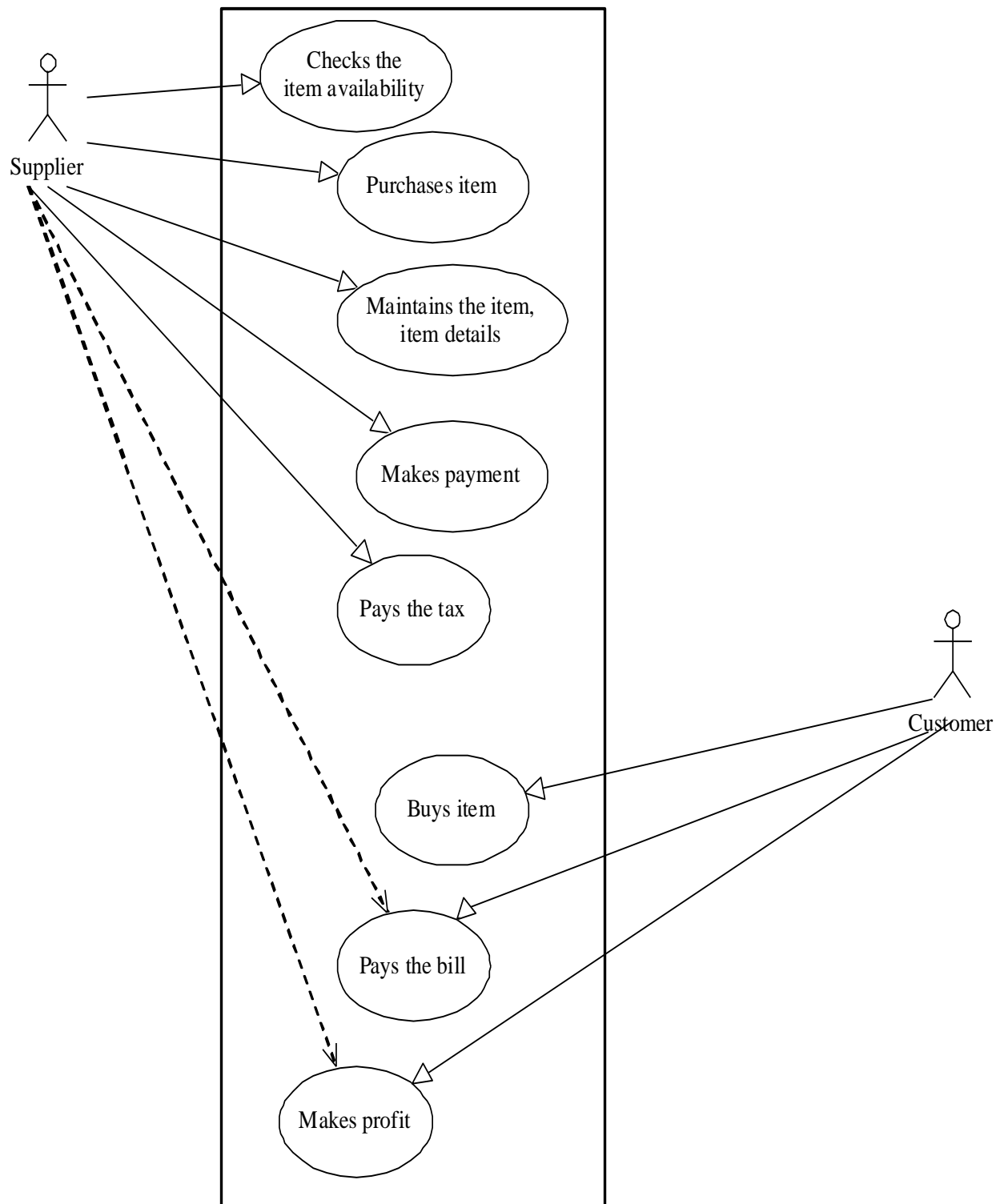
4.10.1 Class Diagram

Class diagram depicts the relationship between different modules within the Biz Retail, as well as the relationship with the supplier and the customer or other modules. The functionality of each module can be seen by the associations. It is evident that the supplier plays a central role in the overall diagram as all modules are inherited from this and most of the functionality passes through it.



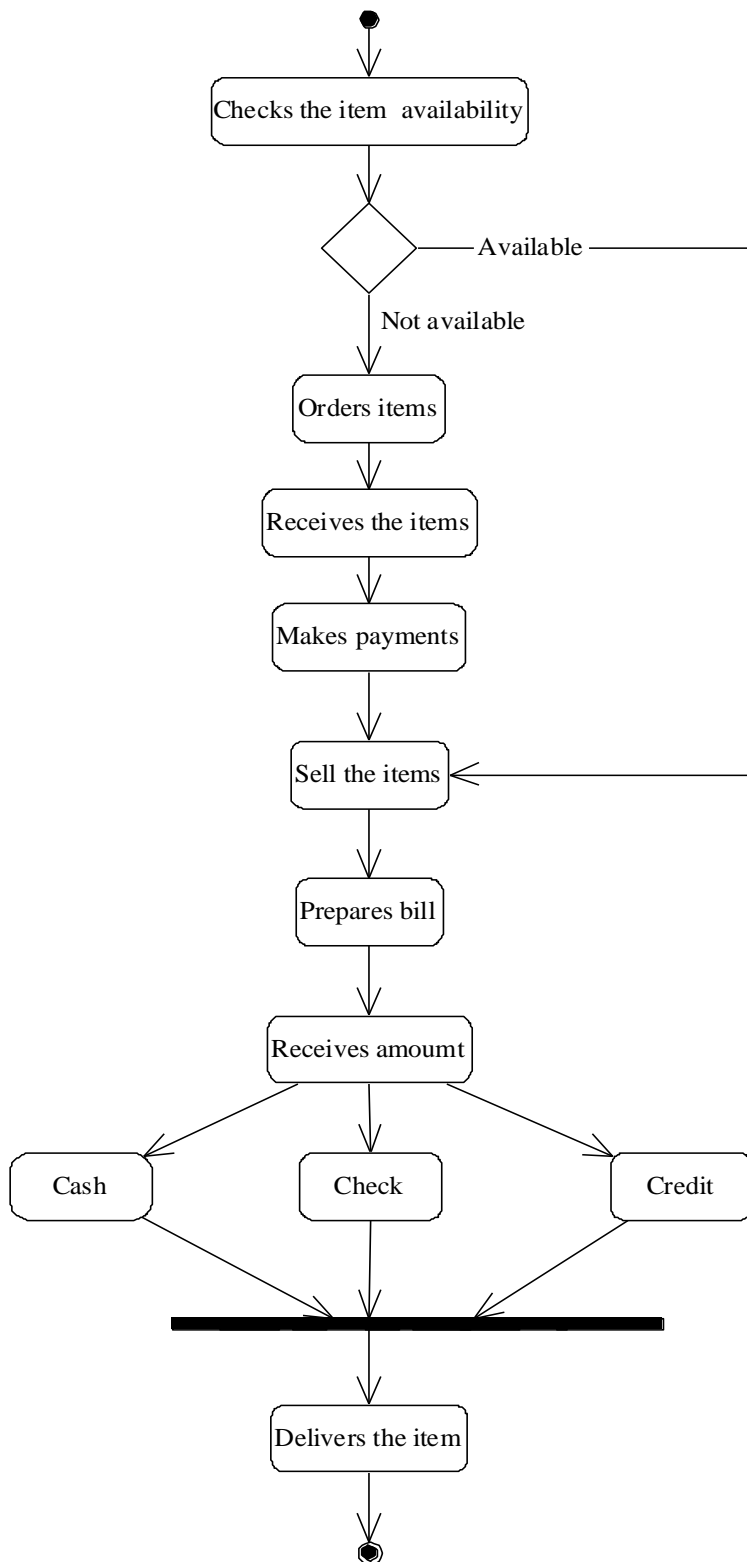
4.10.2 Use Case Diagram

Use case diagram provides the names of use case, roles of actors and their relationship.



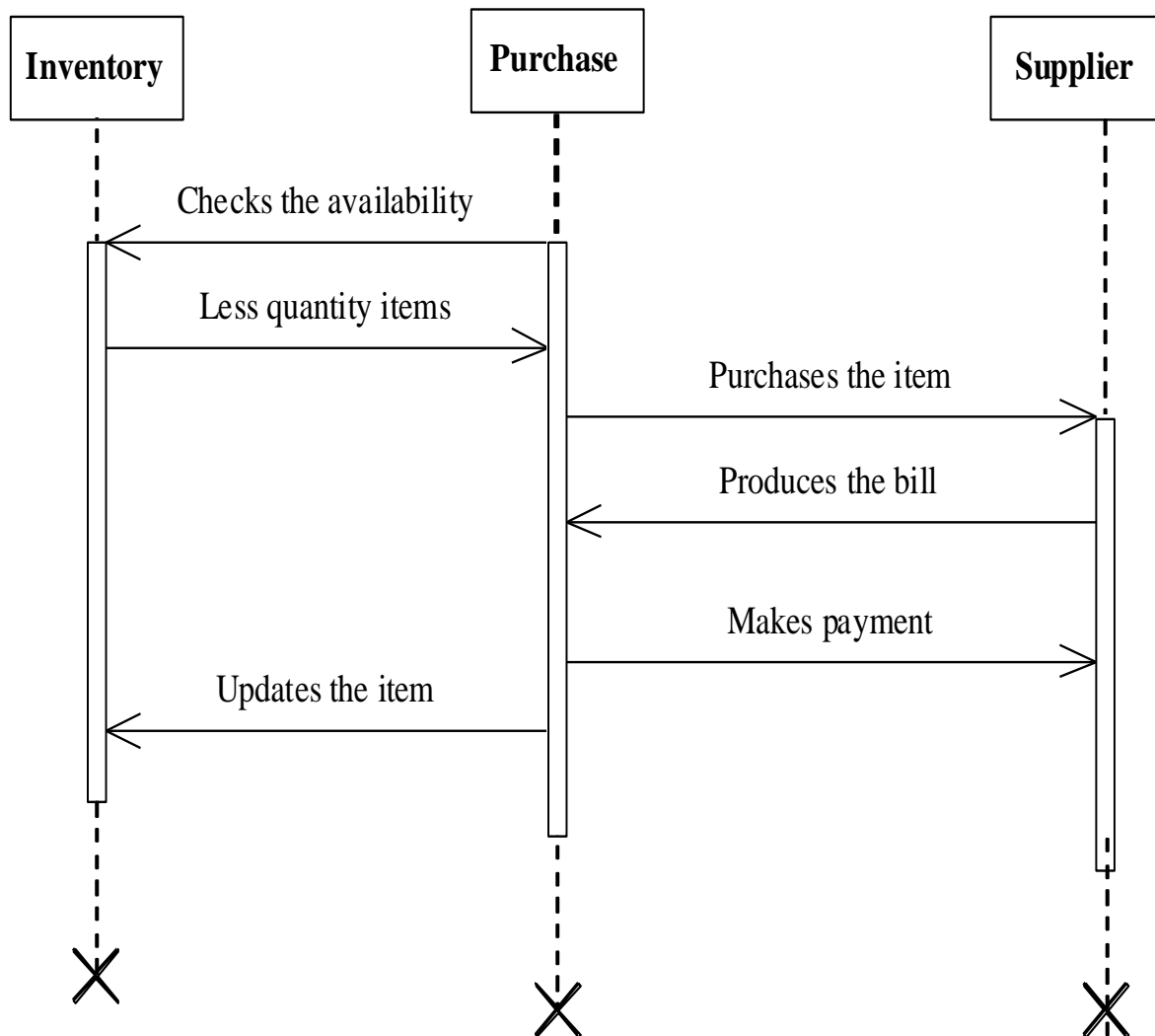
4.10.3 Activity Diagram

Activity diagram shows the sequential and parallel activates in a process, workflows and data flows. In this diagram, activity starts with the checking of available items, proceeds with their corresponding workflow, dataflow processes and stops with the item delivery



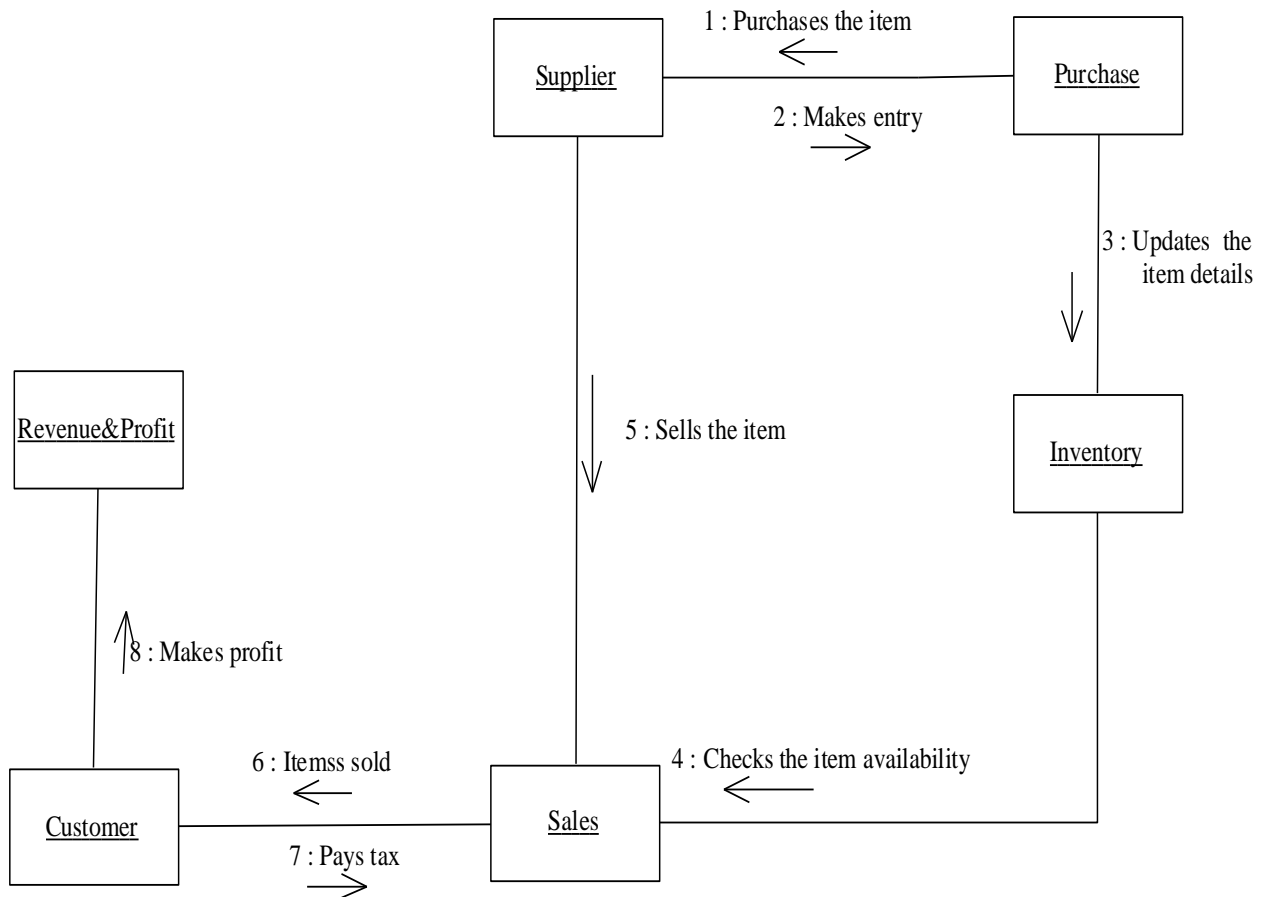
4.10.4 Sequence Diagram

A sequence diagram illustrates a kind of format in which each class or object interacts via message either synchronously or asynchronously. It is generalized between two or more specialized classes.



4.10.5 Collaboration Diagram

Collaboration Diagram illustrates that the object interaction may be either in a graph or network format. In collaboration diagram, the object can be placed anywhere on the diagram.

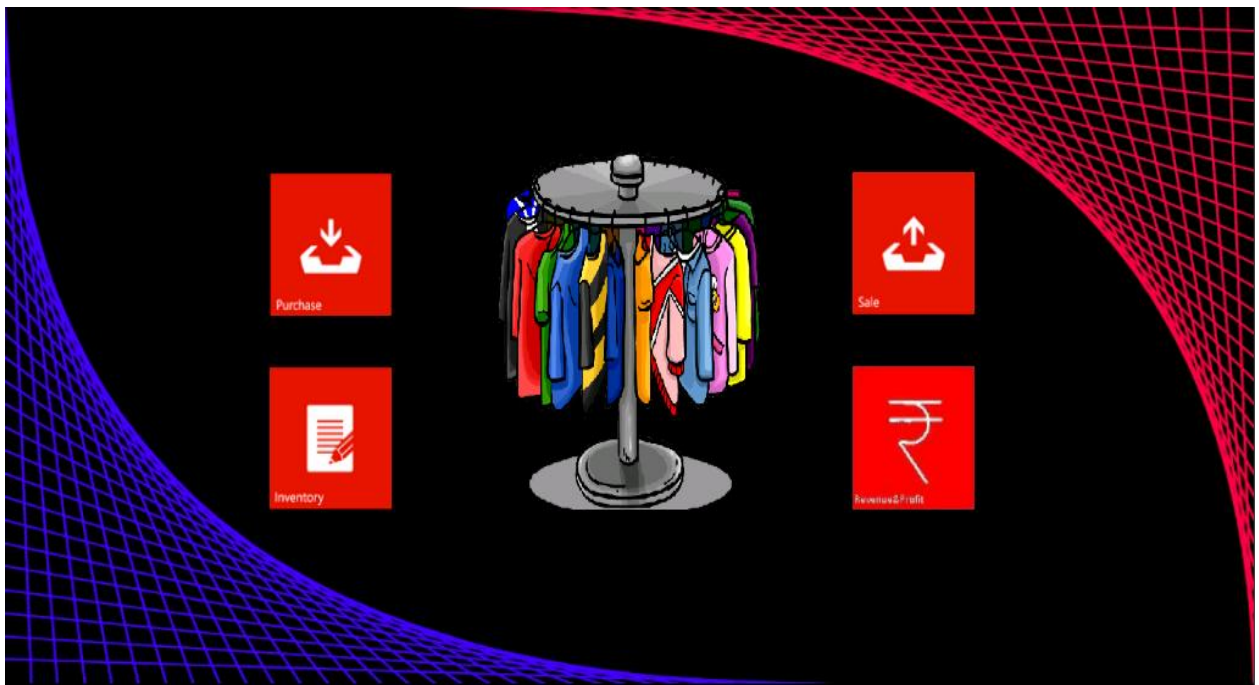


5. RESULTS AND DISCUSSIONS

5.1 Biz Retail



5.2 Main Menu



5.3 Purchase Menu

PURCHASE

Type

Sub-type

Fabric

Quantity

Price

Total Price

☐ Men

☒ Women

☐ Kids

☒ Casuals

☐ Formals

☐ Ethnic

☐ Cotton

☒ Silk

☐ Polyester

2

300

600

id=9,type=Women,subtype=Casuals,fabric=Silk,quantity=2,price=300,totalprice=600

Calculate

Save

View

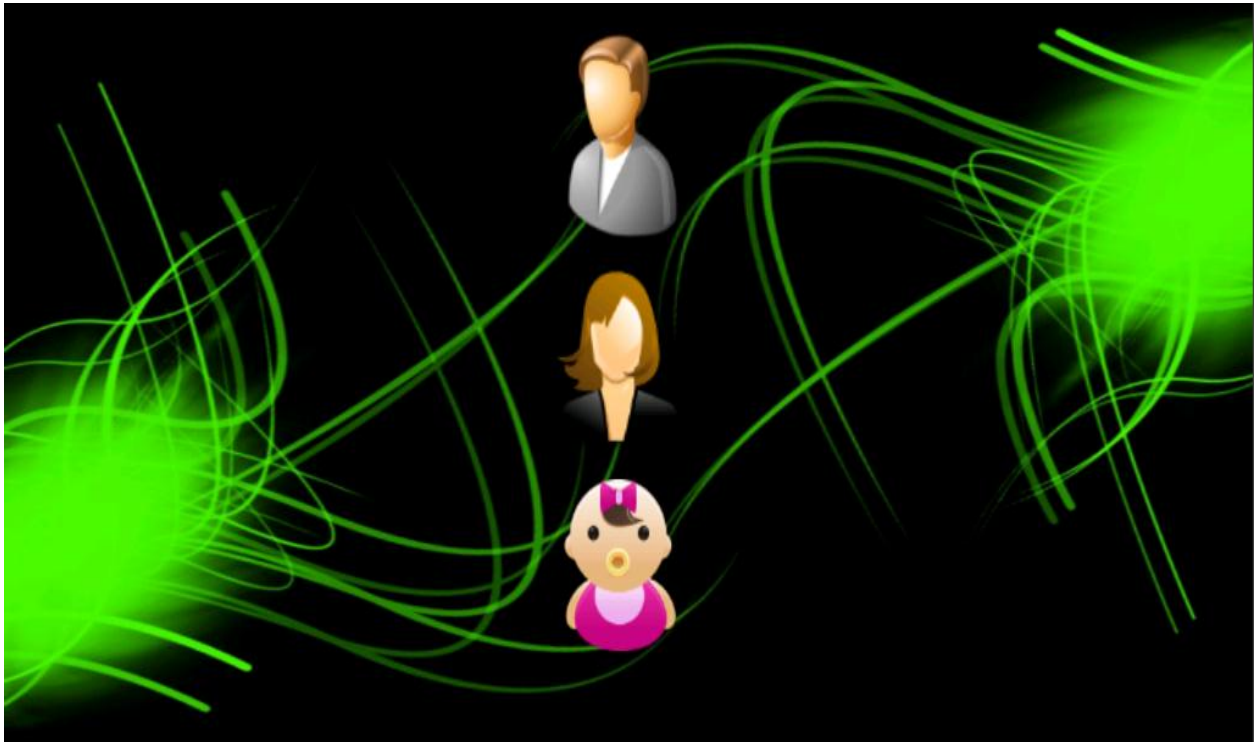
View All Records

5.3.1 All Purchase Records

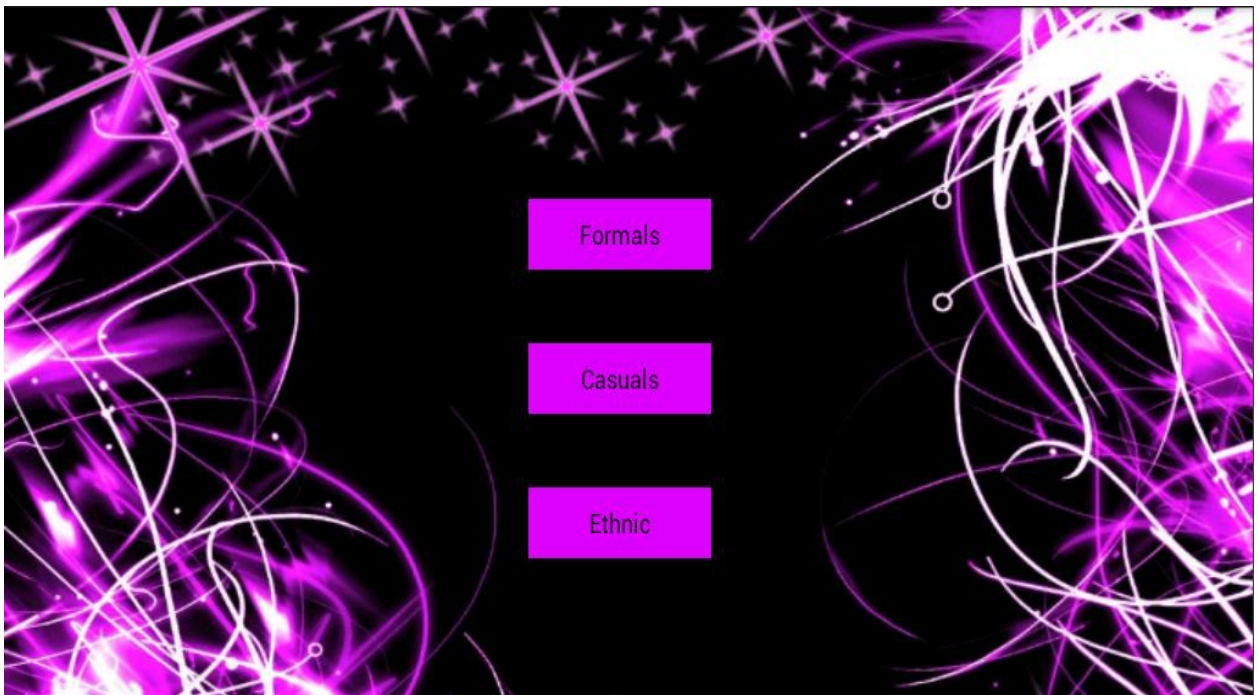
1	Men	Formals	Cotton	1	100	100
2	Men	Formals	Cotton	1	100	100
3	Men	Casuals	Cotton	1	100	100
4	Women	Casuals	Silk	2	300	600
5	Kids	Ethnic	Polyester	1	200	200
6	Kids	Ethnic	Silk	1	300	300
7	Men	Casuals	Silk	2	300	600
8	Women	Casuals	Silk	4	300	1200
9	Women	Casuals	Silk	2	300	600

5.4 Inventory Menu

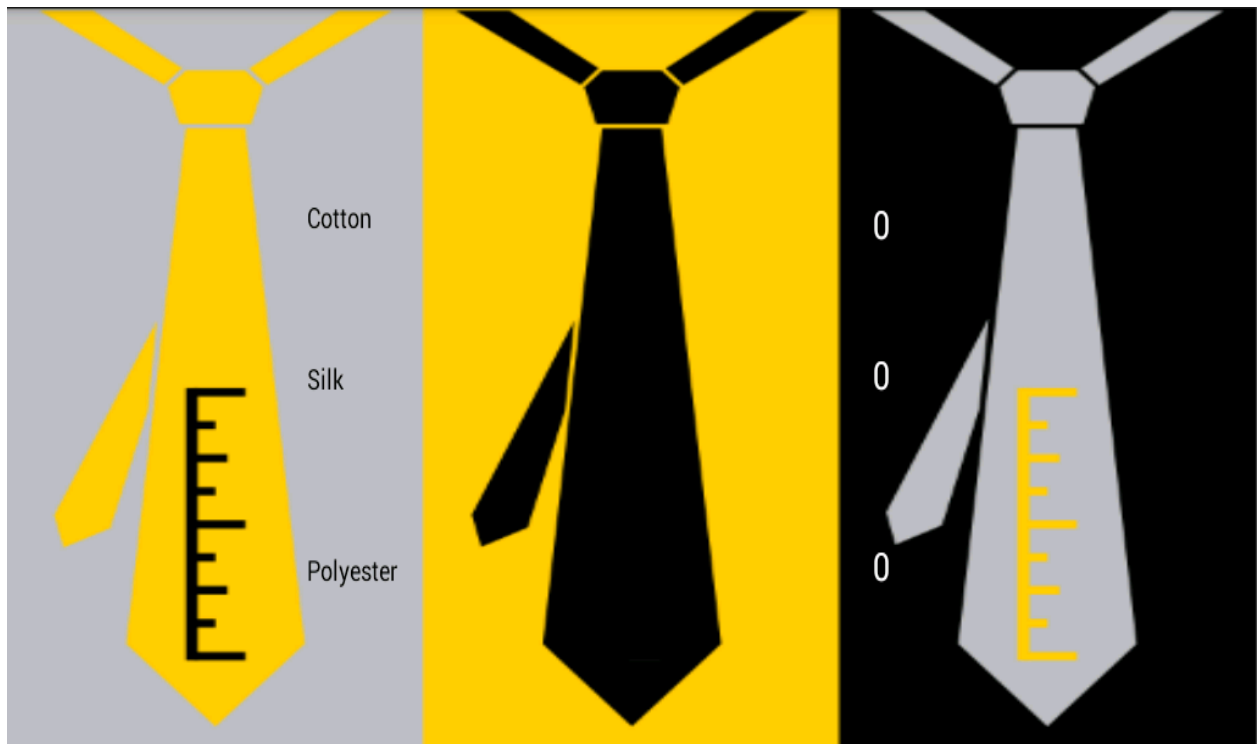
5.4.1 Type Menu



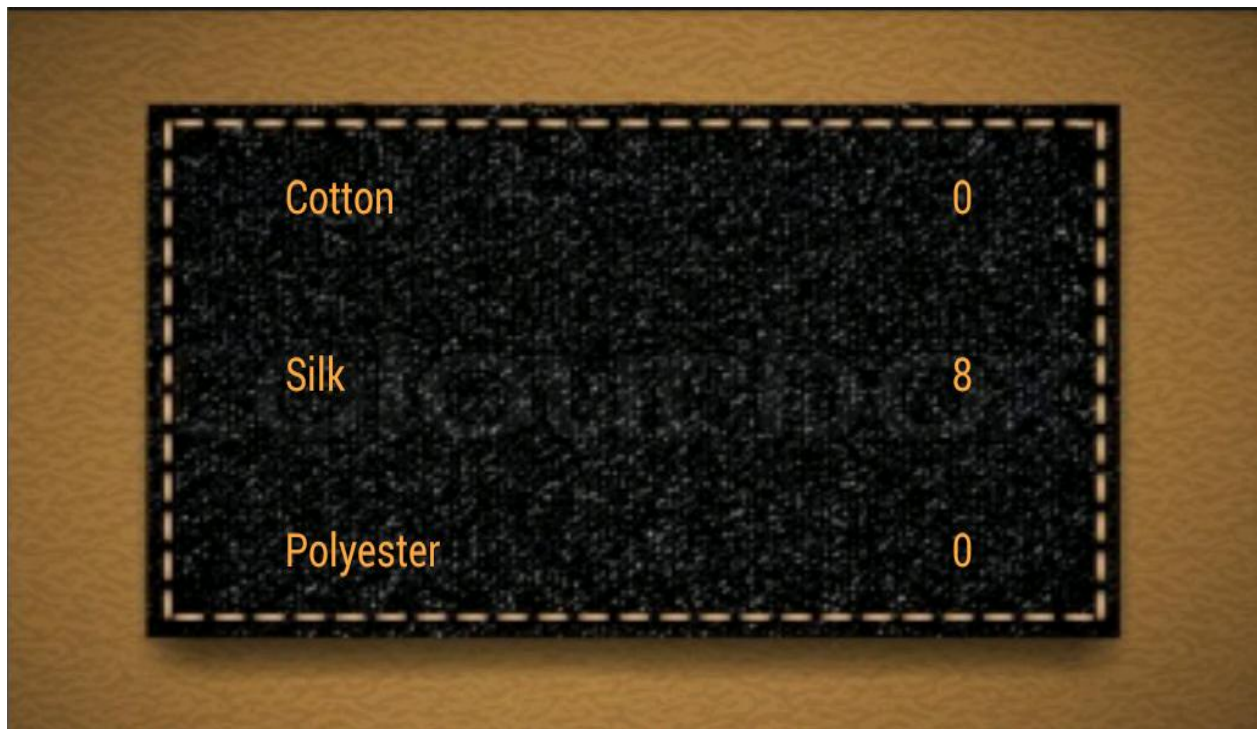
5.4.2 Women Subtype Menu



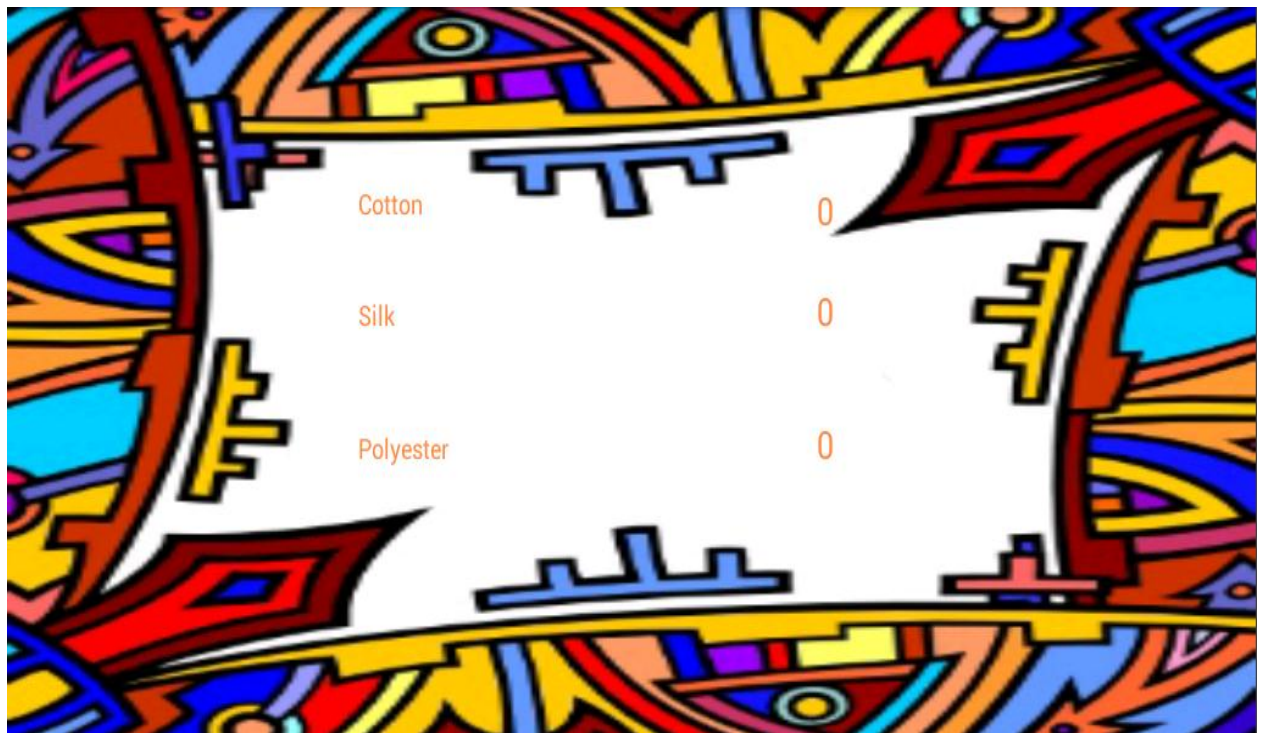
5.4.2.1 Women Formals



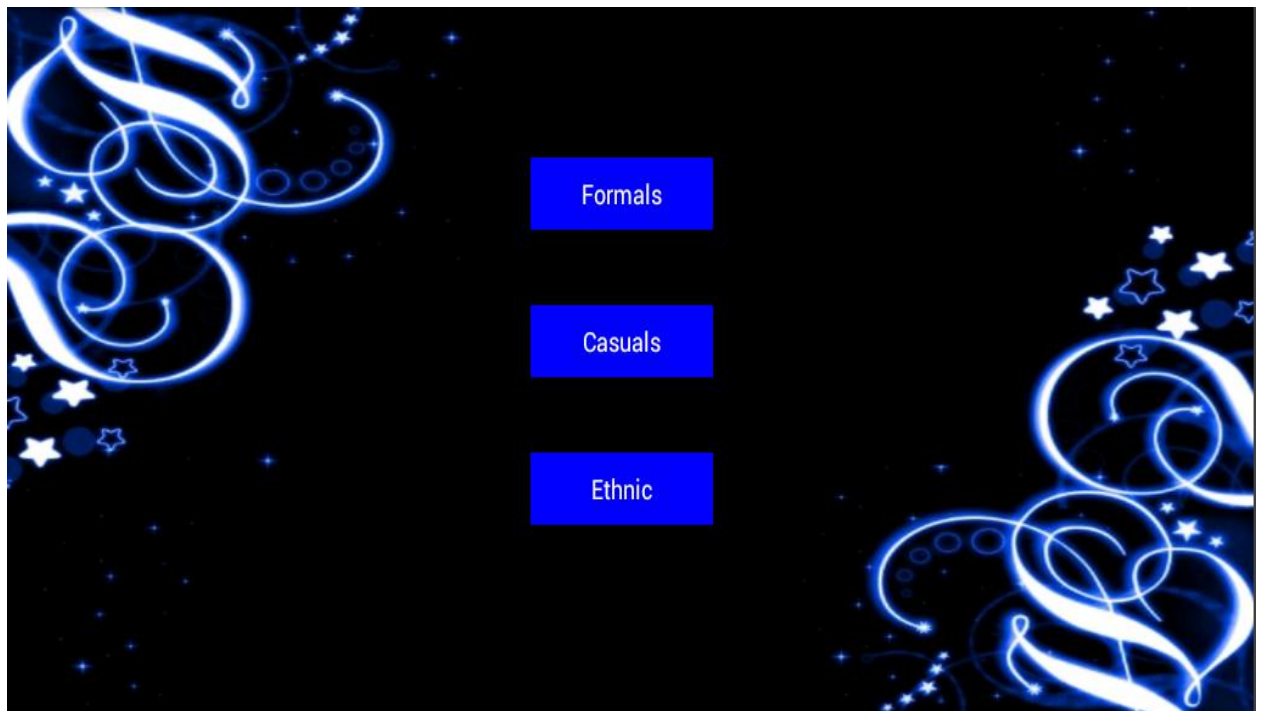
5.4.2.2 Women Casuals



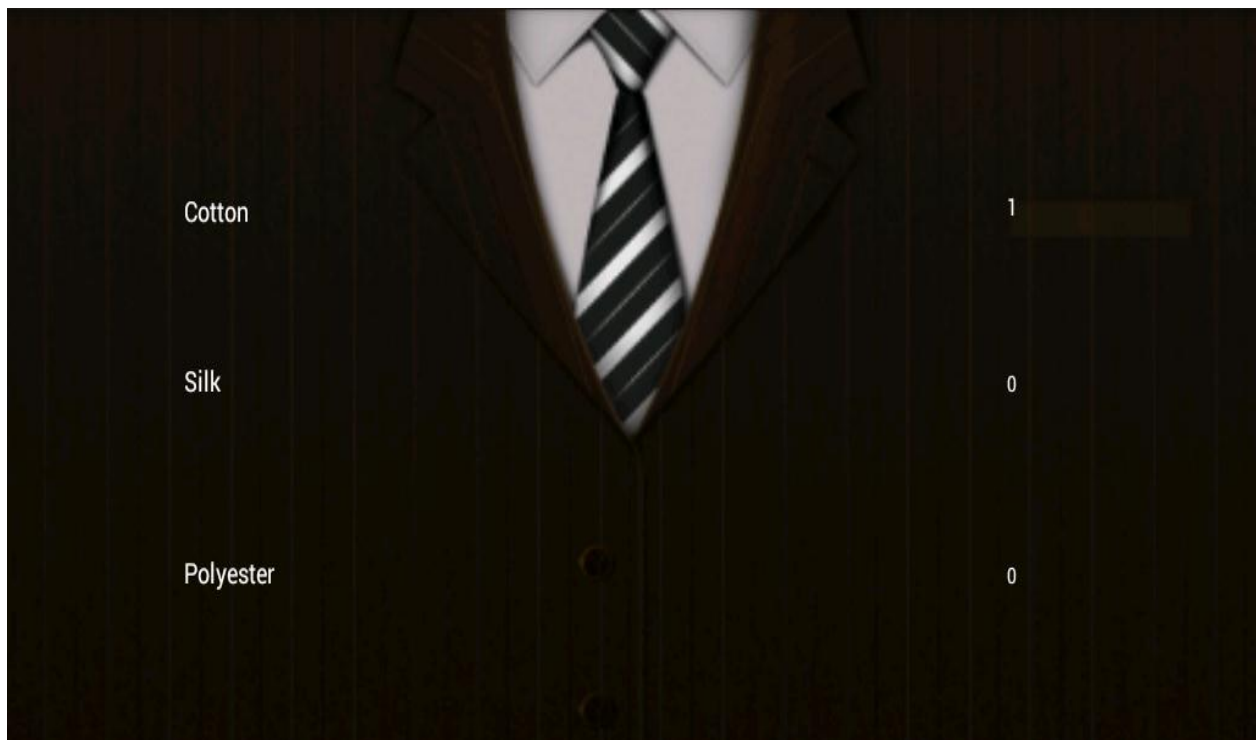
5.4.2.3 Women Ethnic



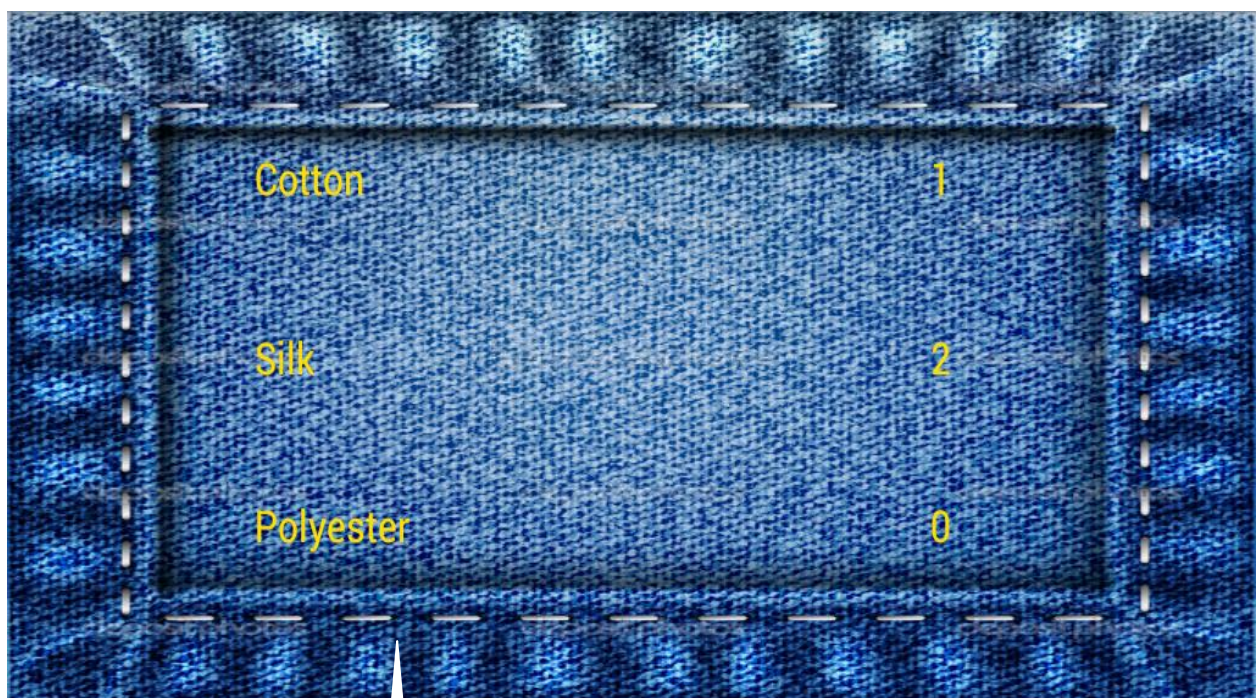
5.4.3 Men Subtype Menu



5.4.3.1 Men Formals



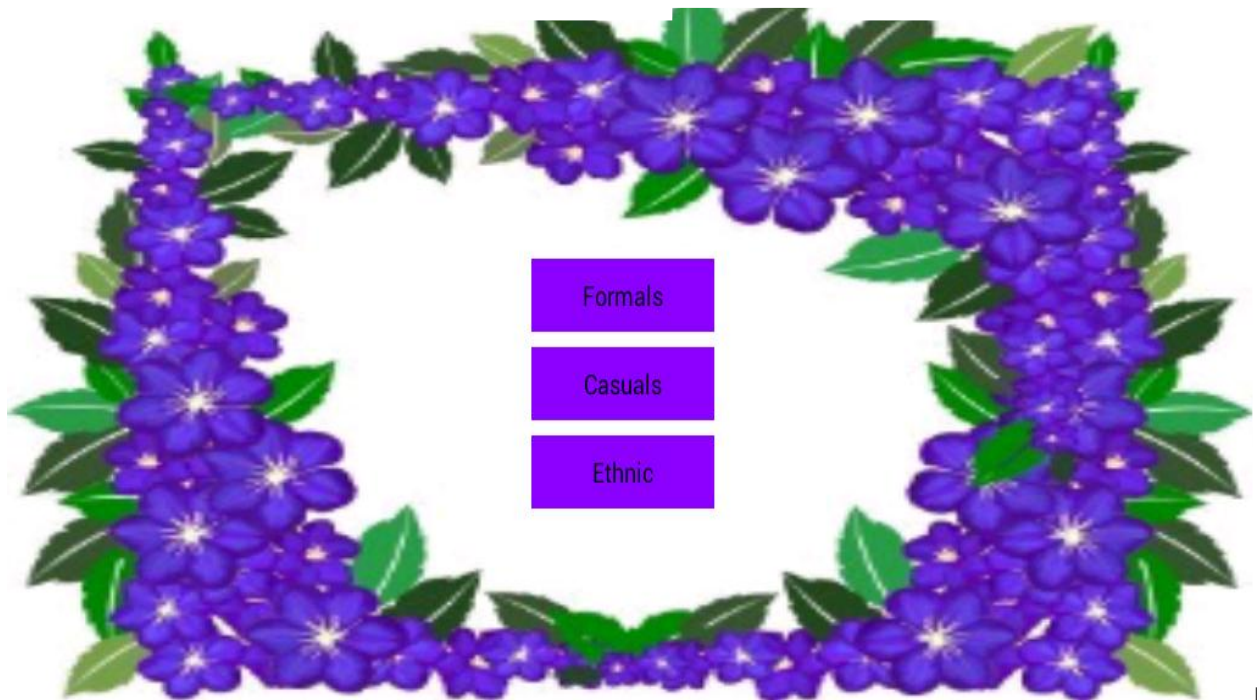
5.4.3.2 Men Casuals



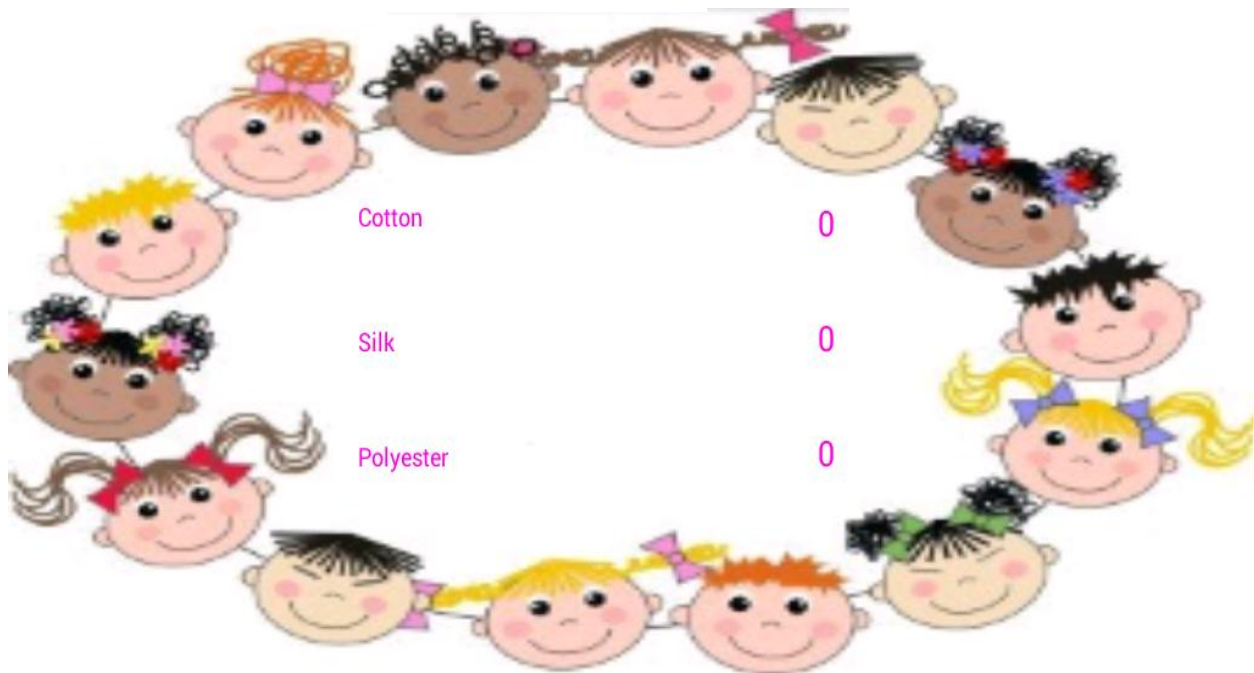
5.4.3.3 Men Ethnic



5.4.4 Kids Subtype Menu



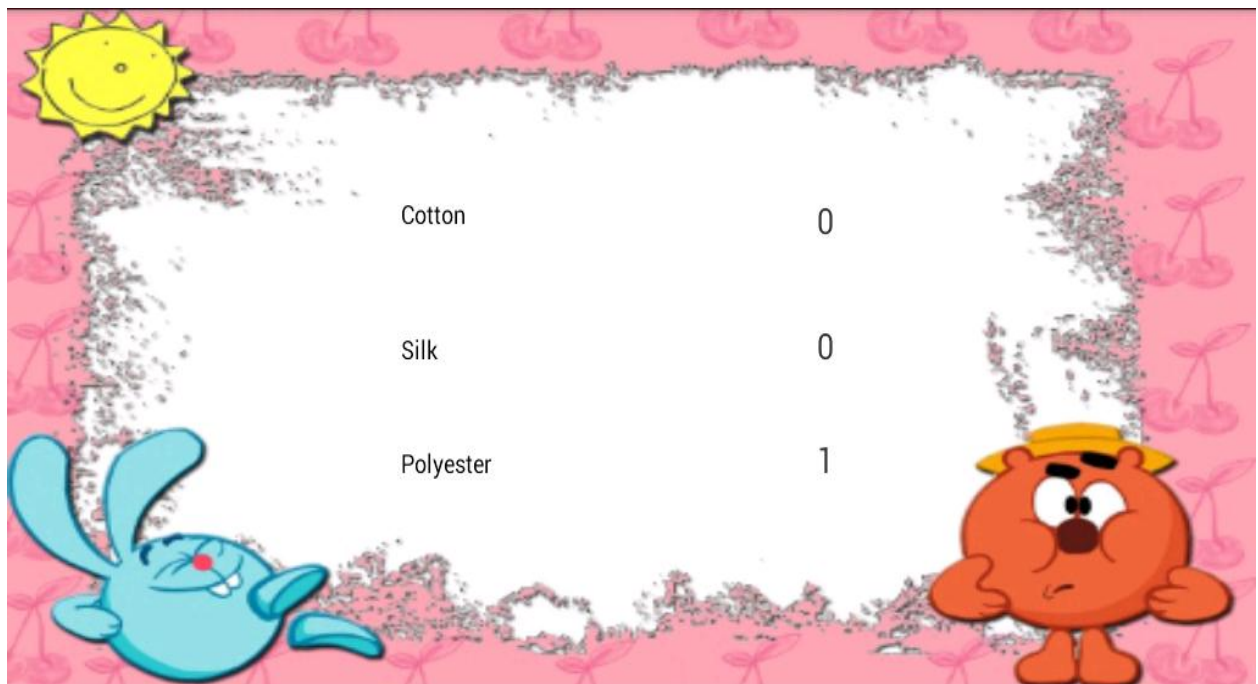
5.4.4.1 Kids Formals



5.4.4.2 Kids Casuals



5.4.4.3 Kids Ethnic



Cotton 0

Silk 0

Polyester 1

5.5 Sales Menu



SALE

Type ☐ Men ☒ Women ☐ Kids

Sub-type ☒ Casuals ☐ Formals ☐ Ethnic

Fabric ☐ Cotton ☒ Silk ☐ Polyester

Quantity

Price

Total Price

id=3,type=Women,subtype=Casuals,fabric=Silk,quantity=1,price=600,totalprice=600

[Calculate](#) [Save](#) [View](#) [View All Records](#)

5.5.1 All Sales Records



1	Men	Formals	Cotton	1	200	200
2	Kids	Ethnic	Silk	1	600	600
3	Women	Casuals	Silk	1	600	600

5.6 Revenue and Profit Menu

Revenue	1400
Tax	140.0
Profit	560.0



APPENDIX

Purchase

```
package com.example.bizretail;

import android.app.Activity;

import android.database.Cursor;

import android.os.Bundle;

import android.view.View;

public class Purchase extends Activity {

    String type,subtype,quant1,price1,total1,fabric;

    int quantity,price,totalprice;

    DBAdapter myDb;

    long newId;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.purchase);

        openDB();

        RadioButton radio0=(RadioButton)findViewById(R.id.radio0);

        radio0.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {

                RadioButton radio1=(RadioButton)findViewById(R.id.radio1);

                RadioButton radio2=(RadioButton)findViewById(R.id.radio2);
```

```

radio1.setChecked(false);

radio2.setChecked(false);

}

});

RadioButton radio1=(RadioButton)findViewById(R.id.radio1);

radio1.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

RadioButton radio0=(RadioButton)findViewById(R.id.radio0);

RadioButton radio2=(RadioButton)findViewById(R.id.radio2);

radio0.setChecked(false);

radio2.setChecked(false);

}

});

RadioButton radioButton1=(RadioButton)findViewById(R.id.radioButton1);

radioButton1.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

EditText pri=(EditText)findViewById(R.id.editText2);

price=100;

fabric="Cotton";

pri.setText(Integer.toString(price));

RadioButton radioButton2=(RadioButton)findViewById(R.id.radioButton2);

RadioButton radioButton3=(RadioButton)findViewById(R.id.radioButton3);

```

```

radioButton2.setChecked(false);

radioButton3.setChecked(false);

}

});

Button button1 = (Button) findViewById(R.id.button1);

button1.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

        EditText quant=(EditText)findViewById(R.id.editText1);

        quant1=quant.getText().toString();

        quantity=Integer.valueOf(quant1);

        EditText pri=(EditText)findViewById(R.id.editText2);

        price1=pri.getText().toString();

        price=Integer.valueOf(price1);

        EditText tot=(EditText)findViewById(R.id.editText3);

        totalprice=quantity*price;

        total1=String.valueOf(totalprice);

        tot.setText(total1);

    }

});

Button button4 = (Button) findViewById(R.id.button4);

button4.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

```

```

setContentView(R.layout.records);

displayRecords(v);

}

});

Button button2 = (Button) findViewById(R.id.button2);

button2.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

        RadioButton radio0=(RadioButton)findViewById(R.id.radio0);

        RadioButton radio1=(RadioButton)findViewById(R.id.radio1);

        RadioButton radioButton1=(RadioButton)findViewById(R.id.radioButton1);

        EditText quant=(EditText)findViewById(R.id.editText1);

        if(radio0.isChecked())

            type="Men";

        if(radio2.isChecked())

            type="Women";

        if(radio1.isChecked())

            type="Kids";

        if(radio3.isChecked())

            subtype="Casuals";

        if(radio4.isChecked())

            subtype="Formals";

        if(radio5.isChecked())

```

```

subtype="Ethnic";

if(radioButton1.isChecked()) {

price=100;

fabric="Cotton";

}

if(radioButton2.isChecked()) {

price=300;

fabric="Silk";

}

if(radioButton3.isChecked()) {

price=200;

fabric="Polyester";

}

quant1=quant.getText().toString();

quantity=Integer.valueOf(quant1);

totalprice=quantity*price;

total1=String.valueOf(totalprice);

myDb.insertRow1(type,subtype,fabric,quantity,price,totalprice);

newId=myDb.insertRow2(type, subtype,fabric, quantity, price, totalprice);

Button button3 = (Button) findViewById(R.id.button3);

button3.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

```

```

Cursor cursor=myDb.getRow2(newId);

displayRecordSet(cursor);

}

});

}

});

}

public void displayRecords(View v) {

Cursor cursor=myDb.getAllRows2();

displayRecordSetAll(cursor);

}

private void displayRecordSet(Cursor cursor) {

String message="";

if(cursor.moveToFirst()) {

do {

int id=cursor.getInt(0);

String type=cursor.getString(1);

String subtype=cursor.getString(2);

String fabric=cursor.getString(3);

int quantity=cursor.getInt(4);

int price=cursor.getInt(5);

int totalprice=cursor.getInt(6);

```

```

message+="id="+id+",type="+type+",subtype="+subtype+",fabric="+fabric
+",quantity="+quantity+",price="+price+",totalprice="+totalprice+"\n";

}while(cursor.moveToNext());

}

cursor.close();

TextView t1=(TextView)findViewById(R.id.textView8);

t1.setText(message);

}

protected void onDestroy() {

super.onDestroy();

closeDB();

}

private void closeDB() {

myDb.close();

}

private void openDB() {

myDb=new DBAdapter(this);

myDb.open();

}

}

```

Inventory

Men Inventory

```

package com.example.bizretail;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

public class Meninv extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.meninv);

        Button button1 = (Button) findViewById(R.id.button1);

        button1.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {

                Intent intent = new Intent(Meninv.this, Menfor.class);

                startActivity(intent);

            }

        });

    }

}

```

Men Formals

```

package com.example.bizretail;

```



```

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

public class Menfor extends Activity {

    DBAdapter myDb;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.menfor);

        openDB();

        TextView formal1=(TextView)findViewById(R.id.textView4);

        String str1=Integer.toString(myDb.sumAll1("Men","Formals","Cotton"));

        formal1.setText(str1);

    }

    protected void onDestroy() {

        super.onDestroy();

        closeDB();

    }

    private void closeDB() {

        myDb.close();

    }

    private void openDB() {

```

```
myDb=new DBAdapter(this);
```

```
myDb.open();
```

```
}
```

```
}
```

Men Casuals

```
package com.example.bizretail;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.TextView;
```

```
public class Mencas extends Activity {
```

```
DBAdapter myDb;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.mencas);
```

```
openDB();
```

```
TextView formal1=(TextView)findViewById(R.id.textView4);
```

```
String str1=Integer.toString(myDb.sumAll1("Men","Casuals","Cotton"));
```

```
formal1.setText(str1);
```

```
}
```

```
protected void onDestroy() {
```

```
super.onDestroy();
```

```

closeDB();

}

private void closeDB() {

myDb.close();

}

private void openDB() {

myDb=new DBAdapter(this);

myDb.open();

}

}

```

Mene Ethnic

```

package com.example.bizretail;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

public class Meneth extends Activity {

DBAdapter myDb;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.meneth);

openDB();

```

```

TextView formal1=(TextView)findViewById(R.id.textView4);

String str1=Integer.toString(myDb.sumAll1("Men","Ethinc","Cotton"));

formal1.setText(str1);

}

protected void onDestroy() {

super.onDestroy();

closeDB();

}

private void closeDB() {

myDb.close();

}

private void openDB() {

myDb=new DBAdapter(this);

myDb.open();

}

}

```

Sales.

```

package com.example.bizretail;

import android.app.Activity;

import android.database.Cursor;

import android.os.Bundle;

import android.view.View;

```

```

public class Sales extends Activity {

String type,subtype,quant1,price1,total1,fabric;

int quantity,price,totalprice,invquant;

DBAdapter myDb;

long newId;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.sales);

openDB();

RadioButton radio0=(RadioButton)findViewById(R.id.radio0);

radio0.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

RadioButton radio1=(RadioButton)findViewById(R.id.radio1);

RadioButton radio2=(RadioButton)findViewById(R.id.radio2);

radio1.setChecked(false);

radio2.setChecked(false);

}

});

RadioButton radio1=(RadioButton)findViewById(R.id.radio1);

radio1.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

```

```

RadioButton radio0=(RadioButton)findViewById(R.id.radio0);

RadioButton radio2=(RadioButton)findViewById(R.id.radio2);

radio0.setChecked(false);

radio2.setChecked(false);

}

});

RadioButton radioButton1=(RadioButton)findViewById(R.id.radioButton1);

radioButton1.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

EditText pri=(EditText)findViewById(R.id.editText2);

price=200;

fabric="Cotton";

pri.setText(Integer.toString(price));

RadioButton radioButton2=(RadioButton)findViewById(R.id.radioButton2);

RadioButton radioButton3=(RadioButton)findViewById(R.id.radioButton3);

radioButton2.setChecked(false);

radioButton3.setChecked(false);

}

});

Button button4 = (Button) findViewById(R.id.button4);

button4.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

```

```

setContentView(R.layout.records);

displayRecords(v);

}

});

Button button1 = (Button) findViewById(R.id.button1);

button1.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

        EditText quant=(EditText)findViewById(R.id.editText1);

        quant1=quant.getText().toString();

        quantity=Integer.valueOf(quant1);

        EditText pri=(EditText)findViewById(R.id.editText2);

        price1=pri.getText().toString();

        price=Integer.valueOf(price1);

        EditText tot=(EditText)findViewById(R.id.editText3);

        totalprice=quantity*price;

        total1=String.valueOf(totalprice);

        tot.setText(total1);

    }

});

Button button2 = (Button) findViewById(R.id.button2);

button2.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

```

```
RadioButton radio0=(RadioButton)findViewById(R.id.radio0);

RadioButton radio1=(RadioButton)findViewById(R.id.radio1);

RadioButton radioButton1=(RadioButton)findViewById(R.id.radioButton1);

EditText quant=(EditText)findViewById(R.id.editText1);

if(radio0.isChecked())

type="Men";

if(radio2.isChecked())

type="Women";

if(radio1.isChecked())

type="Kids";

if(radio3.isChecked())

subtype="Casuals";

if(radio4.isChecked())

subtype="Formals";

if(radio5.isChecked())

subtype="Ethnic";

if(radioButton1.isChecked()) {

price=200;

fabric="Cotton";

}

if(radioButton2.isChecked()) {

price=600;
```



```

fabric="Silk";

}

if(radioButton3.isChecked()) {

price=400;

fabric="Polyester";

}

quant1=quant.getText().toString();

quantity=Integer.valueOf(quant1);

totalprice=quantity*price;

total1=String.valueOf(totalprice);

invquant=myDb.sumAll1(type, subtype,fabric);

TextView errorMsg=(TextView)findViewById(R.id.textView8);

if(quantity>invquant)

errorMsg.setText("Requested stock not available");

else {

errorMsg.setText("");

newId=myDb.insertRow3(type,subtype,fabric,quantity,price,totalprice);

myDb.insertRow1(type,subtype,fabric,-quantity,-(price/2),-(totalprice/2));

Button button3 = (Button) findViewById(R.id.button3);

button3.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

Cursor cursor=myDb.getRow3(newId);

```

```

displayRecordSet(cursor);

}

});

}

}

});

Button button3 = (Button) findViewById(R.id.button3);

button3.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

        TextView textView8=(TextView)findViewById(R.id.textView8);

        textView8.setText("You haven't saved yet");

    }

});

}

public void displayRecords(View v) {

    Cursor cursor=myDb.getAllRows3();

    displayRecordSetAll(cursor);

}

private void displayRecordSet(Cursor cursor) {

    String message="";

    if(cursor.moveToFirst()) {

        do {

```

```

int id=cursor.getInt(0);

String type=cursor.getString(1);

String subtype=cursor.getString(2);

String fabric=cursor.getString(3);

int quantity=cursor.getInt(4);

int price=cursor.getInt(5);

int totalprice=cursor.getInt(6);

message+="id="+id+",type="+type+",subtype="+subtype+",fabric="+fabric
+",quantity="+quantity+",price="+price+",totalprice="+totalprice+"\n";

}while(cursor.moveToNext());

}

cursor.close();

TextView t1=(TextView)findViewById(R.id.textView8);

t1.setText(message);

}

protected void onDestroy() {

super.onDestroy();

closeDB();

}

private void closeDB() {

myDb.close();

}

```

```

private void openDB() {

myDb=new DBAdapter(this);

myDb.open();

}

}

```

Revenue

```

package com.example.bizretail;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.widget.TextView;

public class Revenue extends Activity {

    DBAdapter myDb;

    int proinv,propur;

    double tax;

    int rev;

    double pro;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.revenue);

        openDB();

```

```

TextView revenue=(TextView)findViewById(R.id.textView3);

rev=myDb.sumAllPrice3();

revenue.setText(Integer.toString(rev));

TextView profit=(TextView)findViewById(R.id.textView4);

proinv=myDb.sumAllPrice1();

propur=myDb.sumAllPrice2();

tax=(0.1)*rev;

pro=rev-propur+proinv-tax;

TextView ta=(TextView)findViewById(R.id.textView6);

profit.setText(Double.toString(pro));

ta.setText(Double.toString(tax));

}

```

@Override

```

public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.revenue, menu);

    return true;

}

protected void onDestroy() {

    super.onDestroy();

    closeDB();

}

private void closeDB() {

```

```

myDb.close();

}

private void openDB() {

myDb=new DBAdapter(this);

myDb.open();

}

}

```

RetailDB

```

package com.example.bizretail;

import android.content.ContentValues;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

public class RetailDB {

private static final String TAG = "RetailDB";

public static final String KEY_ROWID = "_id";

public static final int COL_ROWID = 0;

public static final String KEY_TYPE = "type";

public static final String KEY_SUBTYPE = "subtype";

public static final String KEY_FABRIC = "fabric";

public static final String KEY_QUANTITY = "quantity";

public static final String KEY_PRICE = "price";

```

```

public static final String KEY_TOTALPRICE = "totalprice";

public static final int COL_TYPE = 1;

public static final int COL_SUBTYPE = 2;

public static final int COL_FABRIC = 3;

public static final int COL_QUANTITY = 4;

public static final int COL_PRICE = 5;

public static final int COL_TOTALPRICE = 6;

public static final String[] ALL_KEYS = new String[] {KEY_ROWID, KEY_TYPE,
KEY_SUBTYPE,      KEY_FABRIC,      KEY_QUANTITY,      KEY_PRICE,
KEY_TOTALPRICE};

public static final String DATABASE_NAME = "MyDb";

public static final String DATABASE_TABLE1 = "inventory";

public static final int DATABASE_VERSION = 6;

private static final String DATABASE_CREATE_SQL1 =

"create table " + DATABASE_TABLE1+ " (" + KEY_ROWID + " integer primary key
autoincrement, "+ KEY_TYPE + " text not null, "+ KEY_SUBTYPE + " text not null, "

+ KEY_FABRIC + " text not null, "+ KEY_QUANTITY + " integer not null, "+
KEY_PRICE + " integer not null, "+ KEY_TOTALPRICE + " integer not null"+ ");";

private final Context context;

private DatabaseHelper myDBHelper;

private SQLiteDatabase db;

public RetailDB(Context ctx) {

this.context = ctx;

myDBHelper = new DatabaseHelper(context);

```

```

}

public RetailDB open() {

db = myDBHelper.getWritableDatabase();

return this;

}

public void close() {

myDBHelper.close();

}

public long insertRow1(String type, String subtype, String fabric, int quantity, int price, int
totalprice) {

ContentValues initialValues = new ContentValues();

initialValues.put(KEY_TYPE, type);

initialValues.put(KEY_SUBTYPE, subtype);

initialValues.put(KEY_FABRIC, fabric);

initialValues.put(KEY_QUANTITY, quantity);

initialValues.put(KEY_PRICE, price);

initialValues.put(KEY_TOTALPRICE, totalprice);

return db.insert(DATABASE_TABLE1, null, initialValues);

}

public boolean deleteRow1(long rowId) {

String where = KEY_ROWID + "=" + rowId;

return db.delete(DATABASE_TABLE1, where, null) != 0;

}

```



```

public void deleteAll1() {

    Cursor c = getAllRows1();

    long rowId = c.getColumnIndexOrThrow(KEY_ROWID);

    if (c.moveToFirst()) {

        do {

            deleteRow1(c.getLong((int) rowId));

        } while (c.moveToNext());

    }

    c.close();

}

public void deleteAll3() {

    Cursor c = getAllRows3();

    long rowId = c.getColumnIndexOrThrow(KEY_ROWID);

    if (c.moveToFirst()) {

        do {

            deleteRow3(c.getLong((int) rowId));

        } while (c.moveToNext());

    }

    c.close();

}

public int sumAllPrice1(){

    int total = 0;

```

```

Cursor cursor = db.rawQuery("SELECT SUM("+KEY_TOTALPRICE+") FROM
"+DATABASE_TABLE1+"", null);

if (cursor.moveToFirst()) {

total = cursor.getInt(0);

} while (cursor.moveToNext());

return total;

}

public Cursor getAllRows1() {

String where = null;

Cursor c = db.query(true, DATABASE_TABLE1, ALL_KEYS,

where, null, null, null, null, null);

if (c != null) {

c.moveToFirst();

}

return c;

}

public Cursor getRow1(long rowId) {

String where = KEY_ROWID + "=" + rowId;

Cursor c = db.query(true, DATABASE_TABLE1, ALL_KEYS,

where, null, null, null, null, null);

if (c != null) {

c.moveToFirst();

}

```

```

return c;

}

public boolean updateRow1(long rowId, String type, String subtype, String fabric, int
quantity, int price, int totalprice) {

String where = KEY_ROWID + "=" + rowId;

ContentValues newValues = new ContentValues();

newValues.put(KEY_TYPE, type);

newValues.put(KEY_SUBTYPE, subtype);

newValues.put(KEY_FABRIC, fabric);

newValues.put(KEY_QUANTITY, quantity);

newValues.put(KEY_PRICE, price);

newValues.put(KEY_TOTALPRICE, totalprice);

return db.update(DATABASE_TABLE1, newValues, where, null) != 0;

}

public int sum1(String type1,String subtype1,int del){

int total = 0;

Cursor cursor = db.rawQuery("SELECT (" +KEY_QUANTITY+") FROM
"+DATABASE_TABLE1+" WHERE "+KEY_TYPE+"=" +type1+" AND
"+KEY_SUBTYPE+" = "+subtype1+"", null);

total=cursor.getInt(0);

if(total<=del)

{

deleteRow1(cursor.getPosition());

```

```

    }

    return total;

}

private static class DatabaseHelper extends SQLiteOpenHelper {

    DatabaseHelper(Context context) {

        super(context, DATABASE_NAME, null, DATABASE_VERSION);

    }

    @Override

    public void onCreate(SQLiteDatabase _db) {

        _db.execSQL(DATABASE_CREATE_SQL1);

    }

    @Override

    public void onUpgrade(SQLiteDatabase _db, int oldVersion, int newVersion) {

        Log.w(TAG, "Upgrading application's database from version " + oldVersion

        + " to " + newVersion + ", which will destroy all old data!");

        _db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE1);

        onCreate(_db);

    }

}

```

REFERENCES

1. “Android Developers”, www.developers.android.com
2. “Eclipse”, www.eclipse.org