Cairo University

Faculty of engineering

Electronics and Communications department

Checkout system using RFID

By

Gehan Ramadan Mostafa

Hanan Ali Abu setta

Noha Samir Ghopashy

Rehab Ragab Abd El_Maaboud

Under the Supervision of

Dr. Hassan Mustafa

A Graduation Project Report Submitted to
the Faculty of Engineering at Cairo University

In Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Science
in
Electronics and Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

July 2015

ACKNOWLEDGMENTS

We would like to express the deepest appreciation to **Dr. Hassan Mustafa**; he continually and convincingly conveyed a spirit of hard working in regard to research, also for his great effort, without his guidance and persistent help this project would not have been possible.

We would like to thank our department staff, whose work with us through the past five years demonstrated the value of knowledge and hardworking to be the way to achieve targets accomplish dreams as well, also great thanks to graduates of last year who helped us with their knowledge.

It is an honor for us to be sponsored by "**National Telecommunications Regulatory Authority** - **NTRA**". We would like to thank NTRA staff for the financial support and their efforts with us through this full year. Our special thanks are extended to the staff of "**OG-TECH**" company for their technical support .We owe our deepest gratitude to them for allowing access and use of their equipment, especially Eng.Magued Morad , Eng.Ahmed Kandil and Eng.Menna Esmat.

We can never forget to mention that we are all owe our deepest gratitude to our parents, Any words would never express this immense gratitude for their long life helping, supporting, motivating and encouraging us till this moment had come.

In addition, special thanks to our beloved country Egypt, and for those people who really love it. And we swear that we will do our best to push it forward till Egypt become one of the leading countries.

ABSTRACT

Queues and long waiting periods is considered one of the most common complaints in supermarkets. Shoppers are interested in getting the right product at the right time at the right price. Imagine that the wait time is important, customers can with fewer items are waiting for a long time waiting. Supermarkets are using several techniques to reduce the waiting period to get out. It's used technology self or hybrid checkout system so as to reduce the waiting period. This technology allows stores to build a detailed picture of the shopping habits of each client, in exchange for discounts or cash vouchers. But as with everything in the tech world, the trend is towards smaller and cheaper devices. Radio tags track purchases by small operations can be commonplace on supermarket items within the next decade.

In our project, we needed RFID and tags; we needed basic skills in programming using C#. We needed also to know how to deal with database and connect it with C# using SQL Server. We also needed to make user requirement analysis in making the graphical user interface of our project. In order to solve the problem, an intelligent RFID checkout system has been developed. **The aim of this project** is to design a complete strategy for payment automatically. One of the most important advantages of our proposed system is the ability of identification. So our system provides both practically important control information, prevent stealing and data collection. The system also has the ability of **detecting important** over a large wide area. So that, we use this intelligent technology.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Problem definition

Shopping at the moment has a lot of problems:

- There are many customers spend boring time in queues to pay.
- Difficulties encountered in manual inventory, including the number and type of products in the market and goods in securities and the purchase of goods.
- The difficulty of finding sauce because of the use of human supervision.

Our project relies on RFID technology, a smart market, and markets in the future and markets in the future will be more advanced its technology to give greater prosperity for customers and more efficient control and supervision of goods for the owners of the market.

Radio Frequency Identification (RFID) is an emerging technology which is still unexplored exit system. It can be used to identify products and sorted through the use of RFID technique.

Goal and Objectives

The aim of our project (checkout system using RFID) to

Project Overview

The next state diagram is simple about our proposed system to show how our system works and deals with different components in the system as shown in Fig.1.1.

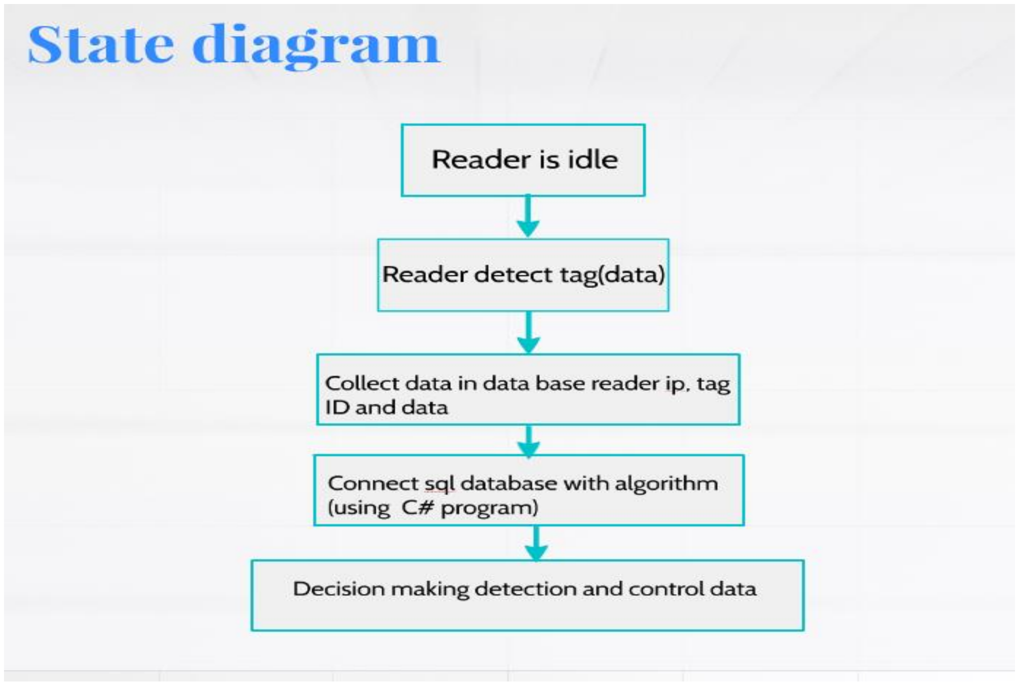

State diagram
Reader is idle
Reader detect tag(data)
Collect data in data base reader ip, tag ID and data
Connect sql database with algorithm (using C# program)
Decision making detection and control data

Fig1.1 state diagram

As shown above, a number of readers are deployed to detect and count items at each cart. The reader captures the Tag ID, the in-field time, and out-fields time for each cart passing within its range, then, all information are collected in a database. This database is connected to our algorithm using C# program which, by its turn is responsible about decision making according conditions.

Project Outcome

The system architecture consists of 2 RFID readers, passive tags and a personal laptop which is our processing unit. We are using C# program connected to database to implement our smart algorithm and is responsible about decision making according to detection conditions.

Chapter 2: RFID

2.1 What is RFID?

Radio frequency identification (RFID) is a wireless communication technology that is used to identify tagged objects or people. This technology uses radio waves to transmit Information describes identity, location, and/or condition of physical object. In recent years automatic identification procedures (Auto-ID) have become very popular in many service industries, purchasing and distribution logistics, manufacturing companies and material flow system. These technologies exist to provide information about people, animals, goods and products in transit. Auto-ID technologies include OCR, Barcodes systems, Smart Cards and some biometric technologies, such as retinal scans. [Figure1.1]These technologies are used to reduce time and labor needed for manually data entry and to enhance data accuracy. Some Auto-ID, like Barcode systems require a person manually scan label to capture the data. Barcode may be extremely cheap, their storage capacity is low and cannot be programmed. But RFID has many benefits than Barcode systems will discuss latte.

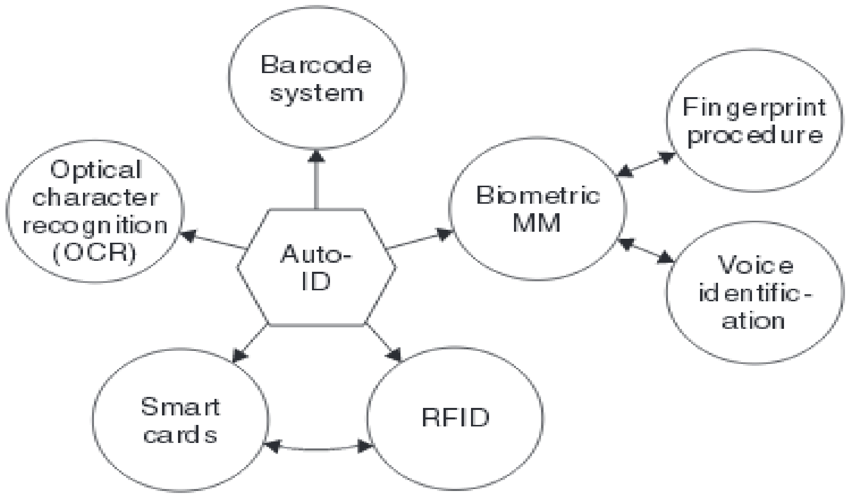


Figure 2.1: Overview of the most important auto-ID procedures

2.2 Automatic Identification and Data Capture (AIDC) Systems

2.2.1 Barcode Systems

Barcode systems are go one better than other identification systems over the past 20 years. The turnover volume for barcode systems totaled around 3 billion DM in Western Europe at the beginning of the 1990s.The barcode is a binary code comprising a field of bars and gaps arranged in a parallel configuration. They are arranged according to a predetermined pattern and represent data elements that refer to an associated symbol. The sequence made up of wide and narrow bars and gaps, can be interpreted numerically and alpha numerically. Barcode is read by optical laser scanning i.e. by the different reflection of a laser beam from the black bars and white gaps. The most popular barcode by some margin is the EAN code (European Article Number), which was designed specifically to fulfill the requirements of the grocery industry in 1976.

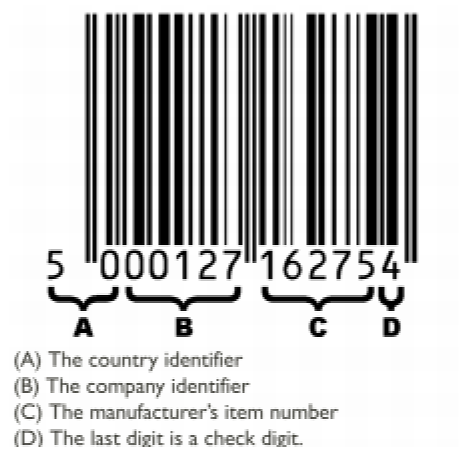

5 000127 162754
A
B
C
D
(A) The country identifier
(B) The company identifier
(C) The manufacturer's item number
(D) The last digit is a check digit.

Figure 2.2: the structure of a barcode in EAN-13

Common barcodes with typical applications

- **Code Coda bar**: medical/clinical applications, fields with high safety requirements.
- **Code 2/5 interleaved**: automotive industry, goods storage and pallets.
- **Code 39**: processing industry, logistics, universities and libraries.

2.2.2 Optical Character Recognition (OCR)

The first use of Optical Character Recognition was in 1960. Special fonts were developed for this application that stylized characters so that they could be read both in the normal way by people and automatically by machines. The advantage of OCR is the high density of information and the possibility of reading data visually in an emergency and checking. Now, OCR is used in production, service and administrative fields, and also in banks for the registration of cheques (personal data, such as name and account number, is printed on the bottom line of a cheque in OCR type). But, OCR systems not universally applicable because of their high price and the complicated readers that they require in comparison with other ID technologies.

2.2.3 Biometric Procedures

Biometrics is the science of counting and (body) measurement procedures involving living beings. Biometry in the identification systems is the general term for all procedures that identify people by comparing unmistakable and individual physical characteristics. In practice, these are fingerprinting and hand printing procedures, voice identification and, less commonly, retina identification.

2.2.4 Smart Cards

A smart card is an electronic data storage system, possibly with additional computing capacity (microprocessor card), which is incorporated into a plastic card the size of a credit card and that for convenience. The first smart cards in the form of prepaid telephone smart cards were launched in 1984. They are placed in a reader, which makes a galvanic connection to the contact surfaces of the card using contact springs. Smart card is supplied with energy and a clock pulse from the reader via the contact surfaces. Data transfer between the reader and the card takes place using a bidirectional serial interface. the primary advantages of the smart card is that the data stored on it can be protected against undesired (read) access and manipulation. Smart cards make all services that relate to information or financial transactions simpler, safer and cheaper. One disadvantage of contact-based smart cards is the vulnerability of the contacts to wear, corrosion and dirt. Readers that are used frequently are expensive to maintain due to their tendency to malfunction. Also, readers that are accessible to the public (telephone boxes) cannot be protected against vandalism.

2.2.5 RFID Systems

RFID systems like smart card systems, data is stored on an electronic data -carrying device- the transponder. But, RFID unlike smart card in data transfer since the data exchange between the data-carrying device and the reader are achieved without the use of galvanic contacts, using instead magnetic or electromagnetic field. The underlying technical procedure is drawn from the fields of radio and radar engineering.

Due to the several advantages of RFID systems compared with other identification systems, RFID systems are now beginning to conquer new mass markets. For example, the use of contactless smart cards as tickets for short-distance public transport.

2.3 Why RFID?

RFID is becoming increasingly prevalent as exists in some systems. Recent developments include increased read ranges and accuracy and increased performance. Declining hardware and infrastructure costs and greater abilities to integrate systems make now a great time to invest in RFID. Return on investment from RFID hasn't higher, more immediate, and available to more industries. All across the world, Radio Frequency Identification have been helping businesses improve their processes and increase their efficiency. Signification process has been made in the last few years in creating global and industry standards for RFID, signification process has been made in the last few years in creating global and industry standards for RFID, therefore increasing compatibility and return on investment of investments in RFID. RFID costs are still extremely high for many applications, but it will at least. While the costs may is quite high to be published in a production environment.

Rapid progress in the RFID technology has made it probably to manufacture RFID tags in many shapes and sizes. Also, the information written on the RFID tags cannot be duplicated easily. RFID tags are thus, better than the barcode because easy duplication is impossible and can be more effectively for security purposes. Previously, the barcode system used for security purposes has known that the failure of many times.

2.4 The Core Components of RFID

Before understanding RFID, we must understand how Radio Frequency communication occurs. RF (Radio Frequency) is connection occurs by transferring data over electromagnetic waves. By generating electromagnetic wave at specific source, its effect can be watched at the receiver far from the source, which determines thereafter and thus the information. In an RFID system, the RFID tag is include the tagged data of the object generates a signal containing the respective information which is read by the RFID reader, then may pass this information to a processor for processing the get information for that particular application.

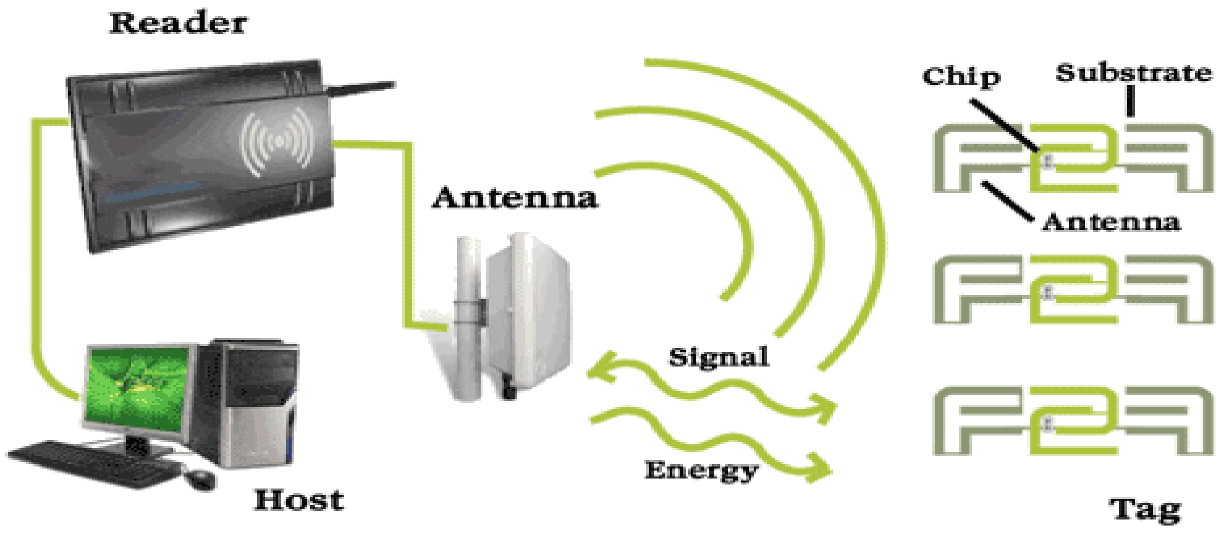


Figure2.3 Components of RFID system

Figure2.3 Components of RFID system

Thus, an RFID system composed of three components:

1. RFID tag or transponder
2. RFID reader or transceiver
3. Data processing subsystem

An RFID tag is consists of an antenna, a wireless adapter, and an encapsulating material. These tags can be either passive or active. While passive tags use the power resulting from the magnetic field of the RFID reader, the active tags have on-chip power. Thus passive tags are cheaper but with lower range (<10mts) and more sensitive to regulation and environmental restriction, as compared to active tags.

An RFID reader composed of an antenna, decoder, and transceiver; it sends periodic signals to ask for any tag in the vicinity. On receiving any signal from a tag, it passes on that information to the data processor. The data processing subsystem provides the device of storing and processing the data. RFID systems can differentiated based on the frequency range it uses. The ranges of Low-Frequency (LF: 125 - 134.2 kHz and 140 - 148.5 kHz), High-Frequency (HF: 13.56 MHz) and Ultra-High-Frequency (UHF: 868 MHz - 928 MHz).

Low-frequency systems have short reading ranges and lower system costs, are used in asset tracking, animal identification applications and security access. High-frequency systems, long reading range (greater than 90 feet) and high reading speeds, are used in some applications such as automated toll collection and railroad car tracking. While, the higher performance of high-frequency RFID systems bears higher system costs.

2.4.1 RFID Tag (transponder)

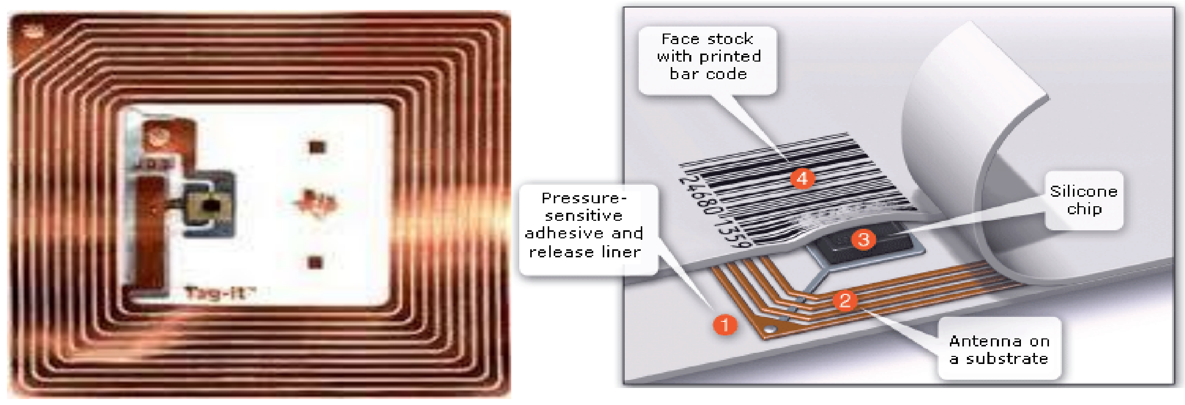

Tag-it™
Face stock with printed bar code
Pressure-sensitive adhesive and release liner
Silicone chip
Antenna on a substrate
1
2
3
4

Figure 2.4 RFID Tag

The basic function of an RFID tag is to store data and transmit data to the integrator. Most RFID tags are made of two main parts. The first part, antenna is receiving radio frequency (RF) waves and the second part, an integrated circuit is used to processing and storing data, as well as modulating and demodulating the radio waves received/sent by the antenna. Generally, the chip contains memory where data may be stored and read from and sometimes written, too, in addition to other important circuitry. Some tags also contain batteries, and this is what differentiates active tags from passive tags.

A- According to Power

I. **Active tags**

   Active RFID tags have transmitter and power source onboard the tag, such as a battery. These are mostly UHF solutions and can read ranges up to 100 m in some instances, their long read range makes active RFID tags for many industries and more expensive than their passive counterparts and are used to track large assets   (like vehicles, cargo containers, and machines). Active RFID tags equipped with sensors are often to measure and transmit temperature, light, humidity, and shock/vibration data for the objects. There are two types of active tags. Transponders only and transmit data when receiving a radio signal from a reader. For example, checkpoint control location would be active when passing through a particular gate.

II. Passive tags

Passive RFID tags don’t possess on-board power source. In the typical passive RFID tag design, it cannot communicate with host applications if it is without the range of an RFID reader. Instead, they are powered by the electromagnetic energy transmitted from the RFID reader. Because the radio waves must be strong enough to power the tags, the reader and reader antenna transmit a signal to the tag, and that signal is used to power on the tag and reflects energy back to the reader. Read ranges that shorter than ranges of active RFID tags, smaller, less expensive, and more flexible than active tags. Passive RFID tags have a read ranges up to25 m. There are passive LF, HF, and UHF systems. For example, Passive UHF tags are used for item-level tracking of consumer goods and pharmaceuticals.

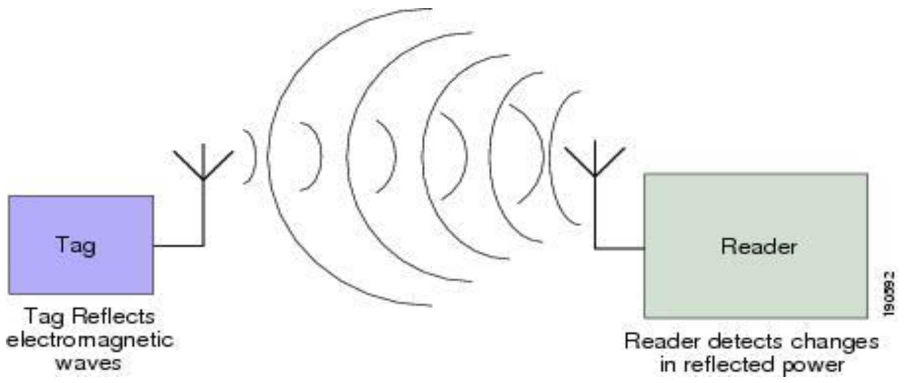

Tag
Tag Reflects electromagnetic waves
Reader
190592
Reader detects changes in reflected power

Figure2.5 Passive Tag Backscatter Modulation

II. Battery-Assisted Passive (BAP) or Semi-passive tags

III. BAP systems, or semi-passive RFID systems, include a power source information of a passive tag. Semi-passive tags have an onboard power source and may have onboard sensors. The onboard power source serves two purposes:

1. It provides continuous power for the sensors.
2. It allows the intelligence contained in the chip to function without harvesting energy.

The onboard power supply ensures that the semi-passive tag can take these readings even in the absence of a reader to power the tag. a semi-passive tag is powered by the battery, the semi-passive tag is not required to harvest energy for operation from the reader signal. Hence, these tags do not need to harvest energy to power the circuitry for the backscatter communication to produce a stronger signal that is easier for the reader to detect. Longer tag transmission intervals increase tag battery life but as tag RSSI is reported less often, the location frequency will be less. A shorter tag transmission intervals decrease tag battery life but tag location may be improve accuracy in some cases. Other uses of semi-passive tags increase the read range or reading the tag in an unfriendly environment. These tags typically make use of reflect and backscatter modulation electromagnetic energy from the RFID reader to generate a tag response similar to passive tags (see Figure2.6).

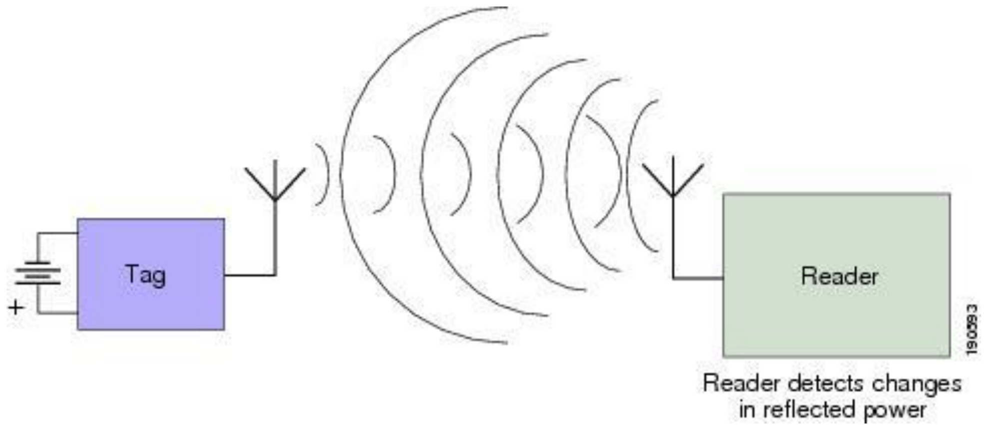


Figure2.6 Backscatter Modulation in Semi-Passive RFID Tags

Feature	Active	Passive	Semi – Passive
Read Range	Long (up to 100m)	Short (up to 10m)	Long (up to 100m)
Storage	128 Kbytes read/ write	128 bytes read/ write	128 Kbytes read/ write
Battery	Yes	No	Yes
Availability	Continuous	Only in field of reader	Only in field of reader
Lifespan	Between 5-10 years	Up to 20 years	Up to 10 years
Coast	Very expensive	Cheap	Expensive
Application	Monitor the condition of fresh produce	EZ-Pass toll payment booths	Measurement of temperature periodically

Table2.1 Features of Types of Tags

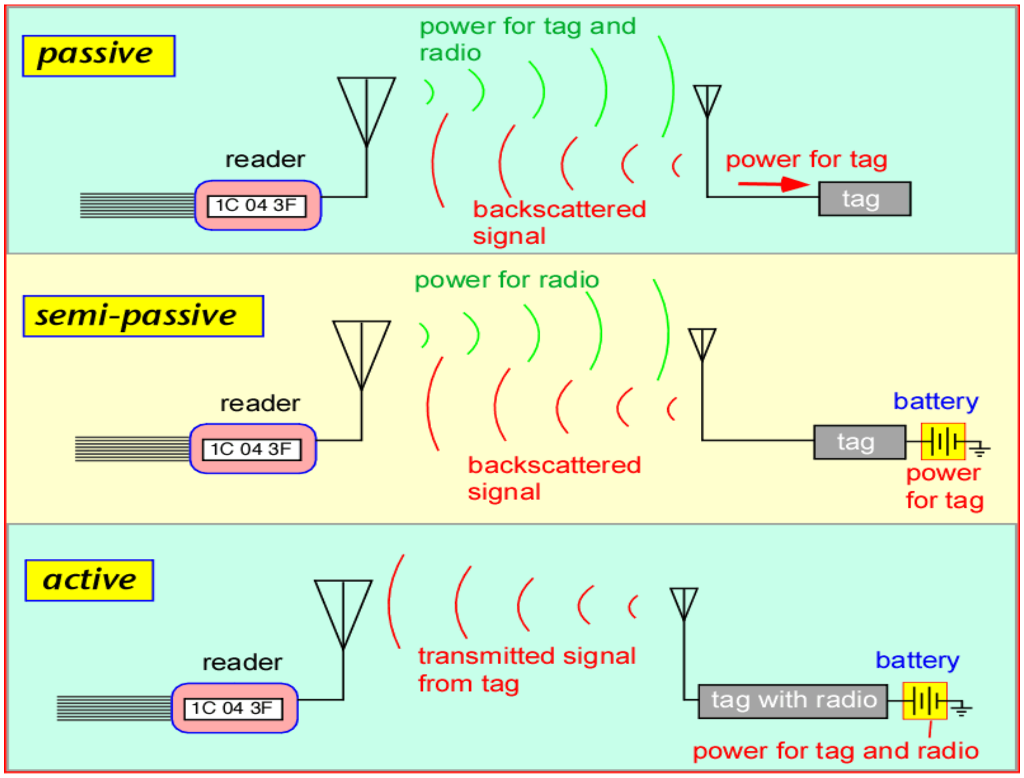


Figure2.7 Types of RFID Tag

B- According to memory

RFID tags may be able to a read-only RFID tag, or a read-write RFID tag. The function of RFID tags is either a read-only RFID tag RFID tags may be able to a read-only RFID tag, or a read-write RFID tag. Because of the cost of manufacturing different types against the quantities made and differences between them, in today more RFID tags are the read-write variety, and the applications which only a read function is used, do not used the writing.

Read only tags (ROM): Data will be a unique identifier and other specified data that cannot be changed, the information is programmed onto chip during manufacturing, the information on read-only chips cannot be changed, often referred to as a 'number plate' application, information constant, the limited processing capability ,and least expensive.

Read-write RFID tags (ROM + EEPROM): When the tag is within range of the reader, the user can add information to the tag or write over existing information but these chips are more expensive that Read-only chips. there is another way used is something, it can be written and then becomes Read-only but can be read many times, it is called a "WORM" chip (Write Once Read Many), the better processing capability.

2.4.2 Antenna

The antenna is medium where communication between the tag and reader occur with each other. It can transfer data and activate a passive tag by emitting wireless impulses that have electromagnetic properties. The antenna comes in different designs, they come from different types: Stick antennas, Di-pole or multi-pole antennas, phased-array element antennas, Circular polarized, Gate antennas, Patch antennas, Linear polarized, and Adaptive antennas.

2.4.3 Reader

The reader is the most fundamental part of the RFID system. The reader can communicate by transferring beam of impulse, which encapsulates commands to the tag and listens for the tag's response. Attempts of the reader to interrogate the tags at varying frequencies, allows the reader to read multiple tags at same time. The reader is connected to the computer for data processing via a USB cable (RS-232, and RS485) or over a wireless connection. it can have anti-collision algorithm and another critical function procedures for deducting multiple tags at one time and a single reader can operate on multiple frequencies. The reader can send information in two directions: it can read information from a tag and send it to the PC (read mode), or Conversely (Write mode), this can be called coupler. Readers (interrogators) can be at a fixed point, such as Entrance/exit and Point of sale. Readers can also be mobile /handheld.

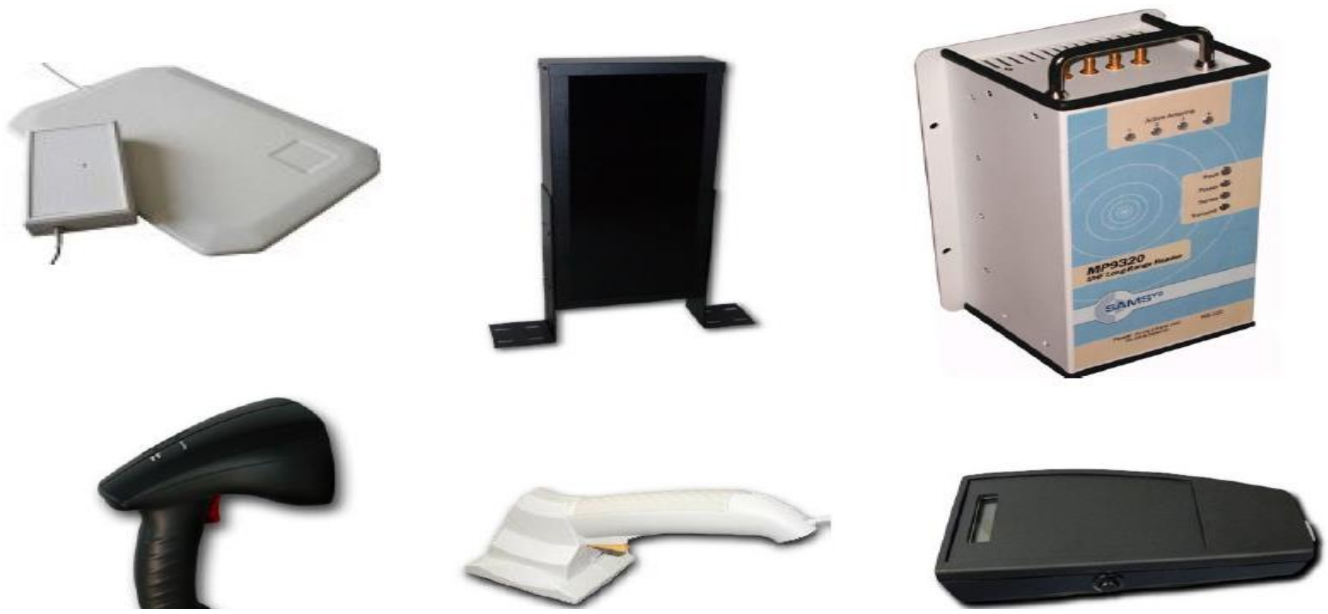


Figure2.8 Some RFID readers

2.4.4 Host Computer

PC provides an interface between the RFID hardware and application based system, which is the “brain” of any RFID system. They are used to network multiple RFID interrogators together and to centrally process information.

2.5 RFID Frequencies

Frequency refers to the size of the radio waves used to communicate between RFID system components .As television can be broadcast in VHF or UHF band, so too RFID systems exist on various frequency ranges as shown in Table 1.2

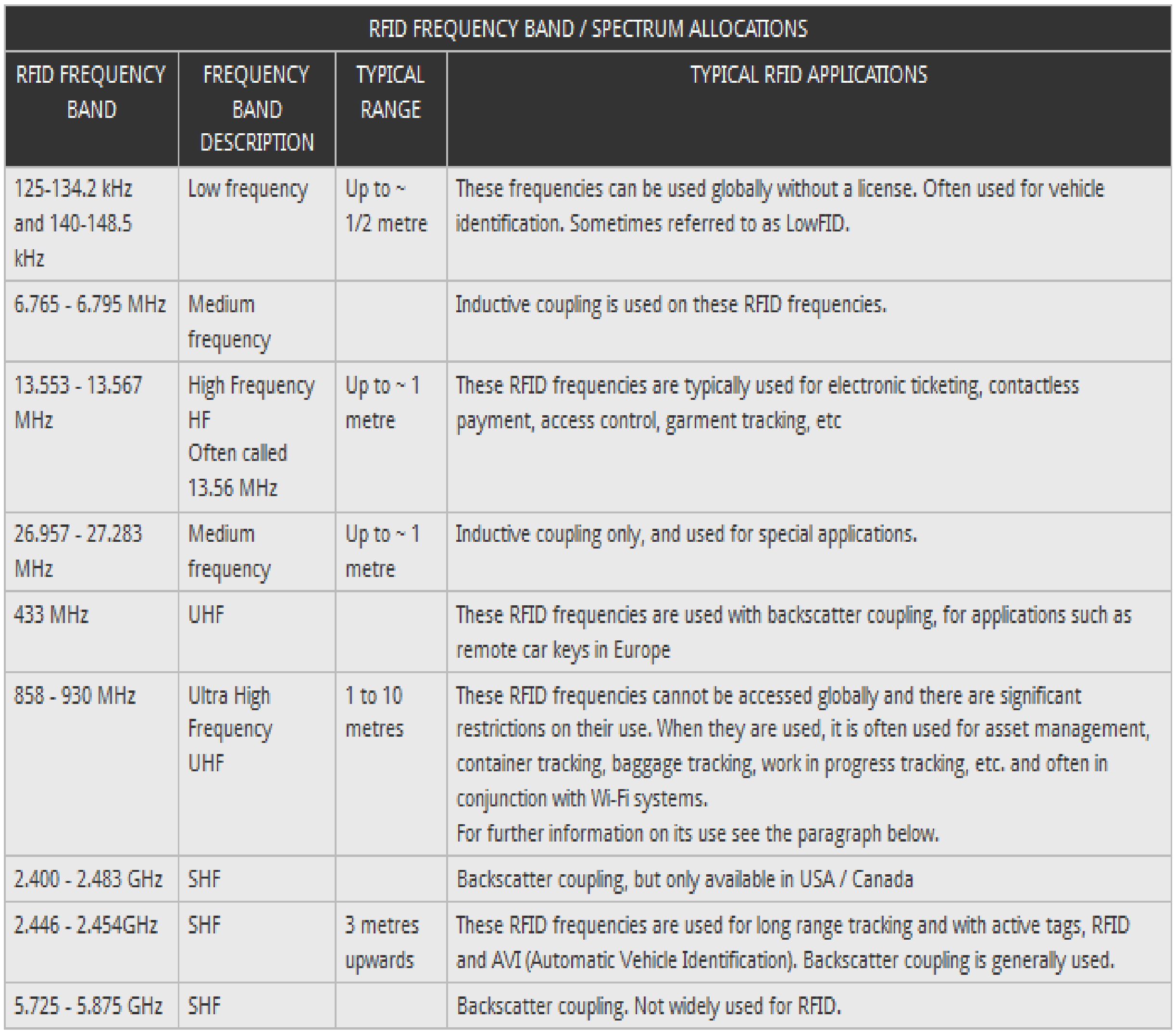
2.5.1 Main frequency bands for RFID systems

1. **Low Frequency RFID Bands**

   - Low Frequency (LF): 125–134 KHz
   - High Frequency (HF): 13.56 MHz

2. **High Frequency RFID Bands**

   - Ultra-High Frequency (UHF): 860–960 MHz
   - Microwave: 2.5 GHz and above



RFID FREQUENCY BAND / SPECTRUM ALLOCATIONS			
RFID FREQUENCY BAND	FREQUENCY BAND DESCRIPTION	TYPICAL RANGE	TYPICAL RFID APPLICATIONS
125-134.2 kHz and 140-148.5 kHz	Low frequency	Up to ~ 1/2 metre	These frequencies can be used globally without a license. Often used for vehicle identification. Sometimes referred to as LowFID.
6.765 - 6.795 MHz	Medium frequency		Inductive coupling is used on these RFID frequencies.
13.553 - 13.567 MHz	High Frequency HF Often called 13.56 MHz	Up to ~ 1 metre	These RFID frequencies are typically used for electronic ticketing, contactless payment, access control, garment tracking, etc
26.957 - 27.283 MHz	Medium frequency	Up to ~ 1 metre	Inductive coupling only, and used for special applications.
433 MHz	UHF		These RFID frequencies are used with backscatter coupling, for applications such as remote car keys in Europe
858 - 930 MHz	Ultra High Frequency UHF	1 to 10 metres	These RFID frequencies cannot be accessed globally and there are significant restrictions on their use. When they are used, it is often used for asset management, container tracking, baggage tracking, work in progress tracking, etc. and often in conjunction with Wi-Fi systems. For further information on its use see the paragraph below.
2.400 - 2.483 GHz	SHF		Backscatter coupling, but only available in USA / Canada
2.446 - 2.454GHz	SHF	3 metres upwards	These RFID frequencies are used for long range tracking and with active tags, RFID and AVI (Automatic Vehicle Identification). Backscatter coupling is generally used.
5.725 - 5.875 GHz	SHF		Backscatter coupling. Not widely used for RFID.

Table2.2: RFID frequency band

2.5.2 How to choose suitable frequency for RFID system?

Choice of frequency affects some characteristics of RFID system.

1) **Read range:**

In lower frequency bands, the read ranges of passive tags are no more than a couple feet, due primarily to poor antenna gain. (At low frequencies, electromagnetic wavelengths are very high, and much longer than the dimensions of the antennas integrated into RFID tags. Antenna gain is directly proportional to antenna size relative to wavelength. Hence, antenna gain at these frequencies is very low).but, at higher frequencies, the read range increases, especially where active tags are used. however, FCC have posed power limits on UHF and microwave systems and this has reduced the read range of these high frequency systems because the high frequency bands pose some health concerns to humans, most regulating bodies.

2) Types of tags (passive tags or active tags):

For historical reasons, passive tags are typically operated in the LF and HF bands, whereas active tags are typically used in the UHF and microwave bands.

3) Presence of liquid or metal:

The RFID systems are affected by water or wet surface. Using HF with wet surface is better than UHF because of its high wavelength so it can penetrate wet surfaces. Signals in the high frequency bands are more likely to be absorbed in
Liquids and cause degradation of performance for the RFID system.as a result, HF tags are better choice for tagging liquid containers.
Radio frequency cannot penetrate metal because it has an electromagnetic reflector. But, the near presence of metal can have adverse effects on the operation of a system and when metal is placed near any antenna the characteristics of that antenna are changed like impedance properties, resonant frequency, and antenna gain and radiation pattern. The high frequency bands are affected by metal more than the lower frequency bands.in order to tag wet or metal surfaces, special precautions have to be taken, which ultimately drives up costs.
The influence of the nearby metal surface against the RFID tag can be reduced with using proper separation between them with Styrofoam and the distance is around 0.5 cm from the research outcome [Figure 2.9].

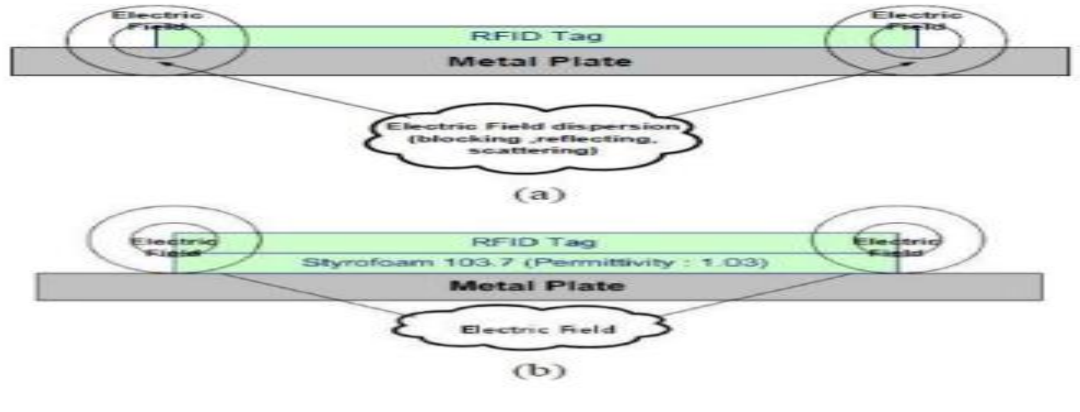


Figure 2.9: Simple RFID Tag Improvement against Metal platform

4) Antenna Size and Type:

Due to long wavelengths, the size of LF and HF antennas are larger than the size of UHF and microwave antennas in order to achieve comparable signal gain and this conflict with the goal of making RFID tags small and cheap.

5) Interference from Other Radio Systems:

RFID systems are prone to interference from other radio systems. At operating in
the LF band are particularly vulnerable, because of LF do not experience much path loss, or attenuate very little over short distances, in comparison to the higher frequencies. This means that the radio signals of other communication systems operating at nearly the same LF frequency will have high field strengths at the antenna of an RFID interrogator which can translate into interference. In microwave systems are the least susceptible to interference, as path loss in the microwave band is much higher than for the LF and generally a line of sight is required in order for microwave radiators to interfere.

6) Size and Price of RFID Tags:

Because of long wavelengths of low frequency radio signals. Antennas of LF and HF systems have to be made much larger than UHF and microwave antennas to achieve comparable signal gain. This conflict with the goal of making RFID tags small and cheap [Figure 2.10].
Shows the two types of RFID antenna/tag coupling concepts:

- Inductive coupling (LF and HF): inductive antennas are used, which are loop-type antennas.
- Capacitive coupling (UHF and Microwave): the antennas are of the dipole type.

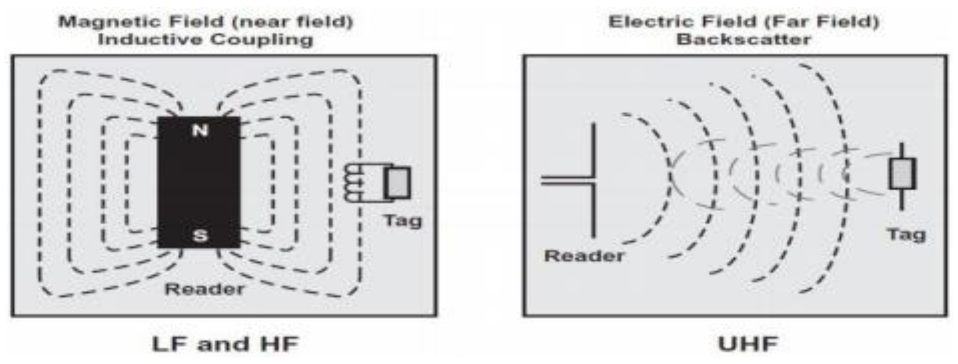


Figure 2.10 Two types of Antenna/Tag coupling

Early RFID systems used the LF band, due to the fact that LF tags are the easiest to manufacture. But , they have many disadvantages, such as large size which translates into a higher price at volume. The HF band is currently the most prevalent worldwide, because HF tags are typically less expensive to produce than LF tags. Recent advances in chip technology have brought prices for UHF tags down to the point that they are competitive with HF tags since The UHF band represents the present state of the art. Microwave RFID tags are similar to UHF tags in that they can be made smaller and ultimately cheaper.

2.6 Applications of RFID

Radio wave technology allows for a small RFID chip to be embedded in any physical object and uniquely identified by an RFID reader. We can use unlimited amount of possibilities for connected devices which tagged with RFID chips like goods, waste management systems, animals and even the humans.

2.6.1 Library

Today, over 5,000 libraries worldwide have already introduced RFID and millions of customers benefit from the technology. RFID makes life easier and for libraries and visitors through fast check out , easier returns, simpler inventory, and extra security. For example, help libraries effectively manage their circulation and collections, while providing convenience for their customers.

2.6.2 Tracking

RFID can track items. For example Document Tracking , RFID tracks and traces documents which increased workforce productivity and reduced operational costs since it enables high-speed identification of documents (up to 700 items per second). Also, RFID tracking avoid kidnappings in the pediatric and maternity rooms. All babies have and RFID TAG in their ankle or wrist so, if the baby is near a hospital door and they still have the transponder, the door will close immediately.

2.6.3 Medical:

RFID has several applications in hospital and a big benefit in a healthcare. It ensures that medical personnel do not leave sponges inside patients after surgery, The unique number in the tag points to the medical record of the patient in the database, Useful when the patient is unconscious or for patients with chronic diseases and Other applications include the calibration and authentication of disposable medical devices (such as blood measurement cartridges).

2.6.4 Security

RFID systems can be used for all the keys at home, work and cars. Secure entrance can be guaranteed by only allowing “tagged” personnel.the log of the interances can be hold in the database.
There is also Fast & Secure Credit Payment which Instead of the regular card slip, customers will use their Pay pass to “tap” the reader. The card or key fob sends payment details to the reader through wireless carrier. The reader transmits this information to an intermediate terminal and then to the MasterCard network to finish the transaction [Figure 1.4].

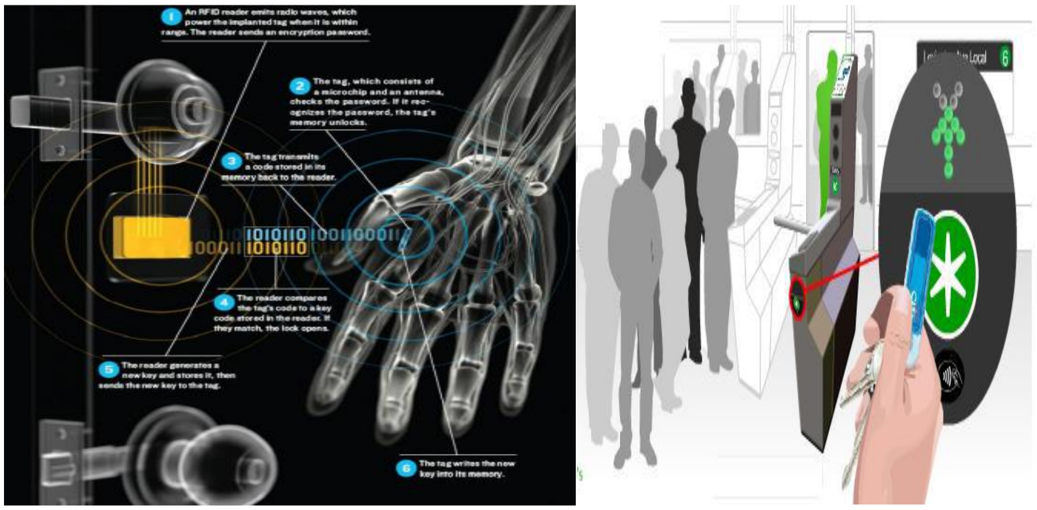
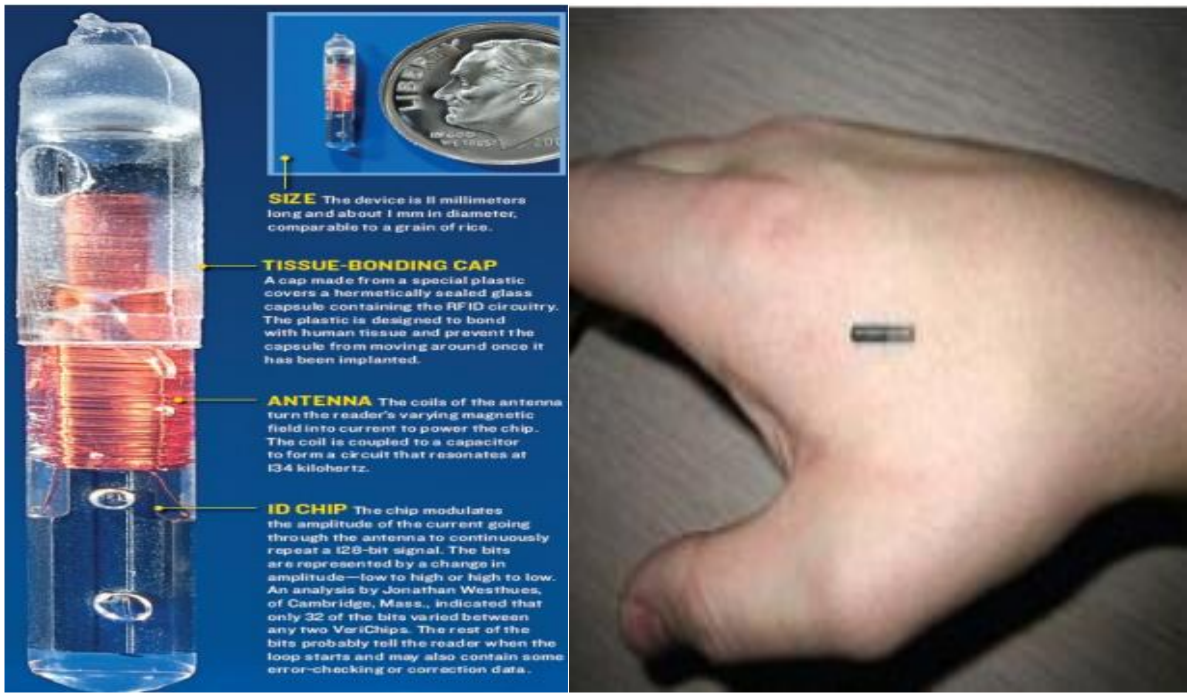

1 An RFID reader emits radio waves, which power the implanted tag when it is within range. The reader sends an encryption password.
2 The tag, which consists of a microchip and an antenna, checks the password. If it recognizes the password, the tag's memory unlocks.
3 The tag transmits a code stored in its memory back to the reader.
4 The reader compares the tag's code to a key code stored in the reader. If they match, the lock opens.
5 The reader generates a new key and stores it, then sends the new key to the tag.
6 The tag writes the new key into its memory.
1010110 10011000
00011 1010110

Figure 2.11 secure iterance using RFID

2.6.5 Technical Aspects:

1) **VeriChip**: VeriChip suggests the chip be inserted in the rear part of the triceps of the right arm under the skin. The chip doesn’t contain any medical records, but its 16-digit number could be linked to a database of patient medical information. When the tag is scanned, the number could be quickly cross-referenced to reveal specific medical data about the patient.[Figure 1.6]
2) **Accurate Timing for Athletes**: The reader and its antenna are embedded in the carpet at the finish lines or check points.

3) **Smart Fridge**: The food items should be tagged by the food suppliers. The fridge will monitor the inventory levels and notify the user for the items running low and transmit this data to the cell-phone of the user through a Bluetooth link. A user interface will generate possible recipes or shopping lists based on the data the fridge provides.



SIZE The device is 11 millimeters long and about 1 mm in diameter, comparable to a grain of rice.

TISSUE-BONDING CAP A cap made from a special plastic covers a hermetically sealed glass capsule containing the RFID circuitry. The plastic is designed to bond with human tissue and prevent the capsule from moving around once it has been implanted.

ANTENNA The coils of the antenna turn the reader's varying magnetic field into current to power the chip. The coil is coupled to a capacitor to form a circuit that resonates at 134 kilohertz.

ID CHIP The chip modulates the amplitude of the current going through the antenna to continuously repeat a 128-bit signal. The bits are represented by a change in amplitude—low to high or high to low. An analysis by Jonathan Westhues, of Cambridge, Mass., indicated that only 32 of the bits varied between any two VeriChips. The rest of the bits probably tell the reader when the loop starts and may also contain some error-checking or correction data.

Figure2.12 Vreichip

2.6.6 Retail:

Products can be tagged and use RFID to modify them .RFID makes retail faster and more security since it has a lot of advantages will discuss in chapter 3.

Chapter 3: Checkout System

3.1 Our problem

When people enter the supermarket the most thing draws their attention is the congestion at the cashier. We formed a survey about the most important problems facing the client during the process of shopping at the supermarket and the results showed that the greatest problem for everyone is the process of checkout. Access to the cashier is the most difficult stage in the process of checkout as a result of overcrowding and wait for a long time in endless queues. This endless because the cashier takes a long time to examine the products of each client and complete the checkout process where the cashier scans each of the elements using a barcode system.

Figure 3.1 Congestion at the cashier at supermarket.

Weisselburg and Coweley (1969); Foote (1976), Crahill, Gross and Magazine (1977) Jones, O'Berski and Gail (1980) produced studies Studies that Interested in the problems of retailers showed how to reduce the time spent waiting in queues of customers checkout.Research gap in previous studies is the limited amount of research in the selection methods checkout system that can help managers in their choice of choosing Appropriate checkout system for their stores.Retailers, especially in grocery stores, have seen decades of changes and developments for improving checkout systems. Today, grocery retailers are leaders in searching for effective ways of checkout in the retail sector.

We can determine the best exit system based on several factors such as speed, efficiency and ease of use for the customer and the store management.

3.2 Definitions:-

Cashier:-

Is the person who has a permission from retailers to hold the checkout process, which is responsible for the registration of goods sold, and to collect and keep records of customer's payment and complete the payment process with the customer.

Checkout:-

The checkout is the process of paying for the groceries, such as registering the products, making the monetary transaction, etcetera. The objective of checkout is to allow the products out of the store versus pay for it.

Checkout line:-

Is the location or queue customers go through to verify and pay for items selected for purchase in a retail store.

Checkout system:-

Is a method designed to checkout customers merchandise in a store.

Consumer:-

Is a person who buys goods or services in a retail store for personal use or for resale.

Customer:-

Is a person who shops or buys from or patronizes a retail store or a business regularly.

Grocery:-

Is food or supplies sold by a grocer (retail store).

Point of sales:-

Is a location or process relating to a place in a business or a retail store where sales are made and purchases checkout.

Retailer:-

Is a person who owns or manages a retail store that sells directly to the consumer.

Self- Checkout:-

Is a process or method in which a customer is only responsible for checking out his or her merchandize(s).

Cashpoint:-

The Cashpoint is a payment terminal that allows the customer to pay the groceries with coins or bank notes.

3.3 Checkout:-

The main reason for the checkout to exist is to allow the products from the store to be paid for. Checkout usually has a cash register to securely store the received money as well as the change money.

3.3.1 Some of the techniques used in the Checkout system:-

1) Checkout system using Barcode techniques.

2) Checkout system using RFID.

Barcode:-

Checkout system using Barcode

Barcodes are printed horizontal strips of vertical bars used for identifying specific items.
A scanning device reads the barcode by moving a beam across the symbol.

6583 3254

Figure 3.2 Barcode.

RFID:-

Checkout system using RFID:-

RFID is generally used to replace are barcodes and magnetic stripes. RFID unlike these technologies does not need contact or line of sight to be read. In order for barcodes to be read the tag must be clean, and correctly oriented to the reader. Magnetic stripes, such as those on credit cards must be swiped, Depending on the needs of the application, RFID tags are available in Passive and Active forms which have maximum read ranges of an inch to more than 75 feet. It is also important to understand the weaknesses of RFID. Because it relies on the propagation electromagnetic fields, the presence water and metal make it very difficult to create accurate systems. But even these weaknesses are being overcome at tags are developed for use in the steel industry. The reliability and time savings possible with RFID have led to its wide adoption in applications from passports to libraries and cars.

Figure 3.3 RFID tag.

Comparison between RFID & Barcode techniques

	RFID	Barcode
Read Rate	High throughput. Multiple (>100) tags can be read simultaneously.	Very low throughput. Tags can only be read manually, one at a time.
Line of Sight	Not required. Items can be oriented in any direction, as long as it is in the read range, and direct line of sight is never required.	Definitely required. Scanner must physically see each item directly to scan, and items must be oriented in a very specific manner.
Human Capital	Virtually none. Once up and running, the system is completely automated.	Large requirements. Laborers must scan each tag.
Read/Write Capability	More than just reading. Ability to read, writes, modify, and update.	Read only. Ability to read items and nothing else.
Durability	High. Much better protected, and can even be internally attached, so it can be read in very harsh environments.	Low. Easily damaged or removed; cannot be read if dirty or greasy.
Security	High. Difficult to replicate. Data can be encrypted; password protected, or includes a “kill” feature to remove data permanently, so information stored is much more secure.	Low. Much easier to reproduce or counterfeit.
Event Triggering	Capable. Can be used to trigger certain events (like door openings, alarms, etc.).	Not capable. Cannot be used to trigger events.
cost	More expensive: $0.10-plus cost to attach.	Cheaper to produce:$0.001.

Table 3.1: Comparison between RFID & Barcode

3.4 Registration and Add Products

In order to calculate the total amount of money a customer has to pay, each product is scanned into the cash register. This tells the register which product it is, and the register then searches the central product database (on a local computer server) for the product’s name and price. The price and name are added to the receipt that will be printed for the customer. To keep track of the store inventory, the turnover and lost products, the sold items are also registered in a separate database after the transaction. Modern day information technology is applied in many stores to automate the ordering of new product stock, a system which is heavily depending on the correct registering of all sold products at the checkouts.

3.5 RFID chips versus barcodes

Possibly the most obvious development in the market is the shift towards smart product tags. Small, cheap chips based on Radio Frequency Identification (RFID) technology are in development at numerous companies. These tags are aimed at replacing the currently widely applied barcodes on products, which are used for identification purposes. For instance, these barcodes are used in supermarkets to identify the products at the checkout, allowing for a quick processing of large quantities of products. The downside to barcodes is that they have to be held before a barcode scanner and that they are easily damaged (for instance by wrinkling the material they are printed on), rendering the code unreadable. RFID chips are small, cheap electronic chips that transmit a radio signal on a specific frequency. The chips, and the products they are attached to, can be identified by this frequency. The RFID chips can be read from a distance through other materials. This is a great advantage over barcodes. The downside of RFID is that the radio signals are continuously transmitted, which could be a breach of the customer’s privacy. A more pressing issue is the costs of the chips, which is a very high compared to the costs of printing a barcode on a product or package. Generally it is expected that RFID chips will become less expensive in the future. However, the novelty of the technology and the relatively high costs has prevented the technology to break through as a mainstream product. In the current generation of supermarkets, the barcode is still the preferred type of product identification despite its shortcomings compared to RFID chips. Scan point’s self -checkout solutions are currently optimized for barcodes. Support for RFID may be incorporated in the future, but is currently not yet requested by the clients and thus not yet implemented in the hardware and software.

3.6 The Electronic Product Code Standard:-

The EPC standard is similar to the barcodes UPC symbology. The main difference is that the EPC standard's ability to assign a unique number to each individual item allowing it to differentiate between the 24th packet of chocolate from the 1024th. An EPC contains 96 bits of information, including a serial number, "an EPC acts – like a URL - as a reference to a document", which exists on a network. The UPC symbology on the other hand is the symbology used by barcodes throughout the American grocery industry. The UPC symbology was developed primarily for point of- sale price look-up and does not allow individual products to be uniquely identified.

Figure 3.4 EPC tag

<u>Types of checkout system:-</u>

1) Traditional checkout (cashier)
2) Self-checkout

3.7 Checkout process is divided into two types according to the scan point:-

1) Fixed scan point(Scanner fixed on the gate)
2) Moveable scan point(Scanner fixed on the cart)

3.7.1 Fixed scan point:-

In this type the client roam the supermarket and collects what he needs from the goods in the cart, and when the ends of the assembly of goods go to the Scanner (which is fixed on the gate) to register what he had bought, Once the customer's shopping cart passage through the gate the Scanner reading the products in the cart automatically. And then heading to the cashier to know and pay the total cost.

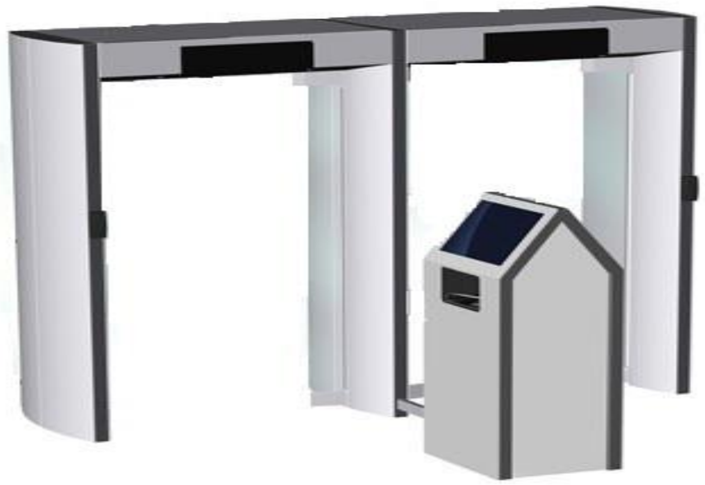


Figure 3.5 Gate which has the reader

3.7.2 Moveable scan point:-

In this type the client roam the supermarket and collects what he needs from the goods in the cart(which has a reader),in this type reader scan each item is put in the cart .Once the customer end his shopping ,all the goods have been registered And then heading to the cashier to know and pay the total cost.

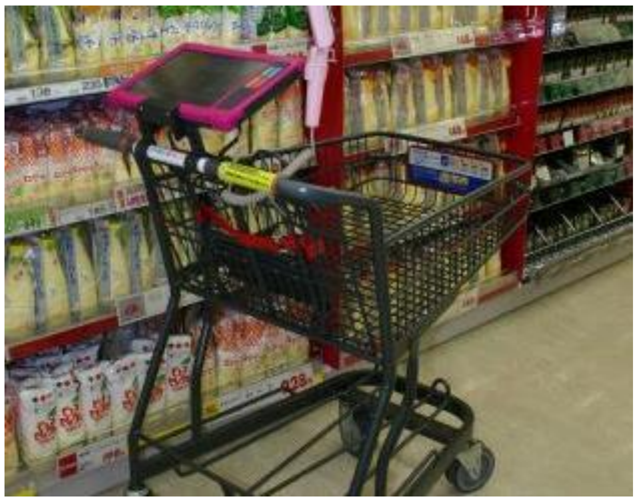


Figure 3.6 Cart which has a reader.

3.7.3 Scheme shows the track of each case:-

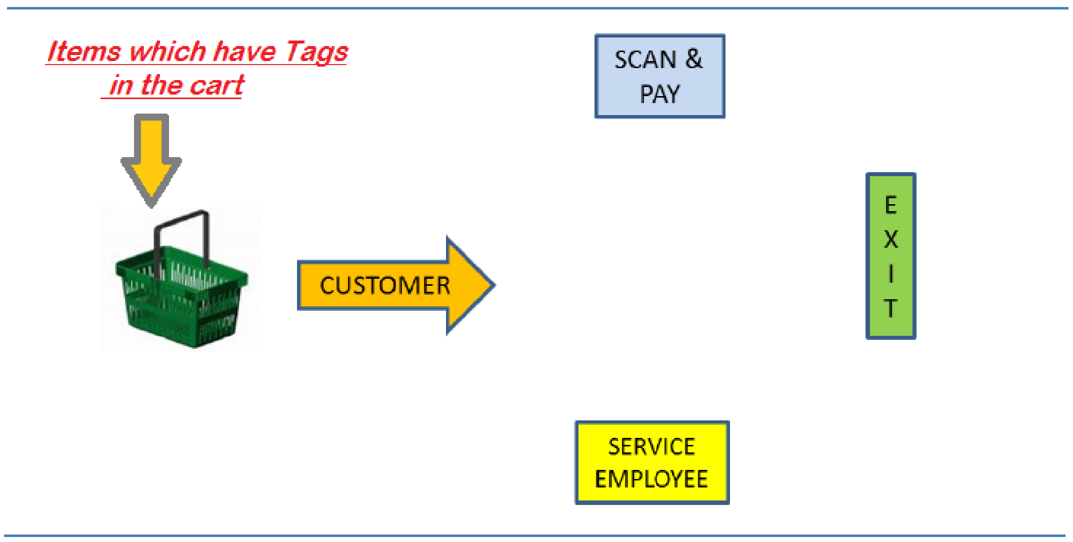


Figure 3.7 The track for 1st type.

`

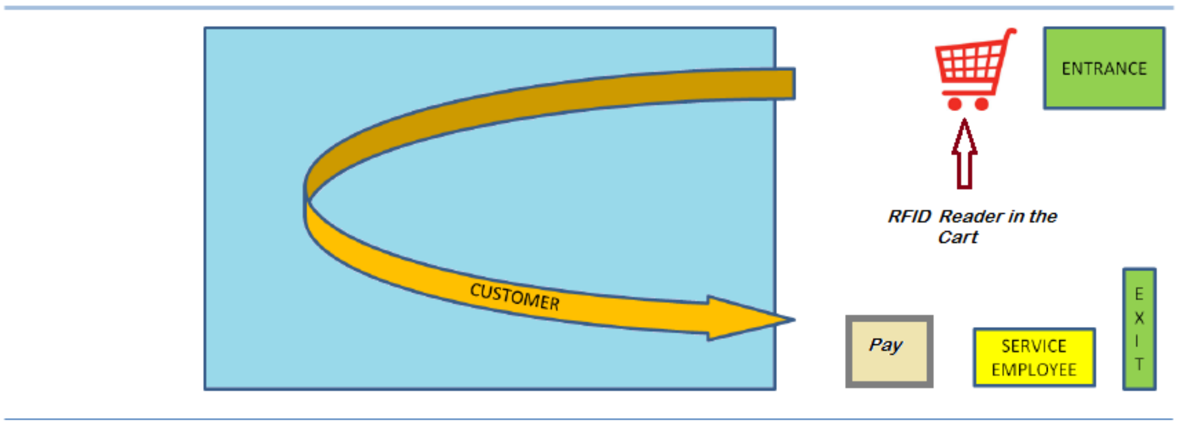


Figure 3.8 The track for 2nd type.

Chapter 4:   System Implantation

4.1 Introduction to Database

Before talking about system implementation we have to know the difference between data and information.

4.1.1 Data versus Information

Information is understood by a person but data are the values stored in a passive medium like a computer disk. The purpose of Database Management System (DBMS) works to bridge the gap between data and information.

4.1.2 Relationship between data and information

Data is The presentation of facts , bits of information but not information itself. When data are processed, interpreted, organized, structured to make them meaningful, they are called information. For example, each student's test score is one piece of data (Data). The average score of a class or of the entire school is information that can be derived from the given data (information).

4.2 What is database?

A database is a collection of organized, inter-related information units. The information unit is a package of information at various levels of granularity. For example a string of letters forming the name of an experimenter or a data value collected from an experiment. It could also be as complex as the protocol of an experiment, the image of a rat brain, the video clip of an experiment in a laboratory. The contents of a database are intended to represent a snapshot of the state of an application environment. The database can be of any size and of varying complexity. For example, the database can contain the information of only a few hundred people working on the same project and could contain the information of an airline company or data collected from scientific experiments. The general purpose of DBMS is collection of integrated mechanisms and tools to support the definition, manipulation, and control of databases for a variety of application.

The basic processes of DBMS:

1. Specification of data types, structures and constraints to be considered in an application.
2. Storing the data itself into persistent storage.
3. Querying the database to retrieve desired data.
4. Updating the content of the database.

4.2 Simple explain of the whole system

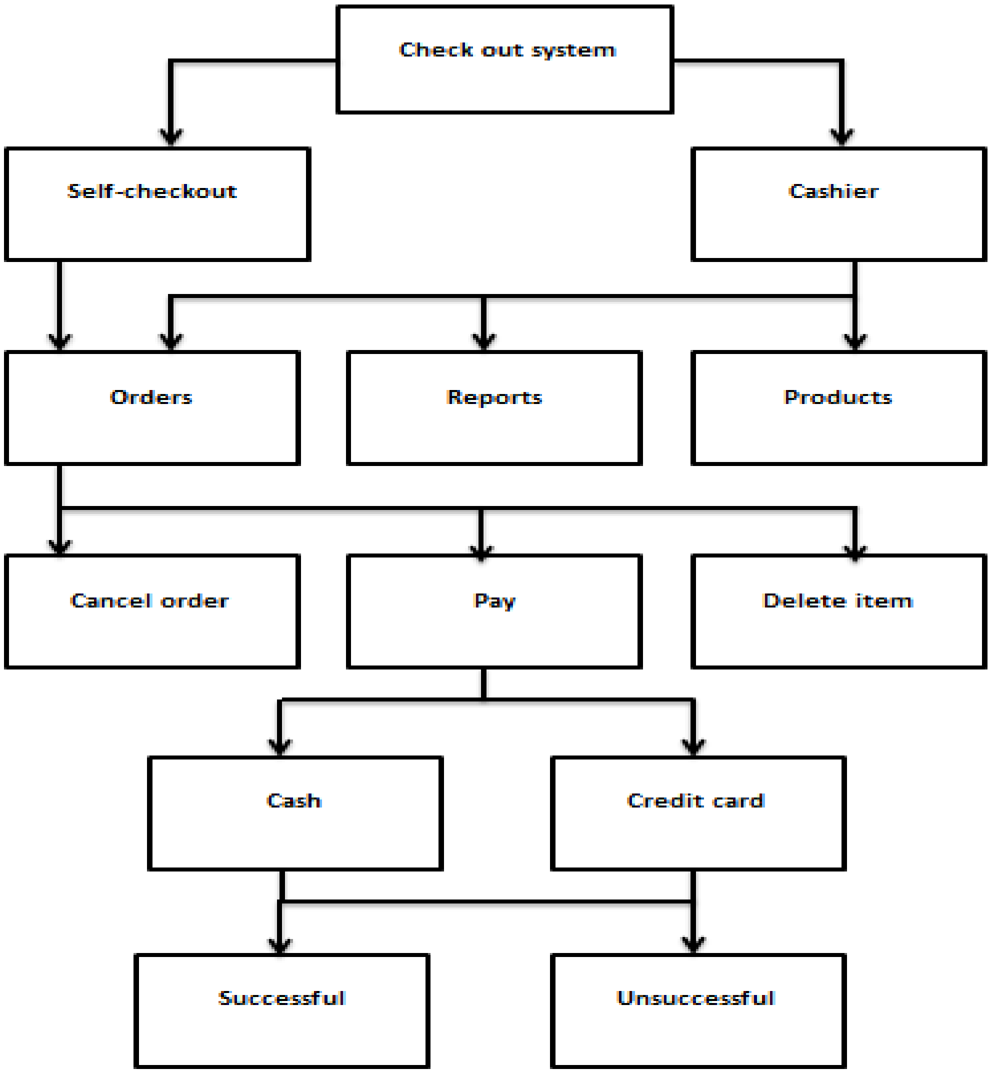


Figure 4.1 system planning

4.3 components used in the system

First: the hardware:

1. 2 HF readers from OG-TECH Company.
2. Tags also from OG-TECH Company.

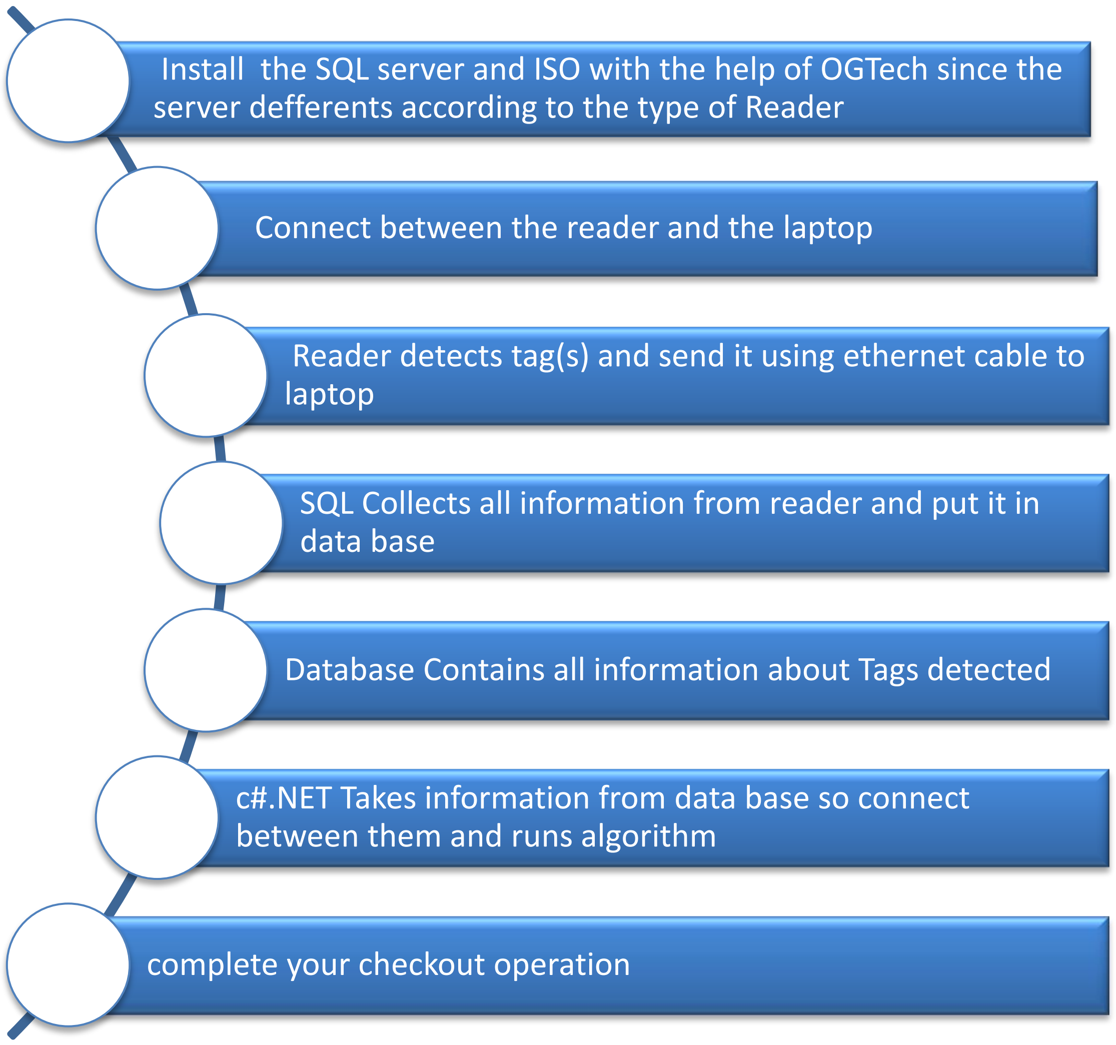
Figure4.2: Reader and tag

3. Switch and Ethernet cables.
4. Ethernet cables.

Second: the software:

1. C#.NET
2. SQL Server 2008 r2
3. ISO Start 2014.

4.4 steps of work


Install the SQL server and ISO with the help of OGTech since the server defferents according to the type of Reader
Connect between the reader and the laptop
Reader detects tag(s) and send it using ethernet cable to laptop
SQL Collects all information from reader and put it in data base
Database Contains all information about Tags detected
c#.NET Takes information from data base so connect between them and runs algorithm
complete your checkout operation

4.5 Algorithm of the system

As mentioned before, the database used to store data. So, the system has some tables to store the serial which comes from the tags and using it to connect the tables with each other.

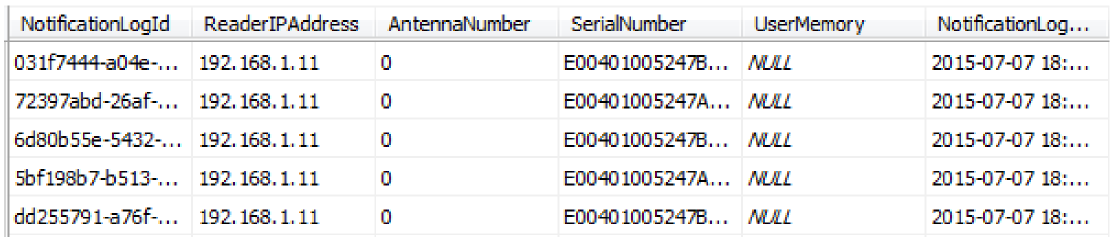
4.5.1 The database of the system has 7 tables in SQL

1- Notification Log
2- Product Serial_ID

3- Product
4- Order
5- My Users
6- History
7- Credit Cards

1 Notification Log

It is a database which stores the serial number of the tags and some information about tags [Figure4.3].

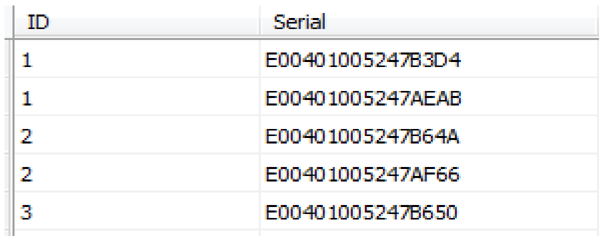


NotificationLogId	ReaderIPAddress	AntennaNumber	SerialNumber	UserMemory	NotificationLog...
031f7444-a04e-...	192.168.1.11	0	E00401005247B...	NULL	2015-07-07 18:...
72397abd-26af-...	192.168.1.11	0	E00401005247A...	NULL	2015-07-07 18:...
6d80b55e-5432-...	192.168.1.11	0	E00401005247B...	NULL	2015-07-07 18:...
5bf198b7-b513-...	192.168.1.11	0	E00401005247A...	NULL	2015-07-07 18:...
dd255791-a76f-...	192.168.1.11	0	E00401005247B...	NULL	2015-07-07 18:...

Figure4.3: db0.NotificationLog

2 Product Serial_ID

Each item of the product has a different serial number. So, the ProductSerial_ID database contains the whole serials of items in the supermarket. When a customer buys an item, the item will be deleted from the database to calculate the number of items which have been sold and alert the cashier that there are no products in the supermarket [Figure4.4].

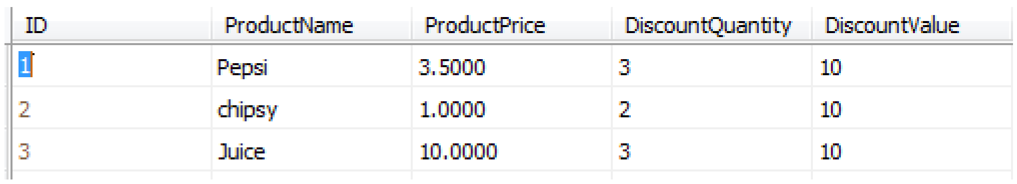


ID	Serial
1	E00401005247B3D4
1	E00401005247AEAB
2	E00401005247B64A
2	E00401005247AF66
3	E00401005247B650

Figure4.4: db0 ProductSerial_ID

3 Product

It is a database which stores the names of products ,prices, discount quantity and the percentage of the discount [Figure4.5].



ID	ProductName	ProductPrice	DiscountQuantity	DiscountValue
1	Pepsi	3.5000	3	10
2	chipsy	1.0000	2	10
3	Juice	10.0000	3	10

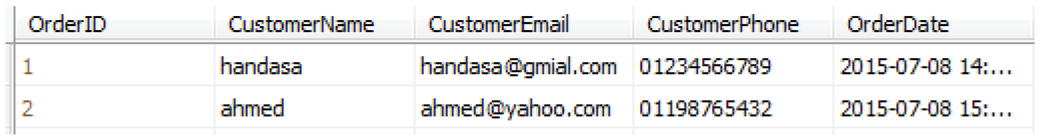
Figure4.5: db0 Product

Note:

Each product has a unique ID and a several serials for its items. So, the ID in the ProductSerial_ID database is repeated.

4 Order

It is a database which stores the data of the customer like customer name, customer email and customer phone. To send the new offers of the products to the customer [Figure4.6].

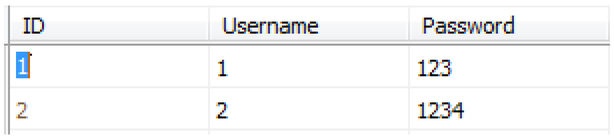


OrderID	CustomerName	CustomerEmail	CustomerPhone	OrderDate
1	handasa	handasa@gmial.com	01234566789	2015-07-08 14:...
2	ahmed	ahmed@yahoo.com	01198765432	2015-07-08 15:...

Figure4.6: db0 Product

5 My users

It is a database which stores the data of the cashier like userID and password to log in [Figure4.7].

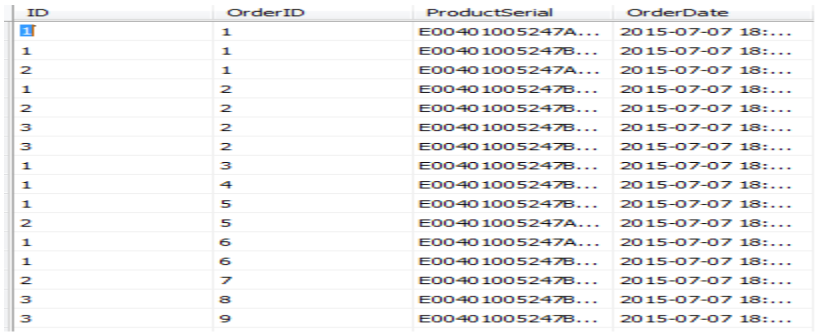


ID	Username	Password
1	1	123
2	2	1234

Figure4.7: db0 myUsers

6 History

Sometimes the customer or the cashier needs a data about old orders. So, the history database stores the data of all orders like ID, orderID, serials of the products and the date [Figure4.8].

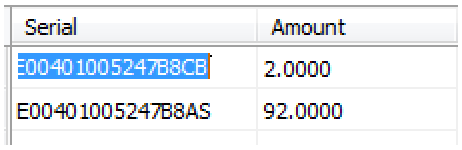


ID	OrderID	ProductSerial	OrderDate
1	1	E00401005247A...	2015-07-07 18:...
1	1	E00401005247B...	2015-07-07 18:...
2	1	E00401005247A...	2015-07-07 18:...
1	2	E00401005247B...	2015-07-07 18:...
2	2	E00401005247B...	2015-07-07 18:...
3	2	E00401005247B...	2015-07-07 18:...
3	2	E00401005247B...	2015-07-07 18:...
1	3	E00401005247B...	2015-07-07 18:...
1	4	E00401005247B...	2015-07-07 18:...
1	5	E00401005247B...	2015-07-07 18:...
2	5	E00401005247A...	2015-07-07 18:...
1	6	E00401005247A...	2015-07-07 18:...
1	6	E00401005247B...	2015-07-07 18:...
2	7	E00401005247B...	2015-07-07 18:...
3	8	E00401005247B...	2015-07-07 18:...
3	9	E00401005247B...	2015-07-07 18:...

Figure4.8: db0 myUsers

7 CreditCards

Since there in no devices to read the credit cards, this database used to store and identify some serials which represent the credit cards [Figure4.9].



Serial	Amount
E00401005247B8CB	2.0000
E00401005247B8AS	92.0000

Figure4.9: db0 CreditCards

4.5.2 Explanation of the code

1 Program.cs

Is the program source which determines which form the system will start with.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Sales_System
{
   static class Program
   {
```

`To open libararies

Name of the system

```
        public static bool _User = true;
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {

            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmWelcome());
        }
    }
}
```

To determine whether the user was a cashier or customer if

To run form Welcome

//

2 Welcome form

Form to choose the user is customer or cashier.

```
namespace Sales_System
{
    public partial class frmWelcome : Form
    {
        public frmWelcome()
        {
            InitializeComponent();
        }

        private void btnCustomer_Click(object sender, EventArgs e)
        {
            frmOrder order = new frmOrder();
            order.Show();
            Program._User = true;
        }

        private void btnCachier_Click(object sender, EventArgs e)
        {
            frmLogin login = new frmLogin();
            login.Show();
            Program._User = false;

        }
    }
}
```

Class form Welcome

Button of Customer

To book place to form Order and then show it when user is customer

Button of Cashier

To book place to form login and then show it when user is cashier

Result

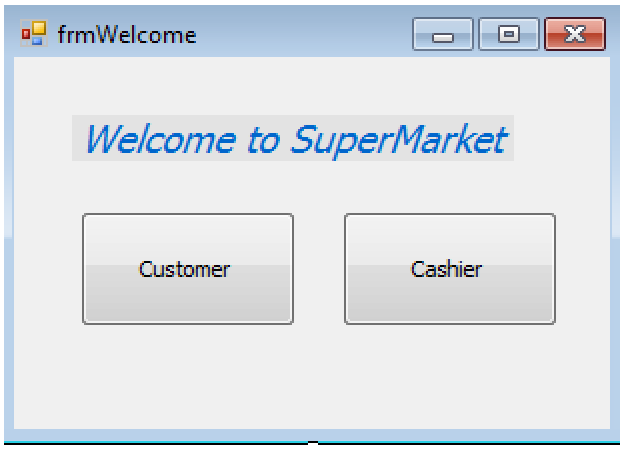


Figure4.10 welcome form

///

3 log in form

form for the cashier to log in using userID and password.

```
namespace Sales_System
{
   public partial class frmLogin : Form
   {
      public frmLogin()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         try
         {
            string ID = txtUserID.Text.Trim();
            string Password = txtPassword.Text.Trim();
            //Check database for userID and password

            string cmd = string.Format("select ISNULL(( select 1 as 'Result' from dbo.myUsers
where ID = {0} and [Password] = '{1}'),0)",ID,Password);
```

Class form login

Enter ID& password

Trim: to remove spaces

```
Check database for user ID and password //string cmd = "select
[Username] as 'Name' from dbo.myUsers where ID = " + ID + " and
[Password] = " + Password;
```

```
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            int result = (int)DAL.selectValue(cmd);
```

To book place in data base, communicate with DAL and get value in the command

```
            if (result == 1)
            {
                Main_Window frm = new Main_Window();
                frm.Show();
                this.Hide();
                Program._User = false;
            }
```

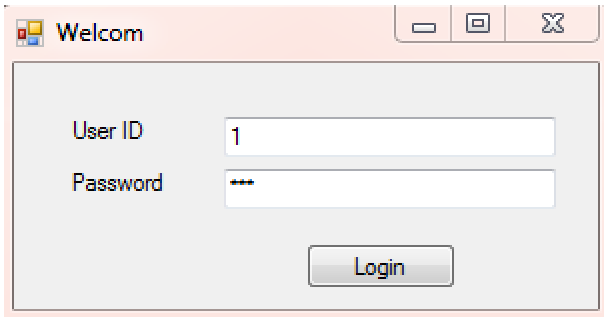
If the condition is true, it enter in the main window when the user is cashier

```
            else
            {
                Program._User = true;
                MessageBox.Show("Check your username and
password","Error",MessageBoxButtons.OK);

            }
        }
        catch (Exception)
        {

            throw;
        }
    }
```

If the condition is false, the user will be customer and the massage will be shown to tell the customer to check

```
    private void label1_Click(object sender, EventArgs e)
    {

    }

    private void txtUserID_TextChanged(object sender, EventArgs e)
    {

    }
  }
}
```

Result


Welcom

User ID 1

Password ***

Login

Figure4.11 log in form

//

4 Main form

Form for the cashier to choose if he wants to make order or add products or see the reports.

Class of main window

```
namespace Sales_System
{
   public partial class Main_Window : Form
   {
      public Main_Window()
      {
         InitializeComponent();
      }
```

Button of orders

To book form orders and show form

```
      private void btnOrders_Click(object sender, EventArgs e)
      {
         frmOrder Orders = new frmOrder();
         Orders.Show();
      }
```

Button of products

To book form product and show

```
      private void btnProducts_Click(object sender, EventArgs e)
      {
         frmAddProduct frm = new frmAddProduct();
         frm.Show();
      }
```

Button of reports

To book form reports and show

```
      private void btnReports_Click(object sender, EventArgs e)
      {
         frmReports frm = new frmReports();
         frm.Show();
      }
   }
}
```

Result

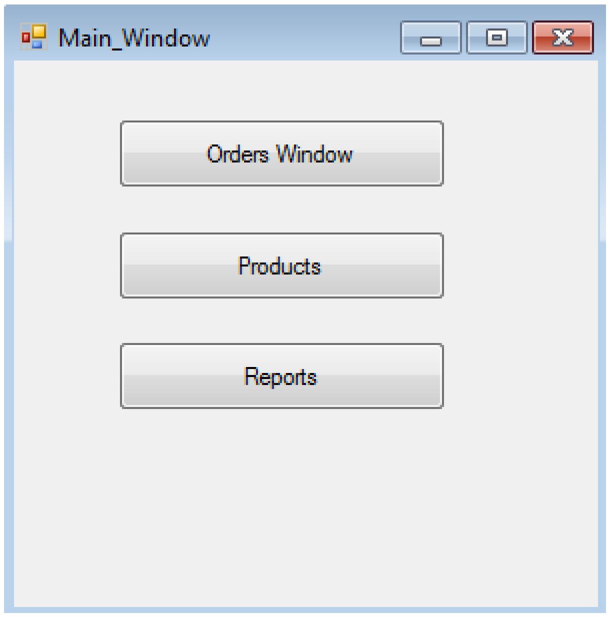


Figure4.12 main_window form

//

5 Order form

Order form shows the items which the customer

```
namespace Sales_System
{
  public partial class frmOrder : Form          Class of form order
  {
    clsDataAccessLayer DAL;

    decimal total = 0;                To calculate total price
    public decimal Total
    {
      get { return total; }
      set { total = value; }
    }

    bool PaymentStatus = false;


    private void GeneralSetting()
    {
      dataGridView1.AutoGenerateColumns = false;
      dataGridView1.MultiSelect = false;
      dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
```

```
            dataGridView1.RowsDefaultCellStyle.BackColor = Color.WhiteSmoke;
            dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.White;

        }
        public frmOrder()
        {
            InitializeComponent();
            DAL = new clsDataAccessLayer();
            GeneralSetting();
        }

        //Cancel the order
        private void btnCancelOrder_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to cancel the order?", "warning", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result == System.Windows.Forms.DialogResult.Yes)
            {
```

To cancel the order

```
                string cmd = "Delete from NotificationLog";
                DAL.ExecuteNonQuery(cmd);
```

To delete items from DB

```
                dataGridView1.DataSource = null;
                lblOrderID.Text = "";
                lblTotal.Text = "";
                txtCustomerName.Text = "";
                MessageBox.Show("Operation Complete", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

        }
```

To clear form

To start new order

```
        private void btnNewCustomer_Click(object sender, EventArgs e)
        {

            string cmd = "select max(OrderID)+ 1 from [Order]";
            lblOrderID.Text =  DAL.selectValue(cmd).ToString();
            //2- Get Scanned item from DB
            cmd = "SELECT   ProductSerial_ID.ID, Product.ProductName, Product.ProductPrice, Product.DiscountQuantity, Product.DiscountValue, count(NotificationLog.SerialNumber) AS 'Quantity' FROM Product INNER JOIN ProductSerial_ID ON Product.ID = ProductSerial_ID.ID INNER JOIN NotificationLog ON ProductSerial_ID.Serial = NotificationLog.SerialNumber group by   ProductSerial_ID.ID, Product.ProductName, Product.ProductPrice, Product.DiscountQuantity, Product.DiscountValue";
```

To generate Order ID

```
        DataTable dt = DAL.SelectTable(cmd);
        if (dt.Rows.Count > 0)
        {
            dt.Columns.Add("Total", typeof(decimal));

            dataGridView1.DataSource = dt;

            for (int i = 0; i < dt.Rows.Count; i++)
            {
                int Quantity = Convert.ToInt32(dt.Rows[i][5].ToString());
                int DiscountQuantity = Convert.ToInt32(dt.Rows[i][3].ToString());
                int Discountvalue = Convert.ToInt32(dt.Rows[i][4].ToString());
                decimal ItemPrice = Convert.ToDecimal(dt.Rows[i][2].ToString());

                decimal ProductTotal = ItemPrice * Quantity;



                if (Quantity >= DiscountQuantity)
                {
                    dt.Rows[i][6] = ProductTotal - (ProductTotal * Discountvalue) / 100;
                }
                else
                {
                    dt.Rows[i][6] = ProductTotal;

                }

                total += Convert.ToDecimal(dt.Rows[i][6].ToString());


            }

            lblTotal.Text = total.ToString();

        }

    }


    private void btnSaveOrder_Click(object sender, EventArgs e)
    {
        if (dataGridView1.Rows.Count < 1)
        {
            MessageBox.Show("No items to buy", "Info", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            return;
```

Get Scanned item from DB

Fill gridview with data

To calculate total price

To calculate total price after discount if the Condition satisfy

To show total price

To save order to DB

```
            }
            frmPayMethod _frmPayMethod = new frmPayMethod();
            _frmPayMethod.TotalCost = this.total;
            _frmPayMethod.ShowDialog(this);
            this.PaymentStatus = _frmPayMethod.PaymentStatus;

            if (PaymentStatus == true)
            {
```

To save customer data (Create new Order ID) by communication with DAL, clear scanned item from DB

```
                string cmd = string.Format("EXECUTE [RFIDNotificationService].[dbo].[SaveOrder]
@CustomerName = '{0}', @CustomerPhone = '{1}', @CustomerEmail= '{2}'",
txtCustomerName.Text.Trim(),txtCustomerPhone.Text,txtCustomerEmail.Text);
                DAL.ExecuteNonQuery(cmd);


                //DAL.ExecuteNonQuery(cmd);
                // 2- clear scanned item from db
                //cmd = "Delete from NotificationLog";
                //DAL.ExecuteNonQuery(cmd);


                frmPrintbill print = new frmPrintbill();

                print.TotalCost = this.Total;
                print.OrderID = Convert.ToInt32( lblOrderID.Text);
                print.dt = (DataTable)dataGridView1.DataSource;
                print.ShowDialog(this);


                // 3- clear form
                dataGridView1.DataSource = null;
                lblOrderID.Text = "";
                lblTotal.Text = "";
                txtCustomerName.Text = "";
                txtCustomerPhone.Text = "";
                txtCustomerEmail.Text = "";
                Total = 0;

                MessageBox.Show("Operation Complete", "Info", MessageBoxButtons.OK,
MessageBoxIcon.Information);

                if (Program._User == false)
                {
```

If the condition true, it will be the

```
                    cmd = "SELECT ProductSerial_ID.ID, Product.ProductName
,COUNT(ProductSerial_ID.Serial)as 'Quantity' FROM ProductSerial_ID INNER JOIN Product
```

```
ON ProductSerial_ID.ID = Product.ID GROUP BY ProductSerial_ID.ID, Product.ProductName
HAVING (COUNT(ProductSerial_ID.Serial) < 6)";
                    //clsDataAccessLayer DAL = new clsDataAccessLayer();
                    DataTable dt = DAL.SelectTable(cmd);
                    if (dt.Rows.Count > 0)
                    {
                        Alert alert = new Alert();
                        alert.ShowDialog();
                    }

                }
            }
            else
            {
                MessageBox.Show("Payment operation did not complete.", "Info",
MessageBoxButtons.OK, MessageBoxIcon.Information);

            }
        }

        private void frmOrder_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
            frmLogin welcome = new frmLogin();
            welcome.Show();
        }

        //delete item
        private void button1_Click_1(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count > 0)
            {

                if (MessageBox.Show("Please scan the item you want to delete.", "Delete Item",
MessageBoxButtons.OKCancel, MessageBoxIcon.Information) ==
System.Windows.Forms.DialogResult.OK)
                {
                    string cmd = string.Format("delete from NotificationLog where SerialNumber =
(SELECT top 1 SerialNumber FROM NotificationLog  group by SerialNumber having
(count(NotificationLogId) > 1))");
                    DAL.ExecuteNonQuery(cmd);
                }
```

Make communication with DAL and get table of command

If condition true, alert massage will be shown

If condition false,the massage will tell you that Payment operation did not complet

To delete item if the condition true, the massage will tell you to scan the item you want

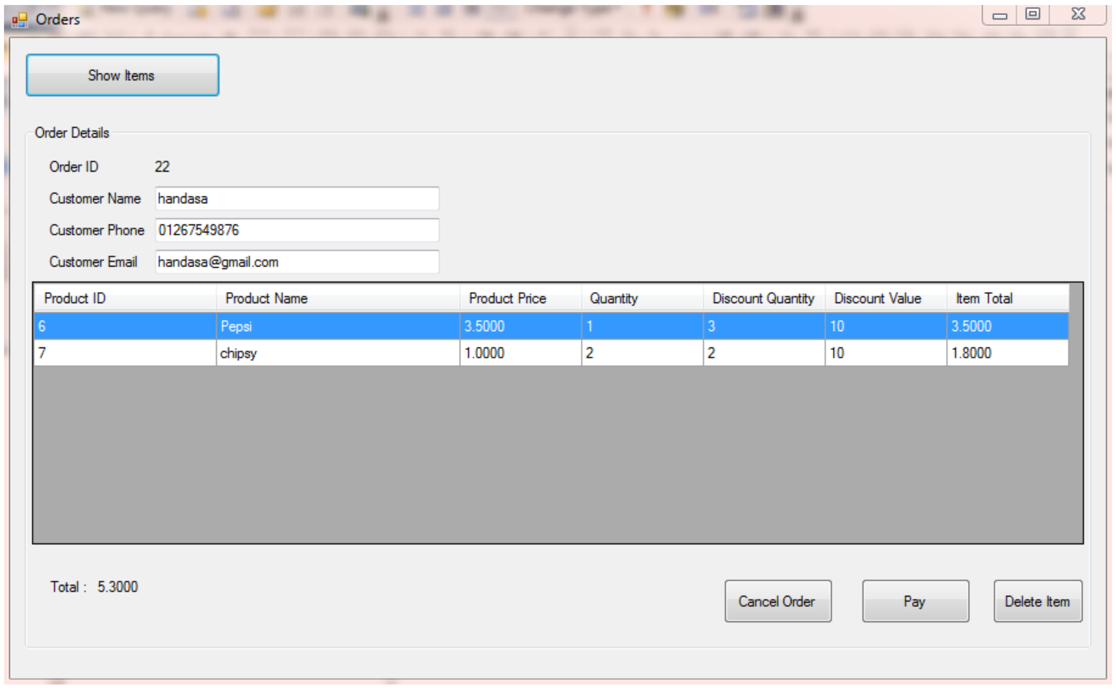
To delete item you should SELECT top 1 Serial Number FROM Notification Log group by Serial Number having (count (Notification Log Id) > 1) and communicate with DAL then execute the command

```
            this.Total = 0;
            btnNewCustomer_Click(null, null);
         }
         else
            MessageBox.  Show("No  items  to  delete",  "Error",  MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

If the condition false, the massage will be shown to tell you `No items to delete`

```
         /*
         if (dataGridView1.Rows.Count > 0)
         {
            int index = dataGridView1.CurrentRow.Index;
            int                                              Quantity                                       =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[3].Value.ToString());
            if (Quantity > 1)
            {
               dataGridView1.CurrentRow.Cells[3].Value = Quantity - 1;

            }
            else if (Quantity == 1)
            {

            }
         }
         */
        //lblTotal.Text                 =                (Convert.ToDecimal(lblTotal.Text)              -
Convert.ToDecimal(dataGridView1.CurrentRow.Cells[2].Value.ToString())).ToString();
         //Get item price string from grid
         //get total from lable string

         // convert 2 price to decimal

         // substraction

         // update lable with total price



      }

      private void textBox1_TextChanged(object sender, EventArgs e)
      {

      }
   }
}
```

Result


Orders

Show Items

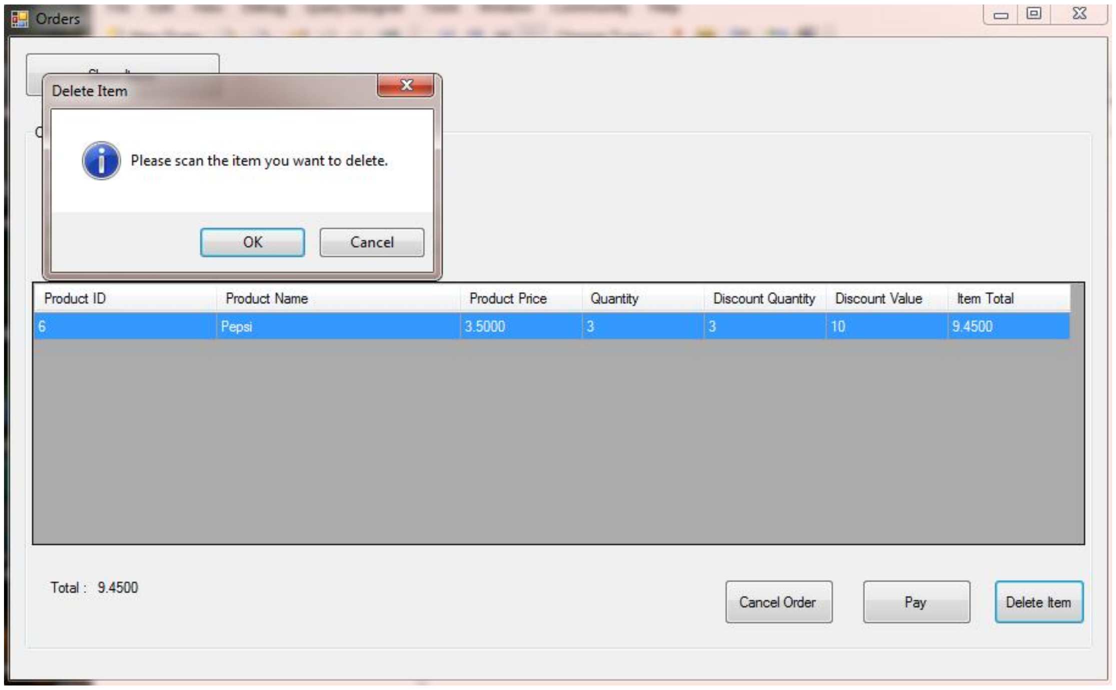
Order Details

Order ID	22
Customer Name	handasa
Customer Phone	01267549876
Customer Email	handasa@gmail.com

Product ID	Product Name	Product Price	Quantity	Discount Quantity	Discount Value	Item Total
6	Pepsi	3.5000	1	3	10	3.5000
7	chipsy	1.0000	2	2	10	1.8000

Total : 5.3000

Figure4.13: order form and show items


Orders

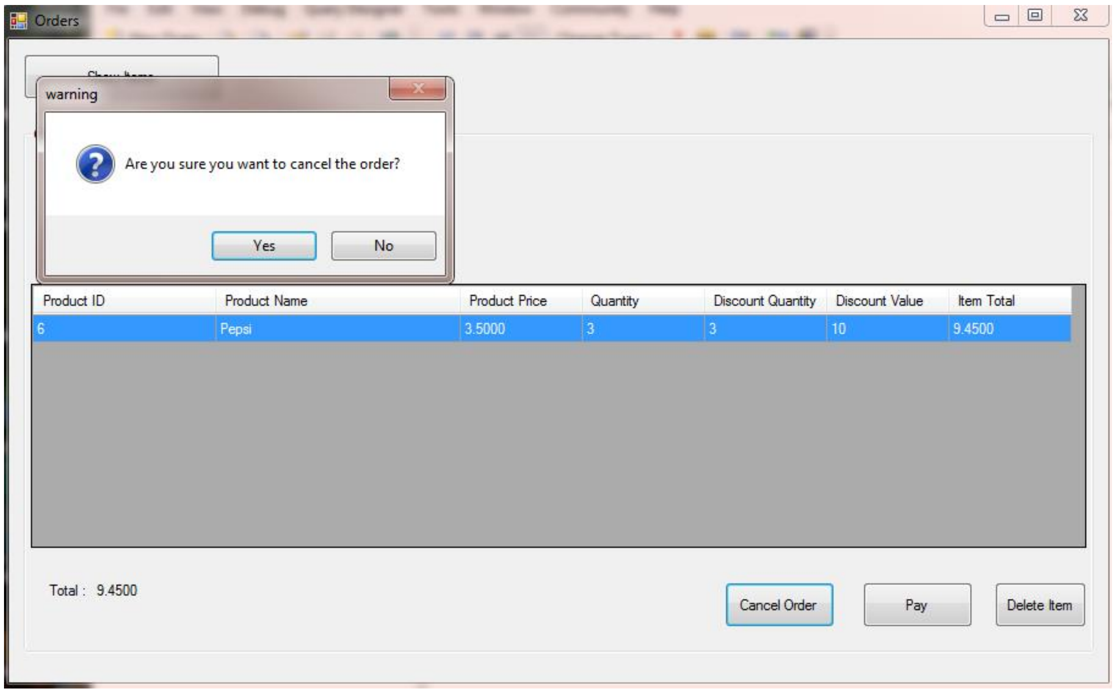
Delete Item

Please scan the item you want to delete.

Product ID	Product Name	Product Price	Quantity	Discount Quantity	Discount Value	Item Total
6	Pepsi	3.5000	3	3	10	9.4500

Total : 9.4500

Figure4.14: delete item message



Are you sure you want to cancel the order?

Product ID	Product Name	Product Price	Quantity	Discount Quantity	Discount Value	Item Total
6	Pepsi	3.5000	3	3	10	9.4500

Total : 9.4500

Figure4.15: cancel order message

//

6 Add product form

Add product form to add a new product and new serials In the database of the supermarket.

```
public partial class frmAddProduct : Form
   {
       private void LoadProductData()
       {
          string cmd = "SELECT [ID] ,str([ID])+' | '+[ProductName]  as 'ProductName' FROM
[Product]";
          clsDataAccessLayer DAL = new clsDataAccessLayer();
          DataTable dt = new DataTable();
          dt = DAL.SelectTable(cmd);
          cmbProduct.DisplayMember = "ProductName";
          cmbProduct.ValueMember = "ID";
          cmbProduct.DataSource = dt;

       }
       public frmAddProduct()
       {
          InitializeComponent();
          LoadProductData();
       }
//Scan tag serial
```

Class of form add

Book place in cls Data Access Layer ,book place for Data Table, make communication with DAL and get table in command, put Product Name, ID

```
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                string cmd = "select isnull((SELECT TOP 1 [SerialNumber] FROM [NotificationLog] 
where [SerialNumber] not in (select [Serial] from [ProductSerial_ID]) order by 
[NotificationLogDate] desc),'')";
                clsDataAccessLayer DAL = new clsDataAccessLayer();
                string result = DAL.selectValue(cmd).ToString();
                if (string.IsNullOrEmpty(result))
                {
                    MessageBox.Show("No new tag to add, Please scan a valid tag", "Error", 
MessageBoxButtons.OK,MessageBoxIcon.Error);
                    return;
                }
                else
                    txtSerial.Text = result;
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);

            }


        }

        private void button2_Click(object sender, EventArgs e)
        {

            string name = txtProductName.Text;
            if (string.IsNullOrEmpty(name))
            {

                MessageBox.Show("Enter product name");
                return;
            }

            string Price = txtProductPrice.Text;
            if (string.IsNullOrEmpty(Price))
            {
                MessageBox.Show("Enter product price");
                return;
            }
            double d_price;

            if (!double.TryParse(Price,out d_price))
            {
                MessageBox.Show("Enter product correct price");
                return;
            }

            string DiscountQuantity = txtDiscountQuantity.Text;
            if (string.IsNullOrEmpty(DiscountQuantity))
            {
                MessageBox.Show("Please enter the Discount Quantity");
                return;
            }

            string DiscountValue = txtDiscountValue.Text;
            if (string.IsNullOrEmpty(DiscountValue))
            {
MessageBox.Show("Please enter the DiscountValue");
                return;
            }


            string cmd = string.Format("INSERT INTO [Product]([ProductName],[ProductPrice]
,[DiscountQuantity],[DiscountValue])VALUES
('{0}',{1},{2},{3})",name,d_price,DiscountQuantity,DiscountValue);
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DAL.ExecuteNonQuery(cmd);

            LoadProductData();


            txtDiscountQuantity.Text = "";
            txtProductName.Text = "";
            txtProductPrice.Text = "";
            txtDiscountValue.Text  = "";

            MessageBox.Show("Save Done");
        }

        private void btnSaveTag_Click(object sender, EventArgs e)
        {
            try
            {
                string Serial = txtSerial.Text;
                if (string.IsNullOrEmpty(Serial))
                {
MessageBox.Show("Please scan the tag");
                    return;
                }
                int id = Convert.ToInt32(cmbProduct.SelectedValue);
                string cmd = string.Format("INSERT INTO [ProductSerial_ID]([ID]
,[Serial],[ExpireDate]) VALUES ({0},'{1}','{2}')",id,Serial,dateTimePicker1.Value);
                clsDataAccessLayer DAL = new clsDataAccessLayer();
                DAL.ExecuteNonQuery(cmd);
                cmd = "Delete from NotificationLog";
                DAL.ExecuteNonQuery(cmd);
                txtSerial.Text = "";
                MessageBox.Show("Save Done");


            }
            catch (Exception exp)

                MessageBox.Show(exp.Message);
            }
        }

        private void frmAddProduct_Load(object sender, EventArgs e)
        {

        }
```

Result

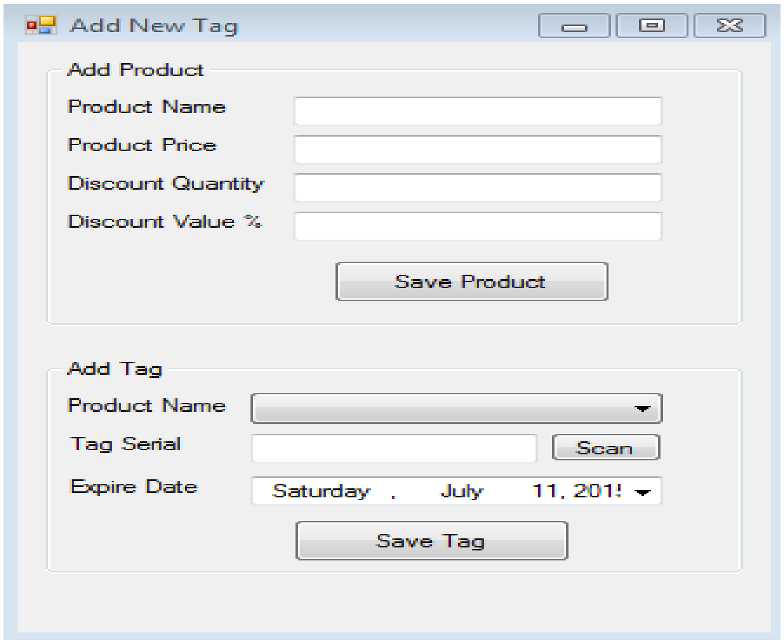
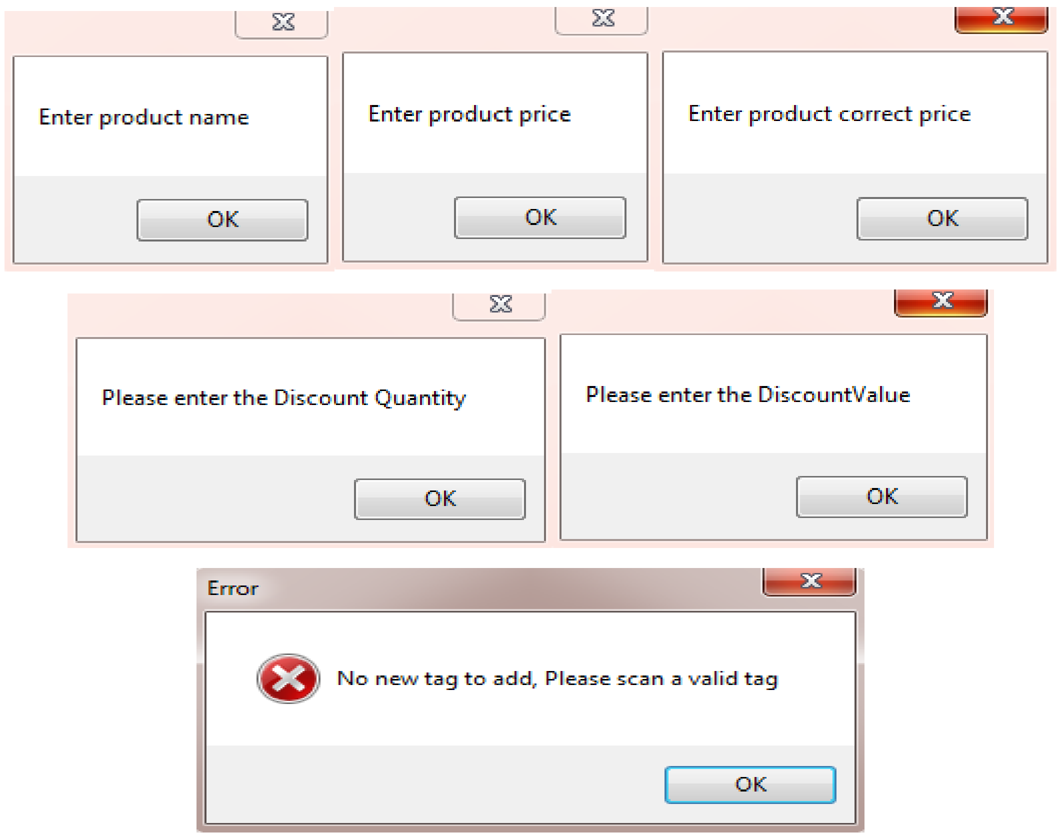


Figure4.16: add product form

In case of errors


Enter product name

Enter product price

Enter product correct price

Please enter the Discount Quantity

Please enter the DiscountValue

Error

No new tag to add, Please scan a valid tag

Figure4.17: error messages

In case of correct data

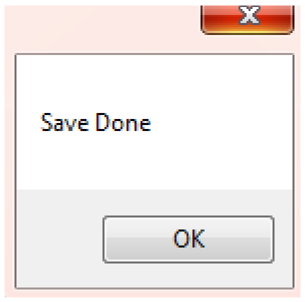

Save Done

Figure4.18: save messages

//

7 Report form

Report form for the cashier. It contains the top 10 of the products, stock, expire date of the products and total price per day.

Class for form

```
public partial class frmReports : Form
   {

       private void GeneralSetting()
       {
          /*
          dataGridView1.AutoGenerateColumns = false;
          dataGridView1.MultiSelect = false;
          dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
          dataGridView1.RowsDefaultCellStyle.BackColor = Color.WhiteSmoke;
          dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.White;
          */
       }
       private void LoadTopProducts()
       {
          string cmd = "SELECT top 10  History.ID, COUNT(History.ProductSerial) AS 
'Quantity', Product.ProductName, Product.ProductPrice FROM History INNER JOIN Product 
ON History.ID = Product.ID GROUP BY History.ID, Product.ProductName, 
Product.ProductPrice order by Quantity desc";
          clsDataAccessLayer DAL = new clsDataAccessLayer();
          DataTable dt = DAL.SelectTable(cmd);
          dataGridView1.DataSource = dt;
       }
```

To get the most ten bought product by make communication

```
       private void LoadStockData()
       {
          string cmd = "SELECT ProductSerial_ID.ID, count(ProductSerial_ID.Serial) as 
'Quantity', Product.ProductName, Product.ProductPrice FROM ProductSerial_ID INNER JOIN 
Product ON ProductSerial_ID.ID = Product.ID group by 
ProductSerial_ID.ID,Product.ProductName, Product.ProductPrice";
          clsDataAccessLayer DAL = new clsDataAccessLayer();
          DataTable dt = DAL.SelectTable(cmd);
          dataGridView2.DataSource = dt;

}
```

To get ProductSerial_ID.ID by making joint with tables and communicate with DAL and put it in dataGridView2.DataSource= dt

```
       void Alert()
       {
          string cmd = "SELECT ProductSerial_ID.ID, Product.ProductName 
,COUNT(ProductSerial_ID.Serial)as 'Quantity' FROM ProductSerial_ID INNER JOIN Product 
ON ProductSerial_ID.ID = Product.ID GROUP BY ProductSerial_ID.ID, Product.ProductName 
HAVING (COUNT(ProductSerial_ID.Serial) < 6)";
          clsDataAccessLayer DAL = new clsDataAccessLayer();
```

```
        DataTable dt = DAL.SelectTable(cmd);
        dataGridView3.DataSource = dt;
    }

    void LoadExpireProducts()

    {
        string cmd = "SELECT ProductSerial_ID.Serial, Product.ID, Product.ProductName, 
ProductSerial_ID.ExpireDate FROM ProductSerial_ID INNER JOIN Product ON 
ProductSerial_ID.ID = Product.ID WHERE     (ProductSerial_ID.ExpireDate < GETDATE())";
        clsDataAccessLayer DAL = new clsDataAccessLayer();
        DataTable dt = DAL.SelectTable(cmd);
        dataGridView4.DataSource = dt;

    }

    void LoadTotalPerDay()
    {
        string cmd = "SELECT  RIGHT( str(day(History.OrderDate)),2)+'_'+
str(datepart(month,History.OrderDate)) +'_'+ str(datepart(year,History.OrderDate)) ,
SUM(Product.ProductPrice) FROM History INNER JOIN Product ON History.ID = Product.ID
where day(History.OrderDate) = DAY(GETDATE()) group by RIGHT(
str(day(History.OrderDate)),2)+'_'+ str(datepart(month,History.OrderDate)) +'_'+
str(datepart(year,History.OrderDate))";
        clsDataAccessLayer DAL = new clsDataAccessLayer();
        DataTable dt = DAL.SelectTable(cmd);
        dataGridView5.DataSource = dt;

    }

    public frmReports()
 {
        GeneralSetting();

        InitializeComponent();


        LoadStockData();

        LoadTopProducts();

        Alert();

        LoadExpireProducts();

        LoadTotalPerDay();
    }
```

If the certain product reduces for certain limit, there will be alert

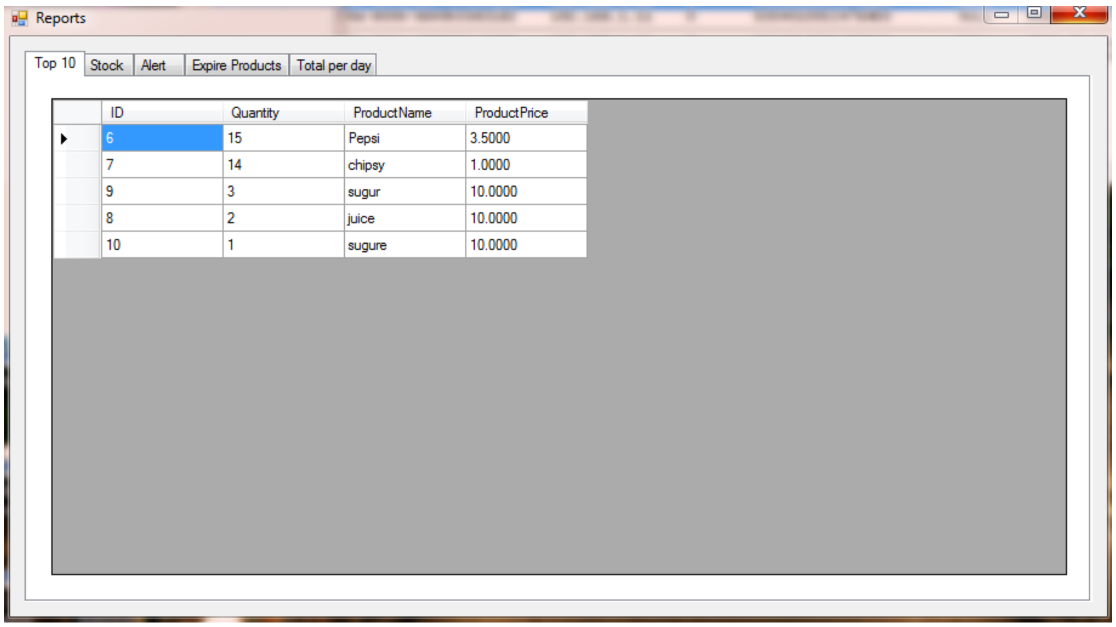
To get total price of product per day

To show stock data, top products, alert, expire products and total price of the products per day in the form reports

```
        private void button1_Click(object sender, EventArgs e)
        {
            DateTime t = dateTimePicker1.Value;

            string cmd = string.Format("SELECT  RIGHT(

    str(day(History.OrderDate)),2)+'_'+ str(datepart(month,History.OrderDate)) +'_'+

    str(datepart(year,History.OrderDate)) , SUM(Product.ProductPrice) FROM History INNER
    JOIN

    Product ON History.ID = Product.ID where day(History.OrderDate) = {0} group by RIGHT(

    str(day(History.OrderDate)),2)+'_'+ str(datepart(month,History.OrderDate)) +'_'+

    str(datepart(year,History.OrderDate))",t.Day);

    clsDataAccessLayer DAL = new clsDataAccessLayer()
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView5.DataSource = dt;
        }}}
```

To determine expire products

Result



ID	Quantity	ProductName	ProductPrice
6	15	Pepsi	3.5000
7	14	chipsy	1.0000
9	3	sugur	10.0000
8	2	juice	10.0000
10	1	sugure	10.0000

Figure4.19: report form

//

8- Pay Method

```
public partial class frmPayMethod : Form
   {
       private decimal totalCost;

       public decimal TotalCost
       {
          get { return totalCost; }
          set { totalCost = value; }
       }

       public bool PaymentStatus;


       public frmPayMethod()
       {
          InitializeComponent();
       }

       private void btnCache_Click(object sender, EventArgs e)
       {
          frmPayCache _frmPayCache = new frmPayCache();
          _frmPayCache.TotalCost = this.totalCost;


_frmPayCache.ShowDialog(this);

 this.PaymentStatus = _frmPayCache.PaymentStatus;

 this.Close();

 this.Dispose();
       }

       private void btnCard_Click(object sender, EventArgs e)
       {

    frmPayWithCard _frmPayWithCard = new frmPayWithCard();
          _frmPayWithCard.TotalCost = this.totalCost;
          _frmPayWithCard.ShowDialog(this);
          this.PaymentStatus = _frmPayWithCard.PaymentStatus;
          this.Close();
```

Class of form payment method

To get total cost

Button of cash

To show total cost then pay with cash

To show form pay cash

Button of card

To book place for form pay with card

To show total cost and pay with card then close

```
    this.Dispose();

}}}
```

Result

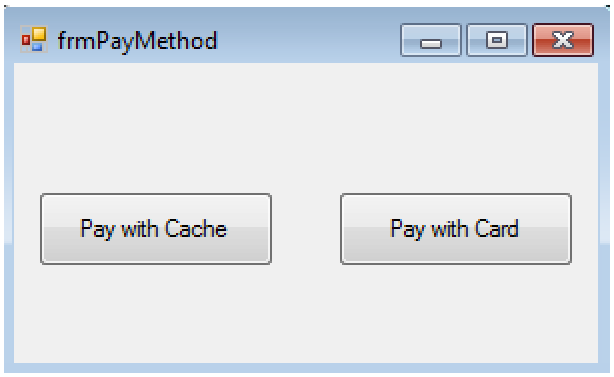


Figure4.20frmPayMethod

//

```
public partial class frmPayCache : Form
    {
        private decimal totalCost;

        public decimal TotalCost
        {
            get { return totalCost; }
            set
            {
                totalCost = value;
                lblTotal.Text = totalCost.ToString("c").TrimStart('$')
            }
        }
        public bool PaymentStatus;

        public frmPayCache()
        {
            InitializeComponent();
        }

        private void btnPay_Click(object sender, EventArgs e)
        {
            this.PaymentStatus = true;
```

Class of from pay

Trim: to delete'$' which appear beside total

Button of pay

```
        this.Close();
        this.Dispose();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        this.PaymentStatus = false;
        this.Close();
        this.Dispose();
    }

    private void frmPayCache_Load(object sender, EventArgs e)
    {

    }}}
```

Button of cancel

Result

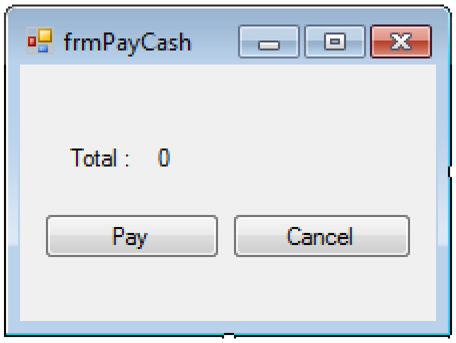
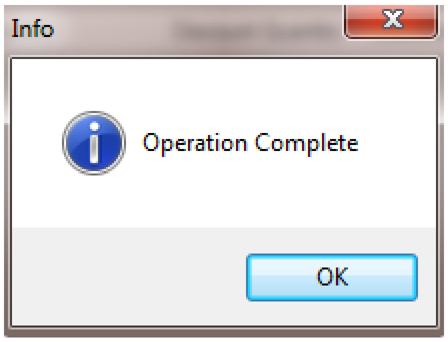


Fig4.21frmPaycash

- Form pay Cash consist of two option (pay, cancel)

1. When pay



Operation Complete

Fig4.22Info

2. when press on cancel return to form order

//

9-Form pay with card

```
public partial class frmPayWithCard : Form
    {

        private decimal totalCost;

        public decimal TotalCost
        {
            get { return totalCost; }
            set
            {
                totalCost = value;
                lblTotal.Text = totalCost.ToString("c").TrimStart('$');
            }
        }

        public bool PaymentStatus;

        string CardID = "";


        public frmPayWithCard()
        {
            InitializeComponent();
        }

        private void btnScan_Click(object sender, EventArgs e)
        {

        private void btnScan_Click(object sender, EventArgs e)
        {
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            try
            {

                string cmd = "SELECT top 1 [SerialNumber]  FROM NotificationLog order by
NotificationLogDate desc";
                CardID = DAL.selectValue(cmd).ToString();
                txtCardID.Text = CardID;


                cmd = string.Format("SELECT [Amount] FROM [CreditCards] where [Serial] =
'{0}'", CardID);
                txtCredit.Text = DAL.selectValue(cmd).ToString();
            }
            catch(Exception exp)
            {
```

Class of form pay with card

Button of scan

Get tag Serial from DB

Get Tag credit from DB

```
            MessageBox.Show(exp.Message);
        }
    }

    private void btnPay_Click(object sender, EventArgs e)
    {

if (Convert.ToDecimal(txtCredit.Text) >= TotalCost)
        {


            string cmd = string.Format("update [CreditCards] set [Amount] = [Amount] -
{0}",totalCost);
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DAL.ExecuteNonQuery(cmd);
            this.PaymentStatus = true;
            this.Close();
            this.Dispose();
        }
 else
        {
            MessageBox.Show("No enough credit");
            this.PaymentStatus = false;
        }
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        this.PaymentStatus = false;
        this.Close();
        this.Dispose();
    }}}
```

Button of pay

To make check if (txtCredit.Text) >= Total Cost

If condition true, we will update credit cards with communication with data base

If condition false, the massage will be shown that No enough credit

Press on button cancel if payment status= false

Result

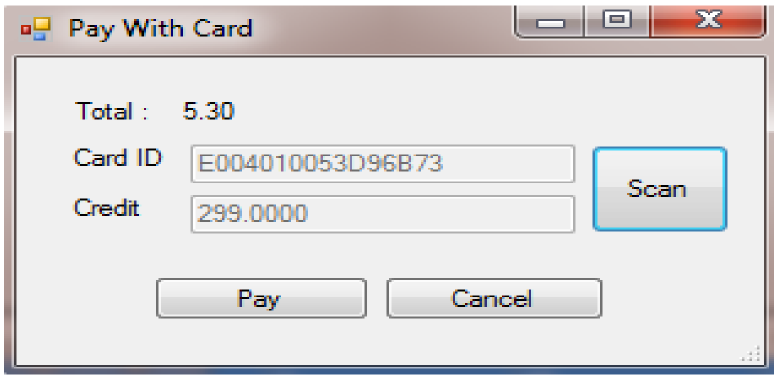


Fig4.23Pay With card

1. When press on Pay button, credit enough.

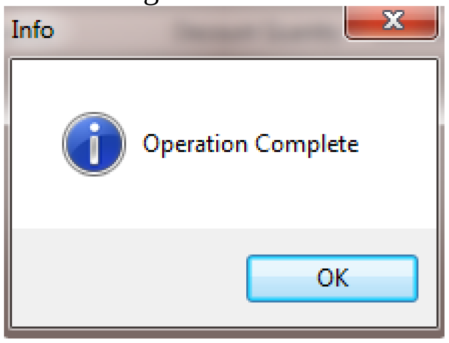


Fig4.24 Info

2. When press on Pay button , credit not enough

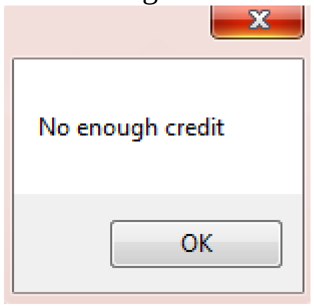


Fig4.25 Error

//

10. Form print bill

```
public partial class frmPrintbill : Form
  {
      private decimal totalCost;

      public decimal TotalCost
      {
          get { return totalCost; }
          set {

              totalCost = value; }
      }

      private int orderID;

      public int OrderID
      {
          get { return orderID; }
          set { orderID = value; }
      }

      public DataTable dt = new DataTable();
```

Class of form print bill

Put value in total

Put value in order ID

Book place for Data

```
        public frmPrintbill()
{
            InitializeComponent();
        }

        private void frmPrintbill_Load(object sender, EventArgs e)
        {
            lblOrderDate.Text = DateTime.Now.ToString();
            lblOrderID.Text = orderID.ToString();
            lblTotal.Text = totalCost.ToString("c").Trim('$');
```

Put Date Time, order ID and total Cost. Trim in their label

```
            dataGridView1.DataSource = dt;
```

Put data in = dataGridView1.DataSource

```
 dataGridView1.Columns.RemoveAt(0);
```

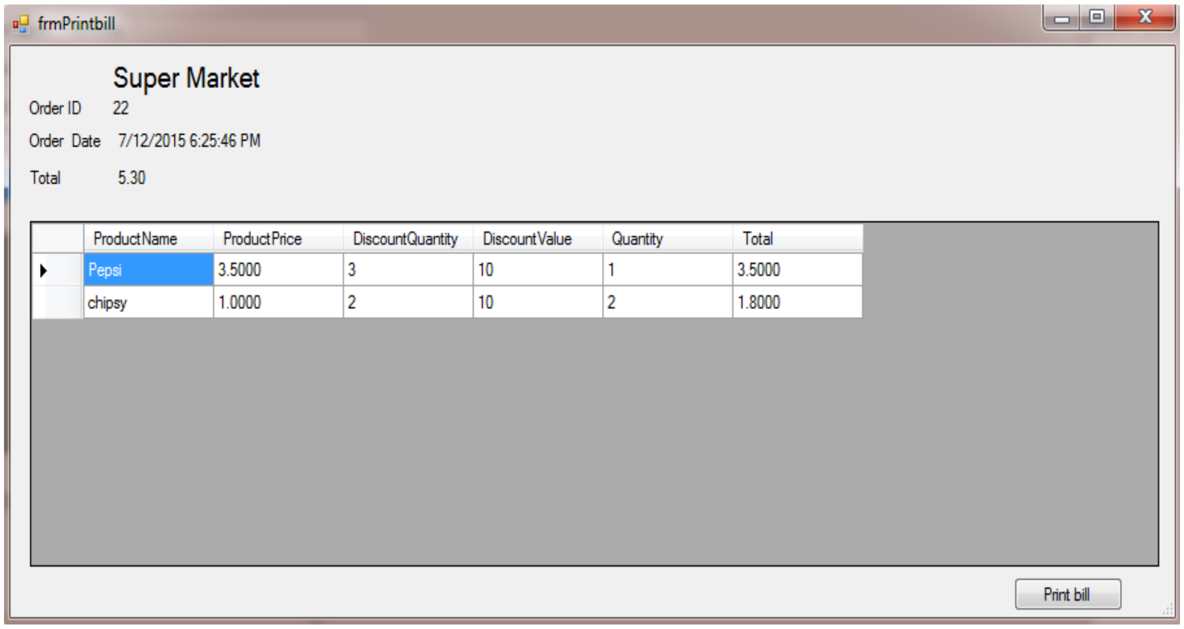
To delete column at specific position from dataGridView1

```
        }

        private void btnPrint_Click(object sender, EventArgs e)
        {
            this.Close();
            this.Dispose();
        }   }}
```

Button of print

To close

Result



Super Market

Order ID	22
Order Date	7/12/2015 6:25:46 PM
Total	5.30

ProductName	ProductPrice	DiscountQuantity	DiscountValue	Quantity	Total
Pepsi	3.5000	3	10	1	3.5000
chipsy	1.0000	2	10	2	1.8000

Fig4.26 Form Print bill

//

SQL:

USE [RFIDNotificationService]

GO

/****** Object: StoredProcedure [dbo].[SaveOrder] Script Date: 07/13/2015 03:42:45 ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

-- ===

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

```
-- Description: <Payment process --> add order --> transfer products to history,,>

-- =============================================

ALTER PROCEDURE [dbo].[SaveOrder]

        -

        @CustomerName nvarchar(250) = " ,

        @CustomerEmail nvarchar(250) = ",

        @CustomerPhone nvarchar(250) = "

        --  <@Param2,  sysname,  @p2>  <Datatype_For_Param2,  ,  int>  =
<Default_Value_For_Param2, , 0>
```

Add the parameters for the stored procedure here

```
AS

BEGIN

        -- SET NOCOUNT ON added to prevent extra result sets from

        -- interfering with SELECT statements.

        SET NOCOUNT ON;

        Declare @OrderID int;
```

Declare variables

Insert order details

```
INSERTINTO
[RFIDNotificationService].[dbo].[Order]([CustomerName],[CustomerEmail],[CustomerPhone])
VALUES (@CustomerName,@CustomerEmail,@CustomerPhone);




 set @OrderID = (select max(OrderID) from [Order]

 -
```

Copy scanned products to history

```
INSERT INTO dbo.[History] (ID,OrderID,ProductSerial,OrderDate)


SELECT          ProductSerial_ID.ID,   [Order].OrderID,   NotificationLog.SerialNumber,
NotificationLog.NotificationLogDate

FROM      NotificationLog INNER JOIN

                ProductSerial_ID  ON  NotificationLog.SerialNumber  =  ProductSerial_ID.Serial
CROSS JOIN

                [Order]

                  where OrderID = @OrderID;

 Delete from ProductSerial_ID where Serial in (select SerialNumber from  NotificationLog);


 Delete from NotificationLog;
```

Delete scanned products

```
END
```

Chapter 5: Conclusion and Future work

CONCLUSION

The application of RFID technique in the checkout system will achieve improvement in services provided to the customer during shopping. Checkout system using RFID provides each of the cashier, customers and approved by retail themselves for more benefits. According to the design, which we implemented.

For the cashier, It make cashier able to collect different information about the product without having to go to the store and inventory work Such as The expire date for each product and, any product closer to be expired, remaining quantity of each product in the store, best-selling products, Supermarket revenue in a certain day, any goods which its Quantity approached completion. For the customer, saving time result for there is no long line to wait at the cashier where RFID reader scans for all products at one time, leading to save a lot of time and make checkout more comfortable. For retailers, application of this system is useful for retail dealers themselves so that it works to reduce theft by force the client to pass through the exit gate (the reader is fixed on it) as soon as the thief pass through it system open a new account and prints a bill, They can also find out store revenues, moment by moment_as the design that we have implemented _can collect revenue automatically without the intervention of the cashier.

FUTURE WORK

It is proposed that a special database for products contain information about products [as example foods] such as caloric value , food that cause allergies ,a recipe for a certain meal preparation and the customer is able to read this information, thus making shopping more enjoyable for the client.
So far, the price of the tag fairly high and this is one of the most important reasons for non-application of this system.
It is proposed that the manufacturer put tags on its product (for each product) so that the product up to the retail traders comes with (its own tag) as is the case in the barcode.

Make it possible for the customer to replace usable tags again versus the amount of material or another commodity.

Connect the checkout system with electronic gate so that when the gate opened and the passage of the client which its goods, which had read by the reader to prevent or reduce the theft.

References :-

[1] http://rfid-handbook.de/downloads/E3E_0470695064.pdf

[2] http://atlasrfid.com/jovix-education/auto-id-basics/rfid-vs-barcode/

[3]http://www.radio-electronics.com/info/wireless/radio-frequency-identification-rfid/low-high-frequency- bands-frequencies.php

[4] https://nxp-rfid.com/applications/

[5] http://www.simonsothcott.com/2011/11/what-is-rfid-10-examples-of-rfid.html

[6]http://nodisinfo.com/fda-approved-rfid-tags-human-use/

[7]http://www.rfidjournal.com/articles/view?781.

[8]http://www.tutorial-reports.com/wireless/rfid/technology.

[9]http://www.rfidjournal.com/article/articleview/258/1/1/.

[10]Krane, J. (2003). Benetton clothing to carry tiny tracking transmitters. Associated Press. Albrecht, K., and McIntyre, L. (2005).

[11] Rieback, M.R., Crispo, B., Tanenbaum, A.S. (2006). Is your cat infected with a computer virus? Pervasive Computing and Communications. IEEE Press. pp 169-179.

[12]Stockman, H. (1948). Communication by Means of Reflected Power. Proceedings of the Institute of Radio Engineers. October. pp 1196- 1204.

[13]http://ijcsi.org/papers/IJCSI-10-1-1-516-521.pdf

[13]http://www.mlo-online.com/articles/201110/automation-in-blood-banking-rfid.

[14]http://retailsmanagement.geo-viz.com/blog/top-challenges-faced-by-grocery-stores-focus-retail-software/

[15]Library of Automation, loc. cit.

[16]Grieco, P. T, et. al. 1989. *Behind Bars: bar coding principles and applications*, PT Publications,
Florida. p. 9.

[17]Granneman, S. 2003. *RFID Chips Are Here.* The Register. [Online]. <URL:

http://www.theregister.co.uk/content/archive/31461.html> Last accessed 21/3/2004.

[18]http://rfid-handbook.de/downloads/E3E_0470695064.pdf

[19]http://atlasrfid.com/jovix-education/auto-id-basics/rfid-vs-barcode/

[20] http://www.radio-electronics.com/info/wireless/radio-frequency-identification-rfid/low- high-frequency-bands-frequencies.php

[21] https://nxp-rfid.com/applications/

[22]http://www.simonsothcott.com/2011/11/what-is-rfid-10-examples-of-rfid.html

[23] http://nodisinfo.com/fda-approved-rfid-tags-human-use/

Appendix :-

The code

Welcome

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Sales_System
{
   public partial class frmWelcome : Form
   {
      public frmWelcome()
      {
         InitializeComponent();
      }

      private void btnCustomer_Click(object sender, EventArgs e)
      {
         frmOrder order = new frmOrder();
         order.Show();
         Program._User = true;
      }

      private void btnCachier_Click(object sender, EventArgs e)
      {
         frmLogin login = new frmLogin();
         login.Show();
         Program._User = false;

      }
   }
}
////////////////////////////////
```

Login

```
    public partial class frmLogin : Form
  {
    public frmLogin()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      try
      {
        string ID = txtUserID.Text.Trim();
        string Password = txtPassword.Text.Trim();
        //Check database for userID and password

        string cmd = string.Format("select ISNULL(( select 1 as 'Result' from dbo.myUsers
where ID = {0} and [Password] = '{1}'),0)",ID,Password);
        //string cmd = "select [Username] as 'Name' from dbo.myUsers where ID = " + ID + "
and [Password] = " + Password;
        clsDataAccessLayer DAL = new clsDataAccessLayer();
        int result = (int)DAL.selectValue(cmd);

        if (result == 1)
        {
          Main_Window frm = new Main_Window();
          frm.Show();
          this.Hide();
          Program._User = false;
        }
        else
        {
          Program._User = true;
          MessageBox.Show("Check your username and
password","Error",MessageBoxButtons.OK);

        }
      }
      catch (Exception)
      {

        throw;
      }
}

    private void label1_Click(object sender, EventArgs e)
    {

    }

    private void txtUserID_TextChanged(object sender, EventArgs e)
    {

    }
  }
}
```

///////////////
Main Window

```
public partial class Main_Window : Form
   {
      public Main_Window()
      {
         InitializeComponent();
      }

      private void btnOrders_Click(object sender, EventArgs e)
      {
         frmOrder Orders = new frmOrder();
         Orders.Show();
      }

      private void btnProducts_Click(object sender, EventArgs e)
      {
         frmAddProduct frm = new frmAddProduct();
         frm.Show();
      }

      private void btnReports_Click(object sender, EventArgs e)
      {
         frmReports frm = new frmReports();
         frm.Show();
      }
   }
}
///////////////
```

Order

```
namespace Sales_System
{
   public partial class frmOrder : Form
   {
      clsDataAccessLayer DAL;

      decimal total = 0; // for total price
      public decimal Total
      {
         get { return total; }
         set { total = value; }
      }

      bool PaymentStatus = false;


      private void GeneralSetting()
      {
         dataGridView1.AutoGenerateColumns = false;
         dataGridView1.MultiSelect = false;
         dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
         dataGridView1.RowsDefaultCellStyle.BackColor = Color.WhiteSmoke;
         dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.White;

      }
      public frmOrder()
      {
         InitializeComponent();
         DAL = new clsDataAccessLayer();
         GeneralSetting();
      }

      //Cancel the order
      private void btnCancelOrder_Click(object sender, EventArgs e)
      {
         DialogResult result = MessageBox.Show("Are you sure you want to cancel the order?",
"warning", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
         if (result == System.Windows.Forms.DialogResult.Yes)
         {
            // 1- delete items from DB
            string cmd = "Delete from NotificationLog";
            DAL.ExecuteNonQuery(cmd);
            // 2- Clear form
            dataGridView1.DataSource = null;
            lblOrderID.Text = "";
            lblTotal.Text = "";
            txtCustomerName.Text = "";
            MessageBox.Show("Operation Complete", "Info", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

    }

    // start new order
    private void btnNewCustomer_Click(object sender, EventArgs e)
    {
        //1- Generate Order ID
        string cmd = "select max(OrderID)+ 1 from [Order]";
        lblOrderID.Text =  DAL.selectValue(cmd).ToString();
        //2- Get Scanned item from DB
        cmd = "SELECT     ProductSerial_ID.ID, Product.ProductName, Product.ProductPrice,
Product.DiscountQuantity, Product.DiscountValue, count(NotificationLog.SerialNumber) AS
'Quantity' FROM Product INNER JOIN ProductSerial_ID ON Product.ID = ProductSerial_ID.ID
INNER JOIN NotificationLog ON ProductSerial_ID.Serial = NotificationLog.SerialNumber group
by  ProductSerial_ID.ID, Product.ProductName, Product.ProductPrice,
Product.DiscountQuantity, Product.DiscountValue";
        DataTable dt = DAL.SelectTable(cmd);
        if (dt.Rows.Count > 0)
        {
            dt.Columns.Add("Total", typeof(decimal));
            // fill gridview with data
            dataGridView1.DataSource = dt;
           // decimal total = 0; // for total price
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                int Quantity = Convert.ToInt32(dt.Rows[i][5].ToString());
                int DiscountQuantity = Convert.ToInt32(dt.Rows[i][3].ToString());
                int Discountvalue = Convert.ToInt32(dt.Rows[i][4].ToString());
                decimal ItemPrice = Convert.ToDecimal(dt.Rows[i][2].ToString());

                decimal ProductTotal = ItemPrice * Quantity;



                if (Quantity >= DiscountQuantity)
                {
```

```
                dt.Rows[i][6] = ProductTotal - (ProductTotal * Discountvalue) / 100;
              }
              else
              {
                dt.Rows[i][6] = ProductTotal;

              }

              total += Convert.ToDecimal(dt.Rows[i][6].ToString());


            }
            // show total price
            lblTotal.Text = total.ToString();

          }

        }

        //Save order to DB
        private void btnSaveOrder_Click(object sender, EventArgs e)
        {
          if (dataGridView1.Rows.Count < 1)
          {
            MessageBox.Show("No items to buy", "Info", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
            return;
          }
          frmPayMethod _frmPayMethod = new frmPayMethod();
          _frmPayMethod.TotalCost = this.total;
          _frmPayMethod.ShowDialog(this);
          this.PaymentStatus = _frmPayMethod.PaymentStatus;

          if (PaymentStatus == true)
          {

            //// 1- save customer data(Create new Order ID)
            string cmd = string.Format("EXECUTE [RFIDNotificationService].[dbo].[SaveOrder]
    @CustomerName  = '{0}', @CustomerPhone = '{1}', @CustomerEmail= '{2}'",
    txtCustomerName.Text.Trim(),txtCustomerPhone.Text,txtCustomerEmail.Text);
            DAL.ExecuteNonQuery(cmd);


            //DAL.ExecuteNonQuery(cmd);
            // 2- clear scanned item from db
            //cmd = "Delete from NotificationLog";
            //DAL.ExecuteNonQuery(cmd);


            frmPrintbill print = new frmPrintbill();

            print.TotalCost = this.Total;
            print.OrderID = Convert.ToInt32( lblOrderID.Text);
            print.dt = (DataTable)dataGridView1.DataSource;
            print.ShowDialog(this);


            // 3- clear form
            dataGridView1.DataSource = null;
            lblOrderID.Text = "";
            lblTotal.Text = "";
            txtCustomerName.Text = "";
            txtCustomerPhone.Text = "";
            txtCustomerEmail.Text = "";
            Total = 0;

            MessageBox.Show("Operation Complete", "Info", MessageBoxButtons.OK,
MessageBoxIcon.Information);

            if (Program._User == false)
            {
                cmd = "SELECT ProductSerial_ID.ID, Product.ProductName
,COUNT(ProductSerial_ID.Serial)as 'Quantity' FROM ProductSerial_ID INNER JOIN Product ON
ProductSerial_ID.ID = Product.ID GROUP BY ProductSerial_ID.ID, Product.ProductName HAVING
(COUNT(ProductSerial_ID.Serial) < 6)";
                //clsDataAccessLayer DAL = new clsDataAccessLayer();
                DataTable dt = DAL.SelectTable(cmd);
                if (dt.Rows.Count > 0)
                {
                    Alert alert = new Alert();
                    alert.ShowDialog();
                }

            }
        }
        else
        {
            MessageBox.Show("Payment operation did not complete.", "Info",
MessageBoxButtons.OK, MessageBoxIcon.Information);

         }
      }

      private void frmOrder_Load(object sender, EventArgs e)
      {

      }

      private void button1_Click(object sender, EventArgs e)
      {
         this.Close();
         frmLogin welcome = new frmLogin();
         welcome.Show();
      }

      //delete item
      private void button1_Click_1(object sender, EventArgs e)
      {
         if (dataGridView1.Rows.Count > 0)
         {

            if (MessageBox.Show("Please scan the item you want to delete.", "Delete Item",
MessageBoxButtons.OKCancel, MessageBoxIcon.Information) ==
System.Windows.Forms.DialogResult.OK)
            {
               string cmd = string.Format("delete from NotificationLog where SerialNumber =
(SELECT top 1 SerialNumber FROM NotificationLog  group by SerialNumber having
(count(NotificationLogId) > 1))");
               DAL.ExecuteNonQuery(cmd);
            }
            this.Total = 0;
            btnNewCustomer_Click(null, null);
         }
         else
            MessageBox.Show("No items to delete", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);

        /*
         if (dataGridView1.Rows.Count > 0)
         {
            int index = dataGridView1.CurrentRow.Index;
            int Quantity = Convert.ToInt32(dataGridView1.CurrentRow.Cells[3].Value.ToString());
            if (Quantity > 1)
            {
                dataGridView1.CurrentRow.Cells[3].Value = Quantity - 1;

            }
            else if (Quantity == 1)
            {

            }
        }
        */
        //lblTotal.Text = (Convert.ToDecimal(lblTotal.Text) -
Convert.ToDecimal(dataGridView1.CurrentRow.Cells[2].Value.ToString())).ToString();


        //Get item price string from grid
        //get total from lable string

        // convert 2 price to decimal

        // substraction

        // update lable with total price



    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {

    }
  }
}
///////////////////
Add product

public partial class frmAddProduct : Form
  {
    private void LoadProductData()
    {
        string cmd = "SELECT [ID] ,str([ID])+' | '+[ProductName]  as 'ProductName' FROM
[Product]";
```

```
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = new DataTable();
            dt = DAL.SelectTable(cmd);
            cmbProduct.DisplayMember = "ProductName";
            cmbProduct.ValueMember = "ID";
            cmbProduct.DataSource = dt;

        }
        public frmAddProduct()
        {
            InitializeComponent();
            LoadProductData();
        }

        //Scan tag serial
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                string cmd = "select isnull((SELECT TOP 1 [SerialNumber] FROM [NotificationLog]
where [SerialNumber] not in (select [Serial] from [ProductSerial_ID]) order by
[NotificationLogDate] desc),'')";
                clsDataAccessLayer DAL = new clsDataAccessLayer();
                string result = DAL.selectValue(cmd).ToString();
                if (string.IsNullOrEmpty(result))
                {
                    MessageBox.Show("No new tag to add, Please scan a valid tag", "Error",
MessageBoxButtons.OK,MessageBoxIcon.Error);
                    return;
                }
                else
                    txtSerial.Text = result;
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);

            }


        }

        private void button2_Click(object sender, EventArgs e)
        {
```

```
string name = txtProductName.Text;
if (string.IsNullOrEmpty(name))
{
   MessageBox.Show("Enter product name");
   return;
}

string Price = txtProductPrice.Text;
if (string.IsNullOrEmpty(Price))
{
   MessageBox.Show("Enter product price");
   return;
}
double d_price;

if (!double.TryParse(Price,out d_price))
{
   MessageBox.Show("Enter product correct price");
   return;
}

string DiscountQuantity = txtDiscountQuantity.Text;
if (string.IsNullOrEmpty(DiscountQuantity))
{
   MessageBox.Show("Please enter the Discount Quantity");
   return;
}

string DiscountValue = txtDiscountValue.Text;
if (string.IsNullOrEmpty(DiscountValue))
{
   MessageBox.Show("Please enter the DiscountValue");
   return;
}


string cmd = string.Format("INSERT INTO [Product]([ProductName],[ProductPrice]
,[DiscountQuantity],[DiscountValue])VALUES
('{0}',{1},{2},{3})",name,d_price,DiscountQuantity,DiscountValue);
clsDataAccessLayer DAL = new clsDataAccessLayer();
DAL.ExecuteNonQuery(cmd);

LoadProductData();
```

```
            txtDiscountQuantity.Text = "";
            txtProductName.Text = "";
            txtProductPrice.Text = "";
            txtDiscountValue.Text  = "";

            MessageBox.Show("Save Done");
        }

        private void btnSaveTag_Click(object sender, EventArgs e)
        {
            try
            {
                string Serial = txtSerial.Text;
                if (string.IsNullOrEmpty(Serial))
                {
                    MessageBox.Show("Please scan the tag");
                    return;
                }


                    int id = Convert.ToInt32(cmbProduct.SelectedValue);
                    string cmd = string.Format("INSERT INTO [ProductSerial_ID]([ID]
,[Serial],[ExpireDate]) VALUES ({0},'{1}','{2}')",id,Serial,dateTimePicker1.Value);
                    clsDataAccessLayer DAL = new clsDataAccessLayer();
                    DAL.ExecuteNonQuery(cmd);
                    cmd = "Delete from NotificationLog";
                    DAL.ExecuteNonQuery(cmd);
                    txtSerial.Text = "";
                    MessageBox.Show("Save Done");


            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);
            }
        }

        private void frmAddProduct_Load(object sender, EventArgs e)
        {

        }
```

```
    }
////////////////////
```

Pay

```
public partial class frmPayMethod : Form
    {
        private decimal totalCost;

        public decimal TotalCost
        {
            get { return totalCost; }
            set { totalCost = value; }
        }

        public bool PaymentStatus;


        public frmPayMethod()
        {
            InitializeComponent();
        }

        private void btnCache_Click(object sender, EventArgs e)
        {
            frmPayCache _frmPayCache = new frmPayCache();
            _frmPayCache.TotalCost = this.totalCost;
            _frmPayCache.ShowDialog(this);
            this.PaymentStatus = _frmPayCache.PaymentStatus;
            this.Close();
            this.Dispose();
        }

        private void btnCard_Click(object sender, EventArgs e)
        {
            frmPayWithCard _frmPayWithCard = new frmPayWithCard();
            _frmPayWithCard.TotalCost = this.totalCost;
            _frmPayWithCard.ShowDialog(this);
            this.PaymentStatus = _frmPayWithCard.PaymentStatus;
            this.Close();
            this.Dispose();

        }
    }
}
////////////////
```

Cache

```
public partial class frmPayMethod : Form
  {
    private decimal totalCost;

    public decimal TotalCost
    {
      get { return totalCost; }
      set { totalCost = value; }
    }

    public bool PaymentStatus;


    public frmPayMethod()
    {
      InitializeComponent();
    }

    private void btnCache_Click(object sender, EventArgs e)
    {
      frmPayCache _frmPayCache = new frmPayCache();
      _frmPayCache.TotalCost = this.totalCost;
      _frmPayCache.ShowDialog(this);
      this.PaymentStatus = _frmPayCache.PaymentStatus;
      this.Close();
      this.Dispose();
    }

    private void btnCard_Click(object sender, EventArgs e)
    {
      frmPayWithCard _frmPayWithCard = new frmPayWithCard();
      _frmPayWithCard.TotalCost = this.totalCost;
      _frmPayWithCard.ShowDialog(this);
      this.PaymentStatus = _frmPayWithCard.PaymentStatus;
      this.Close();
      this.Dispose();

    }
  }
}
///////////////
```

Credit card

```
public partial class frmPayWithCard : Form
```

```
  {

      private decimal totalCost;

      public decimal TotalCost
      {
         get { return totalCost; }
         set
         {
            totalCost = value;
            lblTotal.Text = totalCost.ToString("c").TrimStart('$');
         }
      }

      public bool PaymentStatus;

      string CardID = "";


      public frmPayWithCard()
      {
         InitializeComponent();
      }

      private void btnScan_Click(object sender, EventArgs e)
      {
         clsDataAccessLayer DAL = new clsDataAccessLayer();
         try
         {
            //Get tag Serial from DB
            string cmd = "SELECT top 1 [SerialNumber]  FROM NotificationLog order by
NotificationLogDate desc";
            CardID = DAL.selectValue(cmd).ToString();
            txtCardID.Text = CardID;

            //Get Tag credit from DB
            cmd = string.Format("SELECT [Amount] FROM [CreditCards] where [Serial] = '{0}'",
CardID);
            txtCredit.Text = DAL.selectValue(cmd).ToString();
         }
         catch(Exception exp)
         {
            MessageBox.Show(exp.Message);
         }
```

```
        }

        private void btnPay_Click(object sender, EventArgs e)
        {
            if (Convert.ToDecimal(txtCredit.Text) >= TotalCost)
            {
                string cmd = string.Format("update [CreditCards] set [Amount] = [Amount] -
{0}",totalCost);
                clsDataAccessLayer DAL = new clsDataAccessLayer();
                DAL.ExecuteNonQuery(cmd);
                this.PaymentStatus = true;
                this.Close();
                this.Dispose();
            }
            else
            {
                MessageBox.Show("No enough credit");
                this.PaymentStatus = false;
            }
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.PaymentStatus = false;
            this.Close();
            this.Dispose();
        }
    }
}
/////////////////
```

Reports

```
  public partial class frmReports : Form
    {

        private void GeneralSetting()
        {
            /*
            dataGridView1.AutoGenerateColumns = false;
            dataGridView1.MultiSelect = false;
            dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            dataGridView1.RowsDefaultCellStyle.BackColor = Color.WhiteSmoke;
            dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.White;
            */
        }
        private void LoadTopProducts()
        {
            string cmd = "SELECT top 10  History.ID, COUNT(History.ProductSerial) AS 'Quantity',
Product.ProductName, Product.ProductPrice FROM History INNER JOIN Product ON History.ID =
Product.ID GROUP BY History.ID, Product.ProductName, Product.ProductPrice order by
Quantity desc";
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView1.DataSource = dt;
        }

        private void LoadStockData()
        {
            string cmd = "SELECT ProductSerial_ID.ID, count(ProductSerial_ID.Serial) as 'Quantity',
Product.ProductName, Product.ProductPrice FROM ProductSerial_ID INNER JOIN Product ON
ProductSerial_ID.ID = Product.ID group by ProductSerial_ID.ID,Product.ProductName,
Product.ProductPrice";
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView2.DataSource = dt;

        }
        void Alert()
        {
            string cmd = "SELECT ProductSerial_ID.ID, Product.ProductName
,COUNT(ProductSerial_ID.Serial)as 'Quantity' FROM ProductSerial_ID INNER JOIN Product ON
ProductSerial_ID.ID = Product.ID GROUP BY ProductSerial_ID.ID, Product.ProductName HAVING
(COUNT(ProductSerial_ID.Serial) < 6)";
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView3.DataSource = dt;
        }

        void LoadExpireProducts()

        {
            string cmd = "SELECT ProductSerial_ID.Serial, Product.ID, Product.ProductName,
ProductSerial_ID.ExpireDate FROM ProductSerial_ID INNER JOIN Product ON
ProductSerial_ID.ID = Product.ID WHERE     (ProductSerial_ID.ExpireDate < GETDATE())";
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView4.DataSource = dt;
        }

        void LoadTotalPerDay()
        {
            string cmd = "SELECT  RIGHT( str(day(History.OrderDate)),2)+'_'+
str(datepart(month,History.OrderDate)) +'_'+ str(datepart(year,History.OrderDate)) ,
SUM(Product.ProductPrice) FROM History INNER JOIN Product ON History.ID = Product.ID
where day(History.OrderDate) = DAY(GETDATE()) group by RIGHT(
str(day(History.OrderDate)),2)+'_'+ str(datepart(month,History.OrderDate)) +'_'+
str(datepart(year,History.OrderDate))";
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView5.DataSource = dt;

        }

        public frmReports()
        {
            GeneralSetting();

            InitializeComponent();


            LoadStockData();

            LoadTopProducts();

            Alert();

            LoadExpireProducts();

            LoadTotalPerDay();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DateTime t = dateTimePicker1.Value;

            string cmd = string.Format("SELECT  RIGHT( str(day(History.OrderDate)),2)+'_'+
str(datepart(month,History.OrderDate)) +'_'+ str(datepart(year,History.OrderDate)) ,
SUM(Product.ProductPrice) FROM History INNER JOIN Product ON History.ID = Product.ID
where day(History.OrderDate) = {0} group by RIGHT( str(day(History.OrderDate)),2)+'_'+
str(datepart(month,History.OrderDate)) +'_'+ str(datepart(year,History.OrderDate))",t.Day);
            clsDataAccessLayer DAL = new clsDataAccessLayer();
            DataTable dt = DAL.SelectTable(cmd);
            dataGridView5.DataSource = dt;
        }


      }
}
```
////////////////

Printbill

```
public partial class frmPrintbill : Form
  {
    private decimal totalCost;

    public decimal TotalCost
    {
      get { return totalCost; }
      set {

        totalCost = value; }
    }

    private int orderID;

    public int OrderID
    {
      get { return orderID; }
      set { orderID = value; }
    }

    public DataTable dt = new DataTable();

    public frmPrintbill()
    {
      InitializeComponent();
    }

    private void frmPrintbill_Load(object sender, EventArgs e)
    {
      lblOrderDate.Text = DateTime.Now.ToString();
      lblOrderID.Text = orderID.ToString();
      lblTotal.Text = totalCost.ToString("c").Trim('$');
```

```
            dataGridView1.DataSource = dt;

            dataGridView1.Columns.RemoveAt(0);

        }

        private void btnPrint_Click(object sender, EventArgs e)
        {
            this.Close();
            this.Dispose();
        }
    }
}
```