# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

# BELAGAVI, KARNATAKA - 590018

FILE STRUCTURE LABORATARY MINI PROJECT REPORT ON

# “SUPERMARKET INVENTORY AND SALES MANAGEMENT SYSTEM”

# Submitted by

|  |  |
| --- | --- |
|  |  |
| **SARIN ALEXANDER** | **4DM18IS042** |
| **SUFAIL KHALID PKC** | **4DM18IS052** |
| **ADARSH KV** | **4DM18IS003** |
| **MINTO MOHAN** | **4DM18IS023** |

UNDER THE GUIDANCE OF

**Dr. Keerthi Kumar H M**

**Head of the department, Dept. Of ISE**

***In the partial fulfilment for the award of the degree of***

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

|  |
| --- |
|  |



**YENEPOYA INSTITUTE OF TECHNOLOGY**

N.H. 13, Thodar, Vidyanagar, Moodbidri, Mangalore,

Karnataka - 574225

**2020-2021**

### YENEPOYA INSTITUTE OF TECHNOLOGY

N.H.13, Thodar, Vidyanagar, Moodbidri, Mangalore, Karnataka 574225

(Affiliated of Visvesvaraya Technological University)

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**



**CERTIFICATE**

Certified that the file structure laboratary Mini Project Work titled **“****SUPERMARKET INVENTORY AND SALES MANAGEMENT SYSTEM** carried out by **Mr. Sarin Alexander USN 4DM18IS042, Mr Sufail Khalid PKC USN 4DM18IS052, Adarsh KV USN 4DM18IS003** and **Minto Mohan USN 4DM18IS023** bonafide student of **Yenepoya Institute Of Technology** in partial fulfillment for the award of **Bachelor of Engineering in Information Science & Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year **2020-21**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

**………………………. ….…………………**

**Dr.Keerthi Kumar H M Dr.Keerthi Kumar H M**

**Internal guide Head of the Department**

**Dept. of ISE Dept. of ISE**

**External Viva**

**Name of the examiners Signature with date**

**1.**

**2.**

# ABSTRACT

The supermarket management system has realized the transmission and control of large goods, so as to facilitate the management and decision of sales, and reduce a big burden for supermarkets and supermarket managers. It also can help to improve the work efficiency of supermarket.

Its requirements are to provide the basic information maintenance function of employees, memberships and products so that managers can through the function to add, delete, and modify the basic information of employees and the employees can through it to add, modify and delete the basic information of memberships and goods.

Supermarket management system is very convenient for manage, input, output, and find the data so as to make the messy supermarket data to specific, visualizations, rationalization. In the aspect of software, the supermarket management system using Python language and text file to store the data. In the aspect of software, various configurations in computer including input and output capacity, internal memory and external memory capacity can meet the requirements of users.

### ACKNOWLEDGMENT

The successful completion of any work would be incomplete without a mention of the people who made it possible, whose constant guidance and encouragement served as a beacon light and crowned our efforts with success. We owe our gratitude to many people who helped and supported us during our File Structure laboratory Mini Project “SUPERMARKET INVENTORY AND SALES MANAGEMENT SYSTEM”.

I sincerely express our gratitude to Dr. Keerthi Kumar H M, H.O.D, Dept. of ISE for his constant support and guidance for the successful completion of File Structure laboratory Mini Project and encouragement by providing with the necessary advices and help. We are highly indebted to him for taking keen interest in my work, monitoring and providing guidance throughout the course.

.

I take immense pleasure in thanking my beloved Principal Dr. R. G D’Souza for his constant support.

I also thank all laboratory administrates and assistants who have helped us in making this project a successful one.

At last, but not the least I want to thank our classmates and friends who appreciated our work and motivated us.

**SARIN ALEXANDER 4DM18IS042 SUFAIL KHALID PKC 4DM18IS052 ADARSH KV 4DM18IS003 MINTO MOHAN 4DM18IS023**

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | |
| **Certificate** |  |
| **Abstract** | **i** |
| **Acknowledgment** | **ii** |
| **Table of Contents** | **iii** |

1. INTRODUCTION 1

Introduction to File Structures 1

[History 1](#_TOC_250010)

[About the File 2](#_TOC_250009)

[Various Kinds of Storage of Fields and Records 3](#_TOC_250008)

Applications of File Structure 7

1. **REQUIREMENTS SPECIFICATION 8**
2. SYSTEM ANALYSIS 9

[Analysis of the Project](#_TOC_250003) 9

Structure used to Store the Fields and Records 9

1. BACKEND DESIGN 10

4.1 Python 10

4.2 Data files and it’s fields 12

4.3 Data files 13

**5 FRONT-END DESIGN 16**

5.1 PyCharm IDE 16

5.2 Qt-Designer 16

5.3 Python 16

5.4 Pseudo Code 16

**6 RESULTS AND ANALYSIS 19**

**7 CONCLUSIONS 23**

**REFERENCE 24**

**Chapter 1**

# INTRODUCTION

The Supermarket Inventory System Management provides a user-friendly, interactive Graphical user interface (GUI). All data is stored in files for persistence. The mini project uses 5 files: An User file, to store the user data, a Brands file, to store the brands data, A Category file, to store the category data, A Products file, to store the products data, A Sales file, to store the sales data.

**Introduction to File Structure**

A file structure is a combination of representations for data in files and of operations for accessing the data. A file structure allows to read, write, and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order. An improvement in file structure design may make an application hundreds of times faster. The details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications.

**History**

Early work with files presumed that files were on tape, since most files were. Access was sequential, and the cost of access grew in direct proportion, to the size of the file. As files grew intolerably large for unaided sequential access and as storage devices such as hard disks became available, indexes were added to files.

The indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly. With key and pointer, the user had direct access to the large, primary file. But simple indexes had some of the same sequential flaws as the data file, and as the indexes grew, they too became difficult to manage, especially for dynamic files in which the set of key changes. In 1963, researches developed and elegant self adjusting binary tree structure, called AVL tree, for data in memory, with a balanced binary tree, dozens of accesses were required to find a record in even moderate-sized files.

A method was needed to keep a tree balanced when each node of thee tree was not a single record, as in a binary tree, but a file block containing dozens, perhaps even hundreds, of records took 10 years until a solution emerged in the form of a B-Tree.

Whereas AVL trees grow from the top down as records were added, B-Trees grew from the bottom up. B-Trees provided excellent access performance, but there was a cost: no longer could a file be accessed sequentially with efficiency. The problem was solved by adding a linked list structure at the bottom level of the B-Tree. The combination of a B- Tree and a sequential linked list is called a B+ tree.

B-Trees and B+ trees provide access times that grow in proportion to logkN, where N is the number of entries in the file and k is the number of entries indexed in a single block of the B+ Tree structure. This means that B+ Trees can guarantee that you can find 1 file entry among millions with only 3 or 4 trips to the disk. Further, B+Trees guarantee that as you add and delete entries, performance stays about the same.

Hashing is a good way to get what we want with a single request, with files that do not change size greatly over time. Hashed indexes were used to provide fast access to files. But until recently, hashing did not work well with volatile, dynamic files. Extendible dynamic hashing can retrieve information with 1 or at most 2 disk accesses, no matter how big the file became.

###### About the File

When we talk about a file on disk or tape, we refer to a particular collection of bytes stored there. A file, when the word is used in this sense, physically exists. A disk drive may contain hundreds, even thousands of these physical files.

From the standpoint of a system program, a file is somewhat like a telephone line connection to a telephone network. The program can receive bytes through this phone line or send bytes down it, but it knows nothing about where these bytes come from or where they go. The program knows only about its end of the line. Even though there may be thousands of physical files on a disk, a single program is usually limited to the use of only about 20 files. A file is a operation of computer which stores data, information, message, settings, or commands used with the computer program. It is created using system program on the computer.

The application program relies on the OS to take care of the details of the telephone switching system. It could be that bytes coming down the line into the program originate from a physical file they come from the keyboard or some other input device. Similarly, bytes the program sends down the line might end up in a file, or they could appear on the terminal screen or some other output device. Although the program doesn‟t

know where the bytes are coming from or where they are going, it does know which line it is using. This line is usually referred to as the logical file, to distinguish it from the physical files on the disk or tape.

###### Various Kinds of storage of Fields and Records

A field is the smallest, logically meaningful, unit of information in a file.

###### Field Structures

The four most common methods as shown in Fig. 1.1 of adding structure to files to maintain the identity of fields are:

 Force the fields into a predictable length.

 Begin each field with a length indicator.

 Place a delimiter at the end of each field to separate it from the next field.

 Use a“keyword=value” expression to identify each field and its contents.

Method 1: Fix the Length of Fields

In the above example, each field is a character array that can hold a string value of some maximum size. The size of the array is 1 larger than the longest string it can hold. Simple arithmetic is sufficient to recover data from the original fields.

The disadvantage of this approach is adding all the padding required to bring the fields up to a fixed length, makes the file much larger. We encounter problems when data is too long to fit into the allocated amount of space. We can solve this by fixing all the fields at lengths that are large enough to cover all cases, but this makes the problem of wasted space in files even worse. Hence, this approach isn‟t used with data with large amount of variability in length of fields, but where every field is fixed in length if there is very little variation in field lengths.

Method 2: Begin Each Field with a Length Indicator

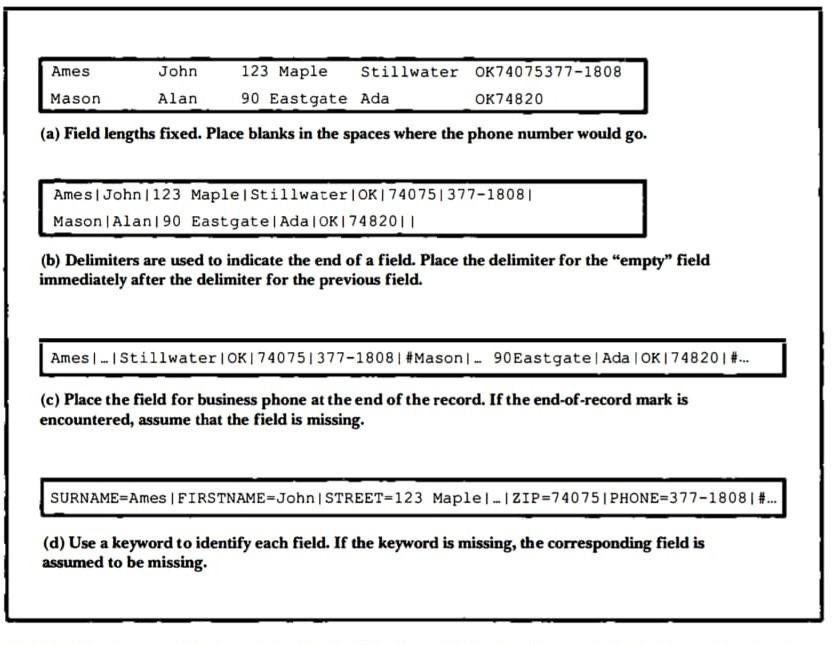
We can count to the end of a field by storing the field length just ahead of the field. If the fields are not too long (less than 256 bytes), it is possible to store the length in a single byte at the start of each field. We refer to these fields as length-based.

Method 3: Separate the Fields with Delimiters

We can preserve the identity of fields by separating them with delimiters. All we need to do is choose some special character or sequence of characters that will not appear within

a field and then insert that delimiter into the file after writing each field. White-space characters (blank, new line, tab) or the vertical bar character, can be used as delimiters.

Method 4: Use a “Keyword=Value” Expression to Identify Fields

This has an advantage the others don‟t. It is the first structure in which a field provides information about itself. Such self-describing structures can be very useful tools for organizing files in many applications. It is easy to tell which fields are contained in a file

###### Figure 1.1 Four methods for field structures

Even if we don‟t know ahead of time which fields the file is supposed to contain. It is also a good format for dealing with missing fields. If a field is missing, this format makes it obvious, because the keyword is simply not there. It is helpful to use this in combination with delimiters, to show division between each value and the keyword for the following field. But this also wastes a lot of space: 50% or more of the file‟s space could be taken up by the keywords.

A record can be defined as a set of fields that belong together when the file is viewed in terms of a higher level of organization.

###### Record Structures

The five most often used methods for organizing records of a file are

 Require the records to be predictable number of bytes in length.

 Require the records to be predictable number of fields in length.

 Begin each record with a length indicator consisting of a count of the number of bytes that the record contains.

 Use a second file to keep track of the beginning byte address for each record.

 Place a delimiter at the end of each record to separate it from the next record.

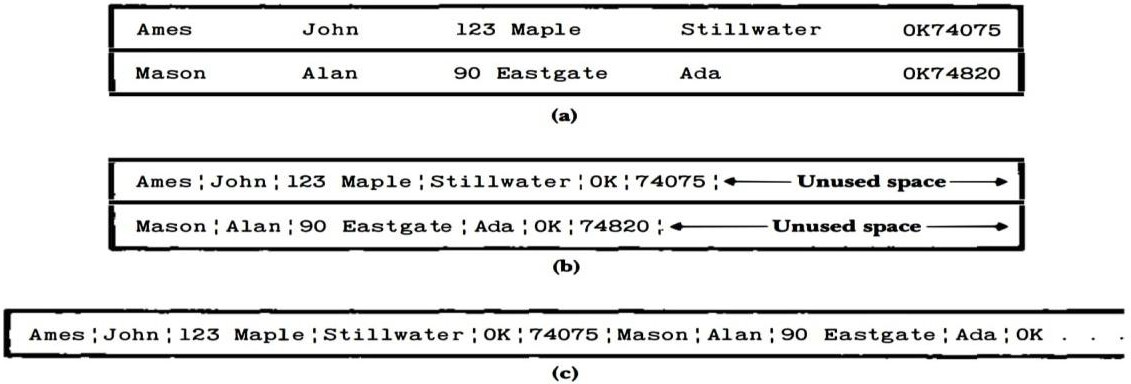
Method 1: Make the Records a Predictable Number of Bytes (Fixed-Length Record)

A fixed-length record file is one in which each record contains the same number of bytes. In the field and record structure shown, we have a fixed number of fields, each with a predetermined length, that combine to make a fixed-length record.

Fixing the number of bytes in a record does not imply that the size or number of fields in the record must be fixed. Fixed-length records are often used as containers to hold variable numbers of variable-length fields. It is also possible to mix fixed and variable-length fields within a record.

Method 2: Make Records a Predictable Number of Fields

Rather than specify that each record in a file contains some fixed number of bytes, we can specify that it will contain a fixed number of fields.



###### Figure 1.2 Making Records Predictable number of Bytes and Fields

Method 3: Begin Each Record with a Length Indicator

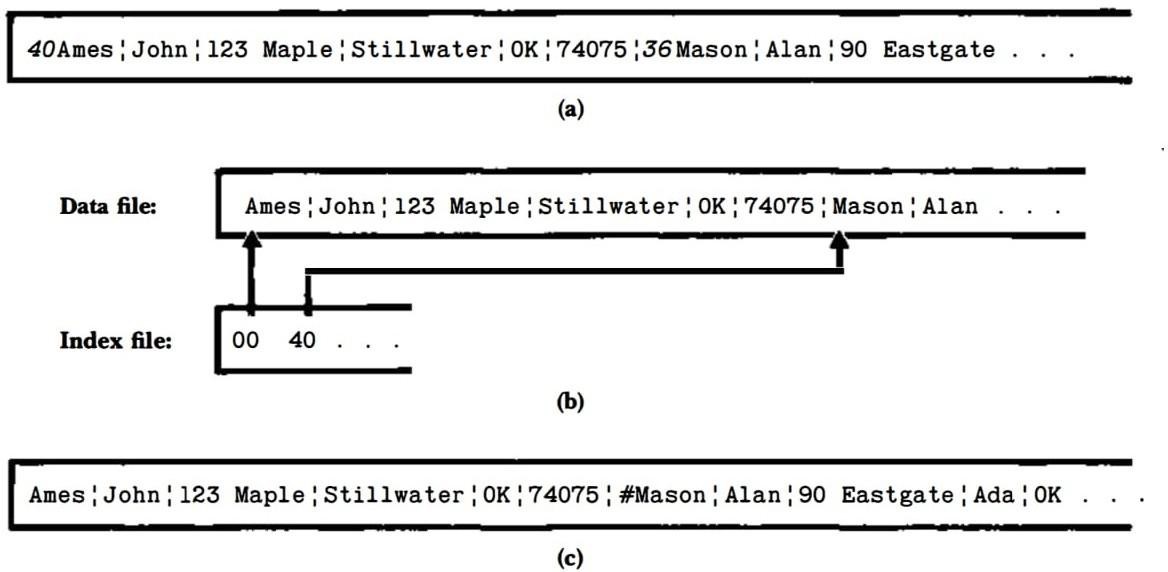
We can communicate the length of records by beginning each record with a filed containing an integer that indicates how many bytes there are in the rest of the record. This is commonly used to handle variable-length records.

Method 4: Use an Index to Keep Track of Addresses

We can use an index to keep a byte offset for each record in the original file. The byte offset allows us to find the beginning of each successive record and compute the length of each record. We look up the position of a record in the index, then seek to the record in the data file.

Method 5: Place a Delimiter at the End of Each Record

It is analogous to keeping the fields distinct. As with fields, the delimiter character must not get in the way of processing. A common choice of a record delimiter for files that contain readable text is the end-of-line character (carriage return/ new-line pair or, on Unix systems**,** just a new line character: „\n‟.



###### Figure 1.3 Using Length Indicator, Index and Record Delimiters

**Application of File Structure**

Relative to other parts of a computer, disks are slow. 1 can pack thousands of megabytes on a disk that fits into a notebook computer.

The time it takes to get information from even relatively slow electronic random access memory (RAM) is about 120 nanoseconds. Getting the same information from a typical disk takes 30 milliseconds. So, the disk access is a quarter of a million times longer than a memory access. Hence, disks are very slow compared to memory. On the other hand, disks provide enormous capacity at much less cost than memory. They also keep the information stored on them when they are turned off.

**Chapter 2**

**REQUIREMENTS SPECIFICATION**

In this chapter, hardware requirements have been discussed in section 2.1 and

software requirements have been discussed in 2.2

System Specification can be divided into two-

2.1 Hardware specifications

2.2 Software specifications

**Table 2.1: Specifications**

|  |  |
| --- | --- |
| **Hardware specifications** | **Software specifications** |
| 32/64-bit operating system | Operating System: Ubuntu/Windows |
| 2 GB RAM or above | Languages: Python |
| 40 GB hard disk or above | IDE: PyCharm |
| VGA COLOR Monitor | External tools: Notepad, Qt designer. |
| Keyboard | External libraries: PySide6 |
| Mouse |  |

**Chapter 3**

15ISL68: File Structures Mini Project Voter ID Management System

# SYSTEM ANALYSIS

#### Analysis of the Project

Supermarket Inventory and Sales Management System is used in any Supermarket. The system is initially used to add the details with all its specifications entered correctly into files, which is different for each and every product. The system can be used to search, delete, modify or display existing records of all the products.

#### Structure used to Store the Fields and Records

###### Fixed number of Fields:

In Supermarket inventory and sales management system, there are five files which contains different fields in it. All files contain fixed number of fields for each record for storing data

1. **Variable length record:**

Variable-length records are the records that vary in size. Our project uses variable length record system in all files. And each record is separated by a ‘\n’.

1. **Separate the fields with delimiters:**

The fields within a record are followed by a delimiting byte or series of bytes. Fields within a record can have different sizes. Different records can have different length fields. Programs which access the fields must know the delimiter. The delimiter cannot occur within the data. If used with delimited records, the field delimiter must be different from the record delimiter. Here the external overhead for field separation is equal to the size of the delimiter per field.

**Chapter 4**

**BACK-END DESIGN**

**4.1 Python**

In our project we use python as our backed language. All file handling and data retrieval and updates are done using python. Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but alike other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let’s start with Reading and Writing files.

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

**Open() function**

We use open () function in Python to open a file in read or write mode. As explained above, open ( ) will return a file object. To return a file object we use open() function along with two arguments, that accepts file name and the mode, whether to read or write. So, the syntax being: open(filename, mode). There are three kinds of mode, that Python provides and how files can be opened:

“ r “, for reading.

“ w “, for writing.

“ a “, for appending.

“ r+ “, for both reading and writing

One must keep in mind that the mode argument is not mandatory. If not passed, then Python will assume it to be “ r ” by default. Let’s look at this program and try to analyze how the read mode works:

# a file named "geek", will be opened with the reading mode.

file = open('geek.txt', 'r')

# This will print every line one by one in the file

for each in file:

print (each)

**read() function**

There is more than one way to read a file in Python. If you need to extract a string that contains all characters in the file then we can use file.read(). The full code would work like this:

# Python code to illustrate read() mode

file = open("file.text", "r")

print (file.read())

Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

# Python code to illustrate read() mode character wise

file = open("file.txt", "r")

print (file.read(5))

Creating a file using write() mode

Let’s see how to create a file and how write mode works:

To manipulate the file, write the following in your Python environment:

# Python code to create a file

file = open('geek.txt','w')

file.write("This is the write command")

file.write("It allows us to write in a particular file")

file.close()

The close() command terminates all the resources in use and frees the system of this particular program.

**append() function**

Let’s see how the append mode works:

# Python code to illustrate append() mode

file = open('geek.txt','a')

file.write("This will add this line")

file.close()

There are also various other commands in file handling that is used to handle various tasks like:

rstrip(): This function strips each line of a file off spaces from the right-hand side.

lstrip(): This function strips each line of a file off spaces from the left-hand side.

It is designed to provide much cleaner syntax and exceptions handling when you are working with code. That explains why it’s good practice to use them with a statement where applicable. This is helpful because using this method any files opened will be closed automatically after one is done, so auto-cleanup.

Example:

# Python code to illustrate with()

with open("file.txt") as file:

data = file.read()

# do something with data

Using write along with with() function

We can also use write function along with with() function:

# Python code to illustrate with() alongwith write()

with open("file.txt", "w") as f:

f.write("Hello World!!!")

split() using file handling

We can also split lines using file handling in Python. This splits the variable when space is encountered. You can also split using any characters as we wish. Here is the code:

# Python code to illustrate split() function

with open("file.text", "r") as file:

data = file.readlines()

for line in data:

word = line.split()

print (word)

**4.2 Data files and it’s fields**

**Users file**

This file is used for storing user data who makes use of the system. It’s fields are:

Username: username of the user

Password: password of the user

User type: type of the user which is either superAdmin or user

**Brands file**

This file is used to store data of various brands. It’s fields are:

Brand id: id of the brand. This is the primary key and it will be unique for each brand

Brand Name: Name of the brand.

**Category file**

This file is used to store data of various categories. It’s fields are:

Category id: id of the category. This is the primary key and it will be unique for each category

Category Name: Name of the category.

**Products file**

This file is used to store data of various products. It’s fields are:

Product id: id of the Product. This is the primary key and it will be unique for each Product

Product Name: Name of the Product.

Brand id: brand id for the particular product

Category id: Category id for the particular product

Stock :stock of the product

Prize: prize of the product per unit

**Sales file**

This file is used to store data of sales on the market. It’s fields are:

sale id: id of the sale. This is the primary key and it will be unique for each sale

date: date of the sale.

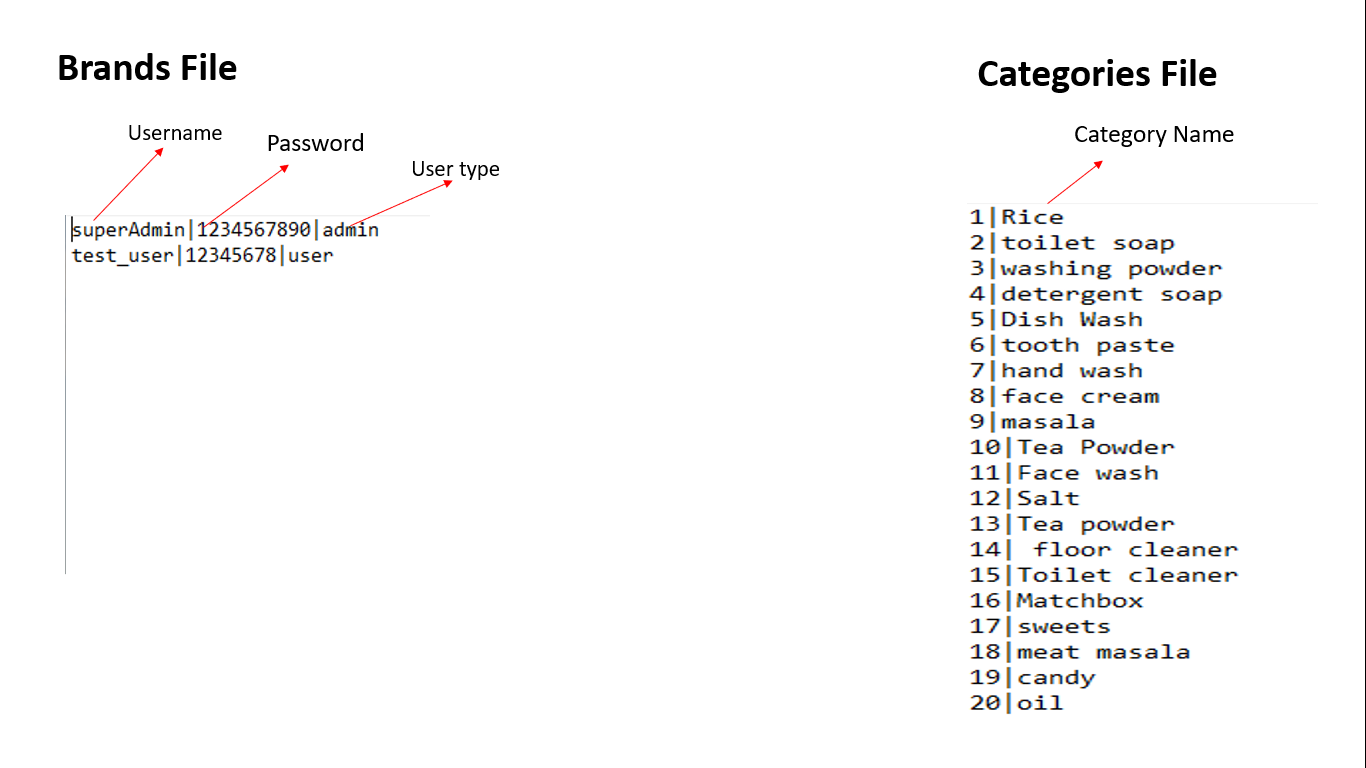
Time: time of the sale.

Amount: amount of the sale

**3.3 Data files**

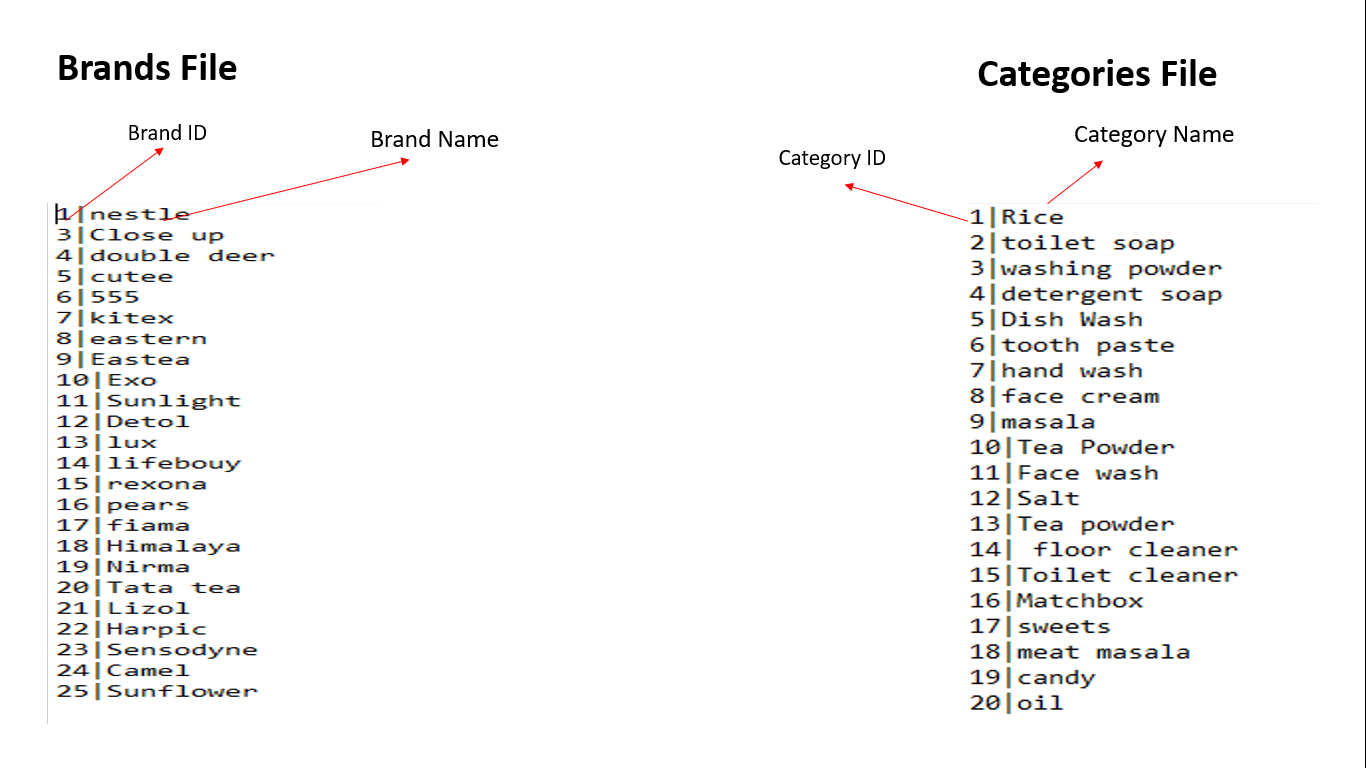
**Users file**

This file is used for storing user data who makes use of the system

****

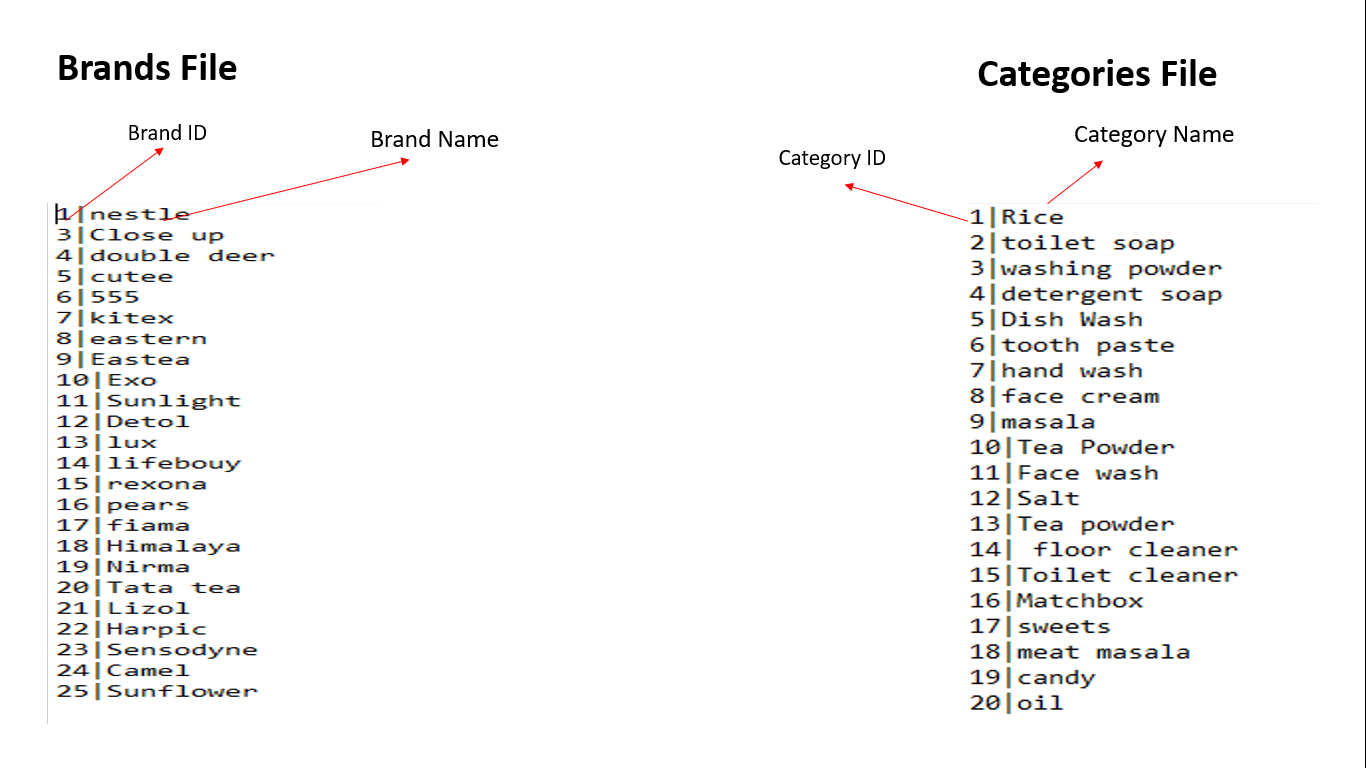
**Brands file**

This file is used to store data of various brands



**Categories file**

This file is used to store data of various categories



**Products file**

This file is used to store data of various products



**Sales file**

This file is used to store data of sales made on the supermarket



**Chapter 5**

**FRONT END DESIGN**

In this chapter we have explained about the front end design tools such as Pycharm along with front end language Python.

**5.1 Pycharm IDE:**

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

**5.2 Qt-Designer:**

*Qt Designer* is a visual tool for designing and building graphical user interfaces (GUIs) from Qt components. It allows you to design and build widgets and dialogs using on-screen forms using the same widgets that will be used in your application. Components created with *Qt Designer* can also take advantage of Qt's signals and slots, and they can be previewed so that you can ensure that they will look and feel exactly as you intended.

**5.3 Python:**

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

**5.5 Pseudo code**

**5.5.1 Login page:**

Step1: Enter the USERNAME in the given field.

Step2: Enter the valid PASSWORD.

Step3: Click on LOGIN button

**5.5.2 Purchase page:**

This page is designed for making purchases from the supermarket.

Step 1: Open the Purchase page

Step 2: Select the product to add to cart.

Step 3: Set the units required for the particular product.

Step 4:Add as many as products and its units to add to cart for buying.

Step 5:Review the products added in the cart table.

Step 6: Click purchase button to make purchase.

Step 7:Cofirm the purchase.

**5.5.3 Products page:**

This page is for creating and updating different products, brands, categories.

Step1: Open the Products page.

Step2:To create a new product, set all its details such as name, brand, category, current units, prize per units and click add button

Step3 :To create a new brand, set brand name and click add button.

Step 4: To create a new category, set category name and click add button.

Step5: To update product name, select product, click update button, set product name and confirm.

Step6: To update brand name, select brand, click update button, set brand name and confirm

Step7: To update category name, select category, click update button, set category name and confirm.

Step 8: To delete category, select category, click delete button, name and confirm.

Step9: To delete brand, select brand, click delete button, name and confirm.

Step10: To delete product, select product, click delete button, name and confirm.

**5.5.4 Stock page:**

In this page the user can review current stocks and update stock details.

Step1: Open the Stock page

Step2:To load stock to the stock table set the filters and click load button.

Step3:To update prize of a product select it from the stock table, click update prize button, set prize and confirm.

Step4:To update stock of a product select it from the stock table, click update stock button, set new stock and confirm.

**5.5.5 Sales page:**

In this page user can see details of sales of a day and review monthly sale.

To see sale of a day

Step1:Select particular date from the calendar.

Step 2: click load button.

Step 3: sales will be displayed in sales table

To review monthly sale

Step 1:Select the month.

Step 2:select the year.

Step 3:click check button.

**5.5.10 Change password section:**

To change password, go to change password in menu bar then follow the steps.

Step1: Enter the new PASSWORD.

Step2: Re-enter the new PASSWORD.

Step3:Click change password.

Password changed.

**5.5.10 Add User section:**

To add new user, go to add user in menu bar then follow the steps.

Step 1: Enter the USERNAME.

Step 2: Enter the PASSWORD.

Step 3: CONFIRM PASSWORD.

Step 4: Click on CREATE USER button.

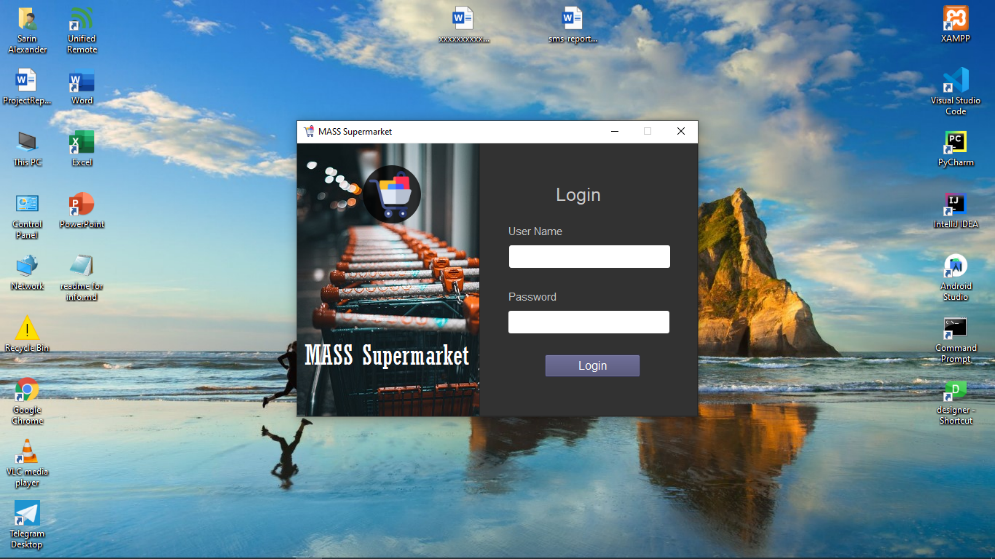
New user is created.

**5.5.11 Logout:**

Once you are done with the work. You can logout using “LOGOUT” button in menu bar.

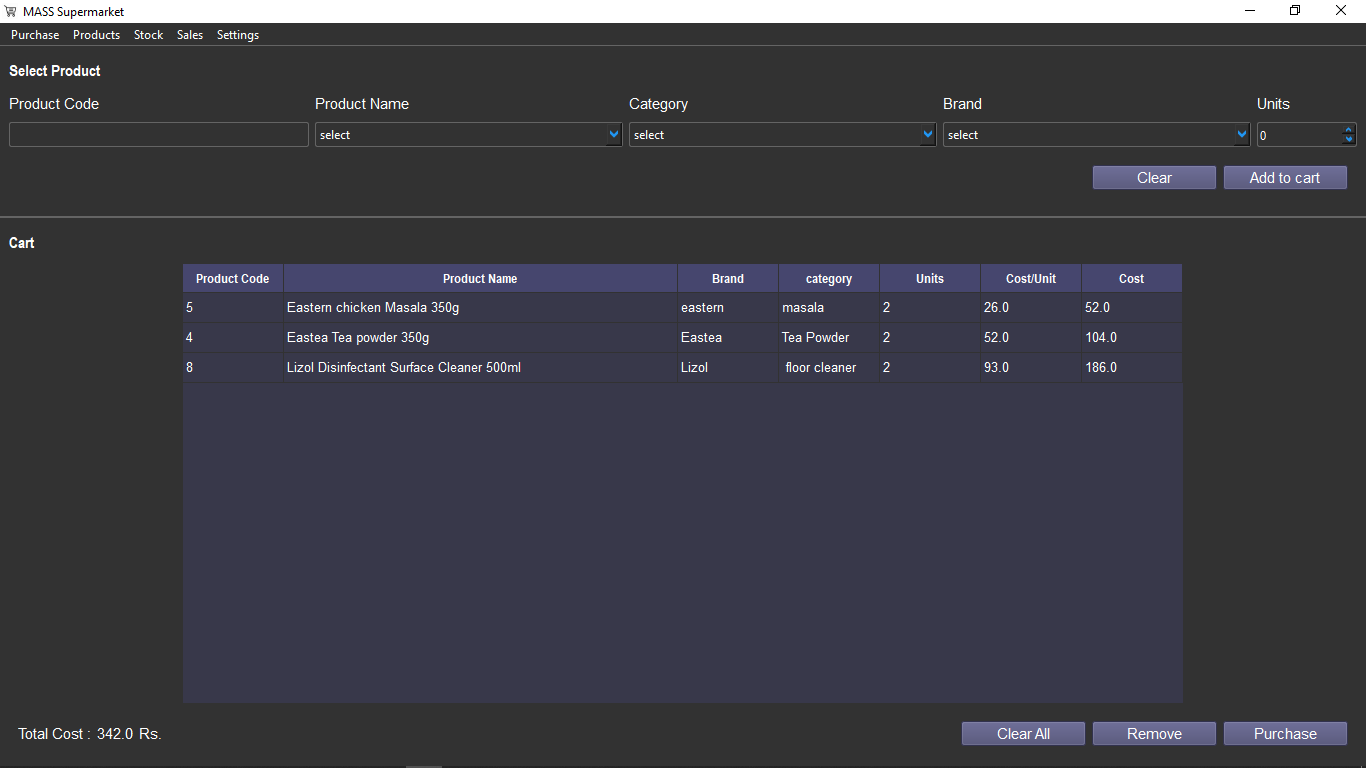
**Chapter 6**

**RESULTS AND ANALYSIS**

****

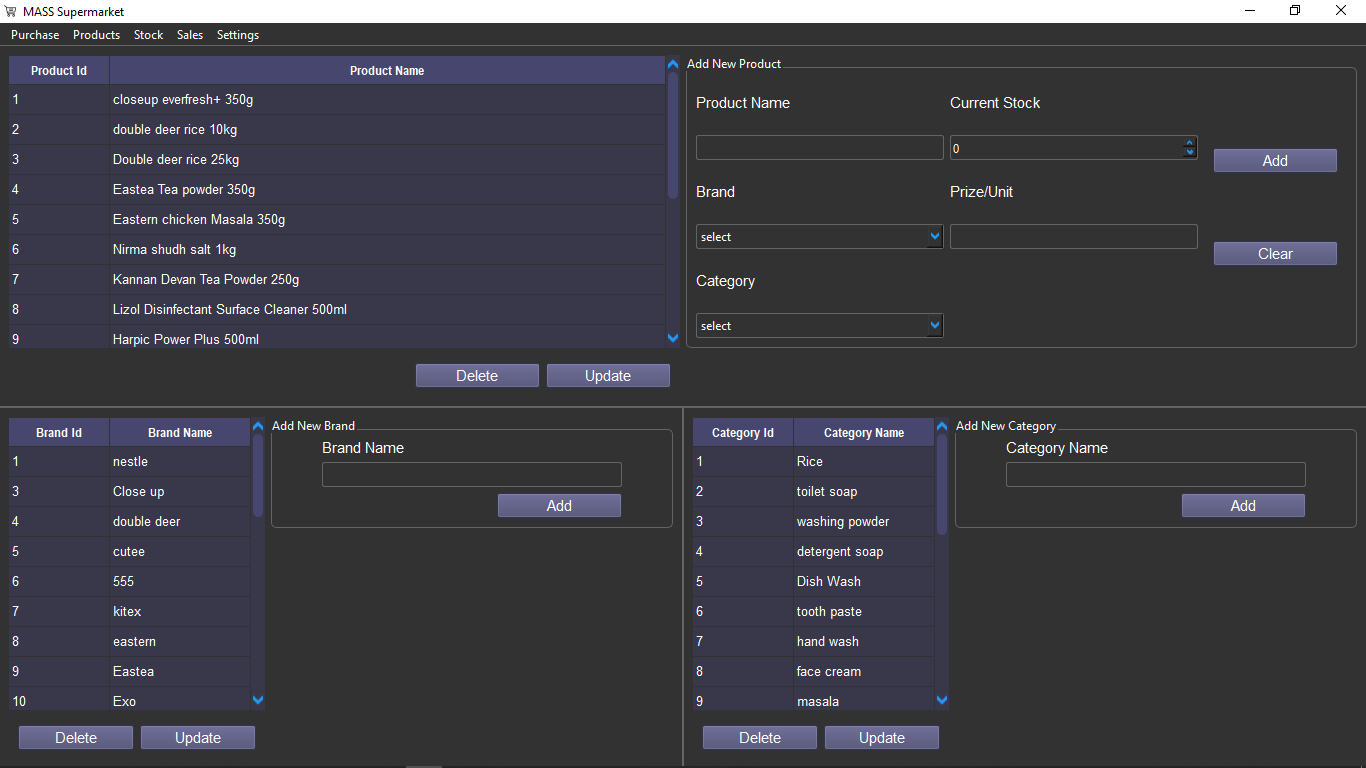
**Fig 6.1: Login Page**

In the above figure shows the login page, if the user wants to login, they must enter valid/registered username and password and click on login button.



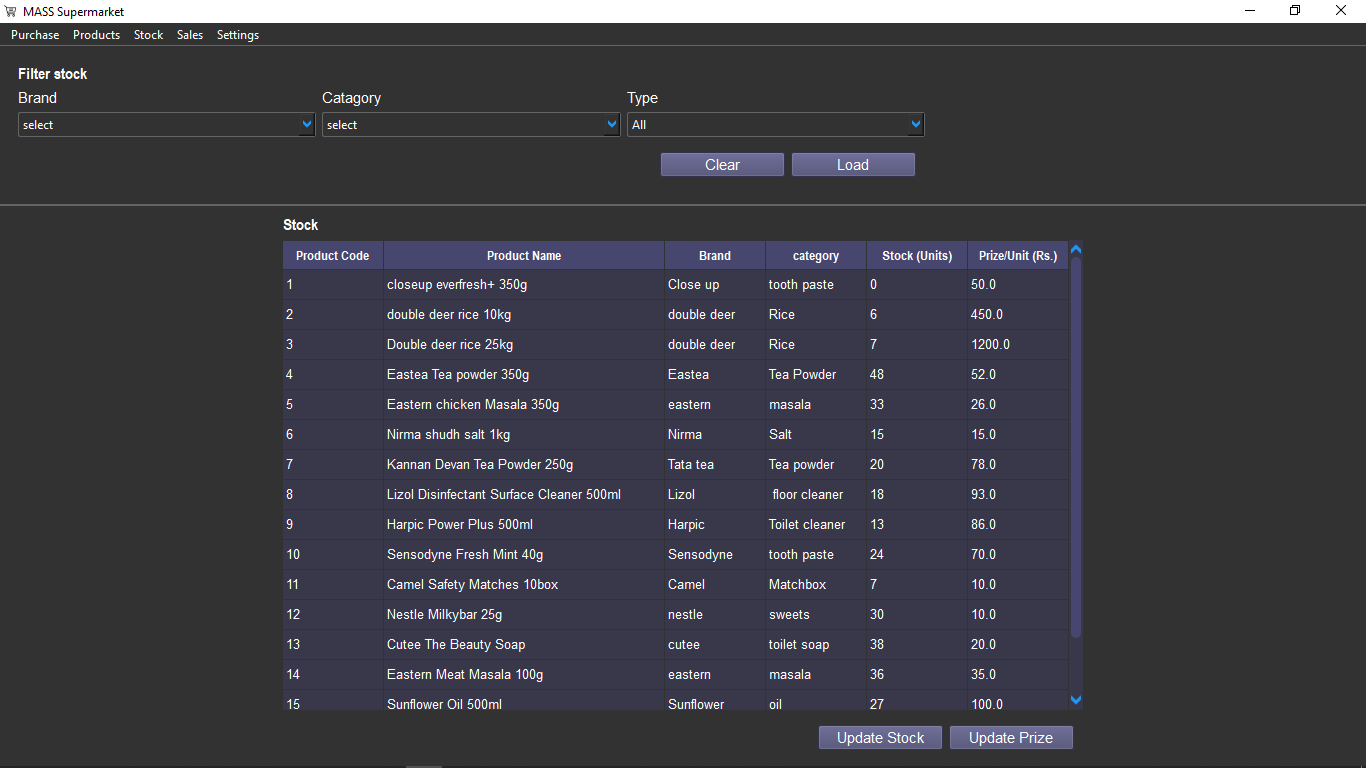
**Fig 6.2:Purchase page**

In the above figure shows the purchase page for making purchase from the supermarket



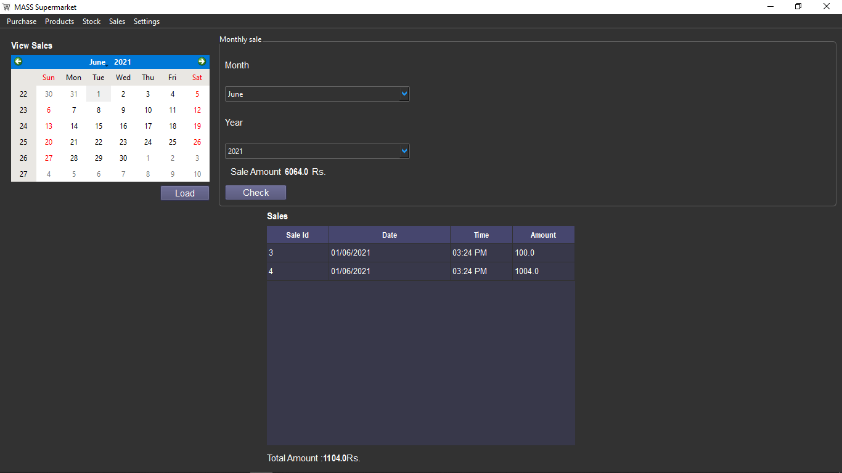
**Fig 6.3: Products page**

In the above figure shows the Add products page, here we can add, update and remove products, brands, categories.



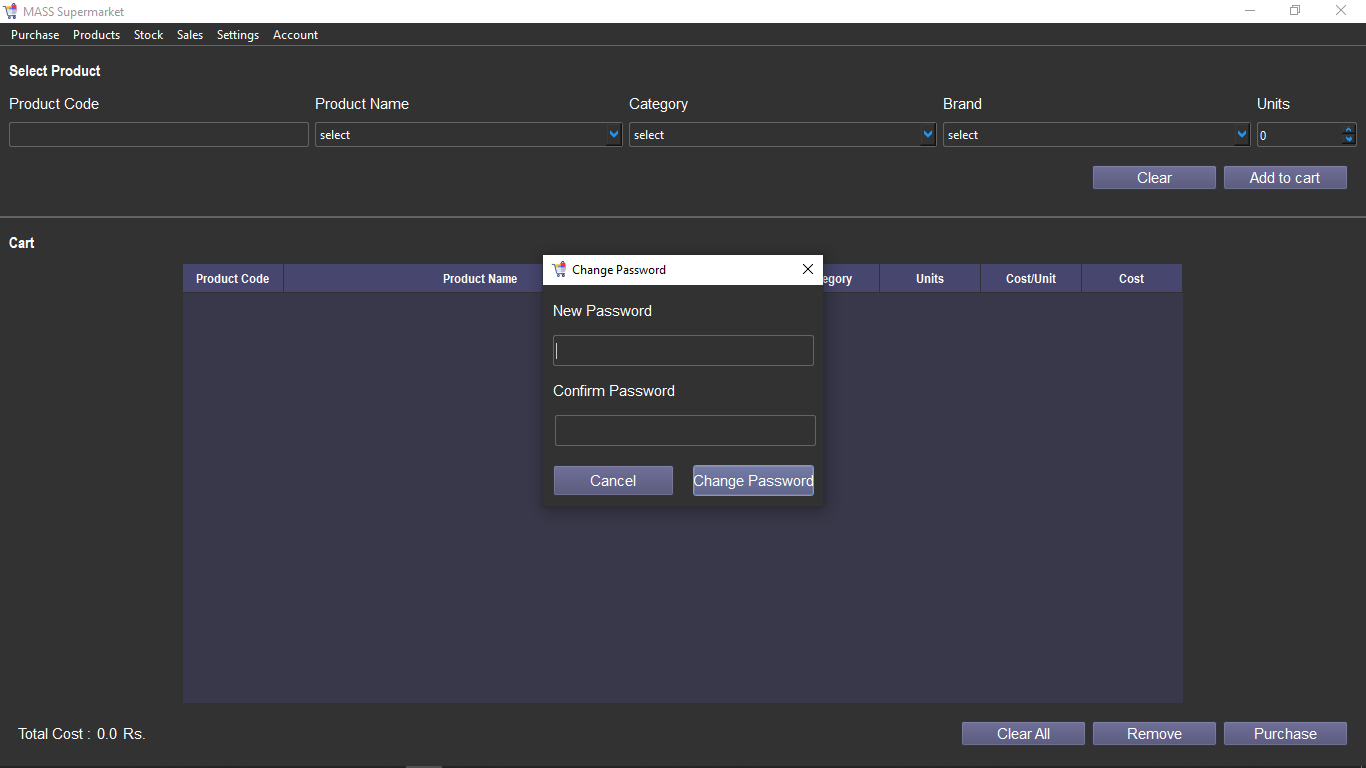
**Fig 6.3.1:Stock page**

In the above figure shows the stock page to manage and review stock details.



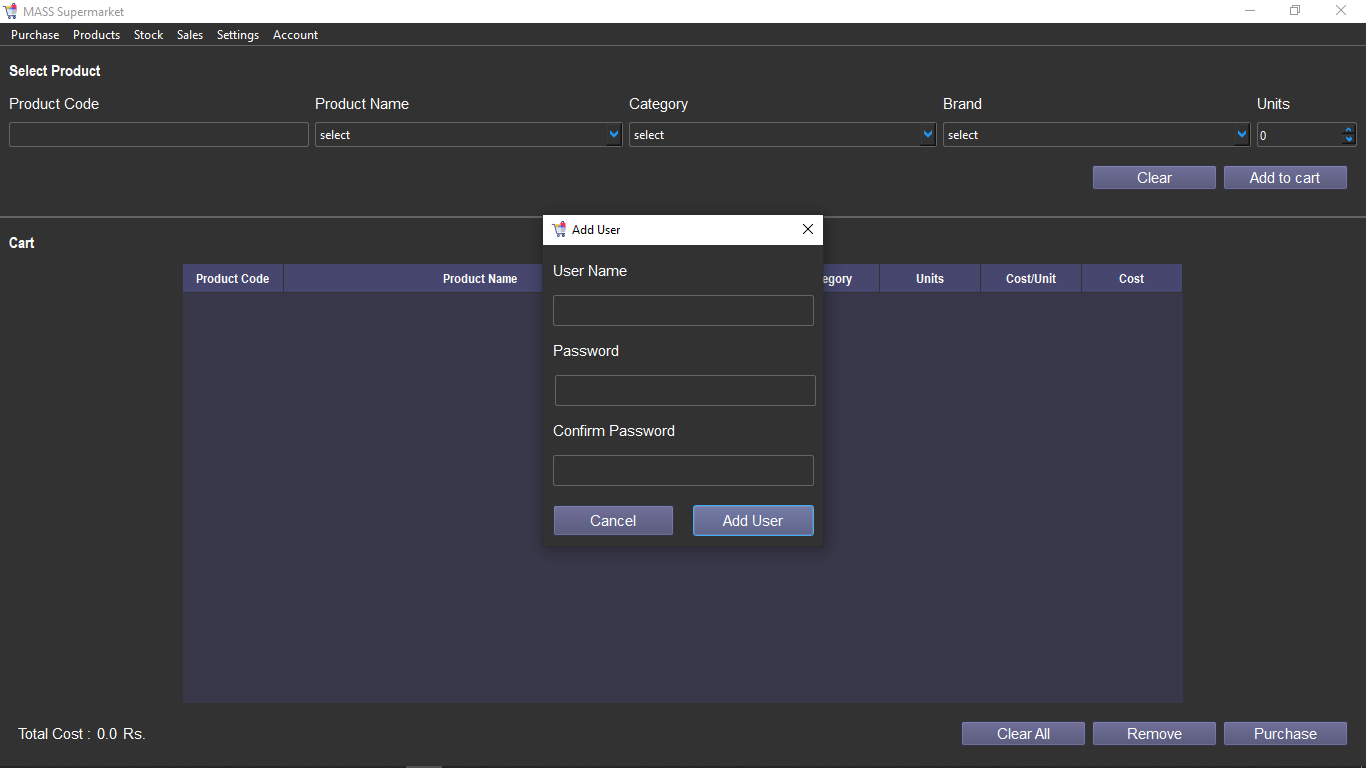
**Fig 6.4:Sales page**

In the figure shows the sales page to review sales and monthly sales.



**Fig 6.5: Change password section**

In the above figure shows the section to change password of the user



**Fig 6.6:Add user section**

In the above figure shows the add user page, this feature is only provided for the super admin of the supermarket. You have to enter the new username, password and confirm the password. Click on create user button. If the super admin password is wrong or password and confirm password don’t match. It will pop invalid password. On successfully creating user it will display “New user created successfully”.

**Chapter 7**

# CONCLUSION

The supermarket management system has realized the transmission and control of large goods, so as to facilitate the management and decision of sales, and reduce a big burden for supermarkets and supermarket managers. It also can help to improve the work efficiency of supermarket.

Its requirements are to provide the basic information maintenance function of employees, memberships and products so that managers can through the function to add, delete, and modify the basic information of employees and the employees can through it to add, modify and delete the basic information of memberships and goods.

The Supermarket Inventory System Management provides a user-friendly, interactive Graphical user interface (GUI). All data is stored in files for persistence. The mini project uses 5 files: An User file, to store the user data, a Brands file, to store the brands data, A Category file, to store the category data, A Products file, to store the products data, A Sales file, to store the sales data.

# REFERENCES

1. Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners
2. [www.stackoverflow.com](http://www.stackoverflow.com/)
3. [www.tutorialpoint.com](http://www.tutorialpoint.com/)
4. [www.wikepedia.com](http://www.wikepedia.com/)
5. [www.softwaretestingfundamentals.com](http://www.softwaretestingfundamentals.com/)