Mini Project

Inventory Tracking System

Submitted in partial fulfilment of the requirements of the degree

SECOND YEAR COMPUTER ENGINEERING

By

Mule Rishi Niranjan - 50

Mulik Shubham Sanjay - 51

Gend Gaurav Kundlik - 24

Padwal Vedant Vasant - 55

Under Guidance of

Prof. J. A. Gaikwad

University of Mumbai

Department of Computer Engineering

DATTA MEGHE COLLEGE OF ENGINEERING,

AIROLI,NAVI MUMBAI -400 708

Abstract

This Inventory Tracker & Management Project is a System Application designed for a particular Department in an Academic Institution.

Tracking things manually and maintaining record in books may lead to manual errors and it increases human efforts.

So, *"The Main Aim of this project is to reduce the stress of tracking by Substituting manpower with Automation."*

This application provides the users to handle brands, categories, orders, products, as well as report of the location of a particular item.

The system merely records restocking data and provides warning of low stock at any location through notification at a specified interval.

The system is used to track items and parts as they are imported to the Department, transferred between Classrooms, and finally if loaned to a person (staff or student).

In future, we can develop android application for this project. So that current status of inventory in stock can be seen on mobile phones anywhere in the world. This leads to easy and effective management of the inventory.

Acknowledgement

We would like to take this opportunity to express our gratitude towards all the people who have in various ways, helped in the successful completion of our project.

We must convey our gratitude to Prof. Jyoti Gaikwad for giving us the constant source of inspiration and help in preparing the project, personally correcting our work, and providing encouragement throughout the project.

We also thank all our faculty members for steering us through the tough as well as easy phases of the project in a result-oriented manner with concern attention.

Thank You.

INDEX

Sr No.	Chapter Name	Page No.
1.	**Abstract**	2
2.	**Acknowledgement**	3
3.	**Introduction** 1. System Introduction 2. Objectives	 5 7
4.	**Problems in Existing System**	8
5.	**Proposed System** 1. Scope of Proposed System 2. Architecture and Framework a. Tkinter b. SQLite3 c. pyMYSQL 3. System Requirements a. Hardware Requirements b. Software Requirements	 9 11 13
6.	**Testing Process Design**	14
7.	**Source Code** 1. Windows Version 2. MacOS Version	 15 25
8.	**Conclusion**	35
9.	**References**	36

Introduction

System Introduction:

For optimal sales and inventory management processes, you need robust functionality for managing your logistics facilities.

Support for inventory management helps you record, and track materials based on both quantity and value. Inventory management functions cover internal warehouse movements and storage.

Using this software, we can reduce costs for warehousing, transportation, order fulfillment, and material handling.

Additional benefits of inventory management include improved cash flow, visibility, and decision making.

This software is user friendly and hence easy to use.

Objectives

- The main objective of this system is to keep records of the complete inventory.
- It supports for inventory management; helps you record and track materials on the basis of both quantity and value.
- It improves the flow, visibility, and decision making.
- For warehouse management, you can track quantity and value of all your materials, perform physical inventory, and optimize your warehouse resources.

Problems In existing system

As we know manual system are quite tedious, time consuming and less efficient and accurate in comparison to the computerized system.

So, following are some disadvantages of the old system:

1. Time consuming
2. Less accurate
3. Less efficient
4. Lot of paperwork
5. Slow data processing
6. Not user-friendly environment
7. Difficult to keep old records

Scope of Proposed System

The scope of this system is to provide user efficient working environment and more output can be generated through this. This system provides user friendly interface resulting in knowing every usability features of the system.

This system helps in tracking records so that past records can be verified through them and one can make decisions based on the past records. This system completes the work in a very less time resulting in less time consumption and high level of efficiency.

This system is developed in such a way that even a naïve user can also operate the system easily. The calculations are made very quickly, and the records are directly saved into databases and the databases can be maintained for a longer period of time. Each record can be retrieved and can be verified for the future transactions.

Also, this system provides high level of security for data leaking.

Architecture and Framework

Tkinter:

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. **Tk(screenName=None, baseName=None, className='Tk', useTk=1):** To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

```
m=tkinter.Tk() where m is the name of the main window object
```

2. **mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

```
m.mainloop()
```

SQLite3:

SQLite3 is a compact free database you can use easily create and use a database.

Though SQLite3 is not a full-featured database, it supports a surprisingly large set of the SQL standard, and is ideal for those just starting to learn SQL as well for developers that need a simple database engine to plug into their applications.

As such, SQLite has become very popular with smart phone developers.SQLite is a software library that provides a relational database management system. The lite in SQLite means lightweight in terms of setup, database administration, and required resources.

SQLite has the following noticeable features:

- self-contained
- serverless
- zero-configuration
- transactional

pyMYSQL:

PyMySQL is a pure-Python MySQL client library, based on PEP 249. Most public APIs are compatible with MySQL client and MySQLdb. PyMySQL works with MySQL 5.5+ and MariaDB 5.5+.

MySQL is a leading open-source database management system PyMySQL is an interface for connecting to a MySQL database server from Python.

It implements the Python Database API v2.0 and contains a pure-Python MySQL client library. The goal of PyMySQL is to be a drop-in replacement for MySQLdb.
The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

Once a database connection is established, we are ready to create tables or records into the database tables using execute method of the created cursor.

Hardware and Software Requirement

HARDWARE REQUIREMENTS:

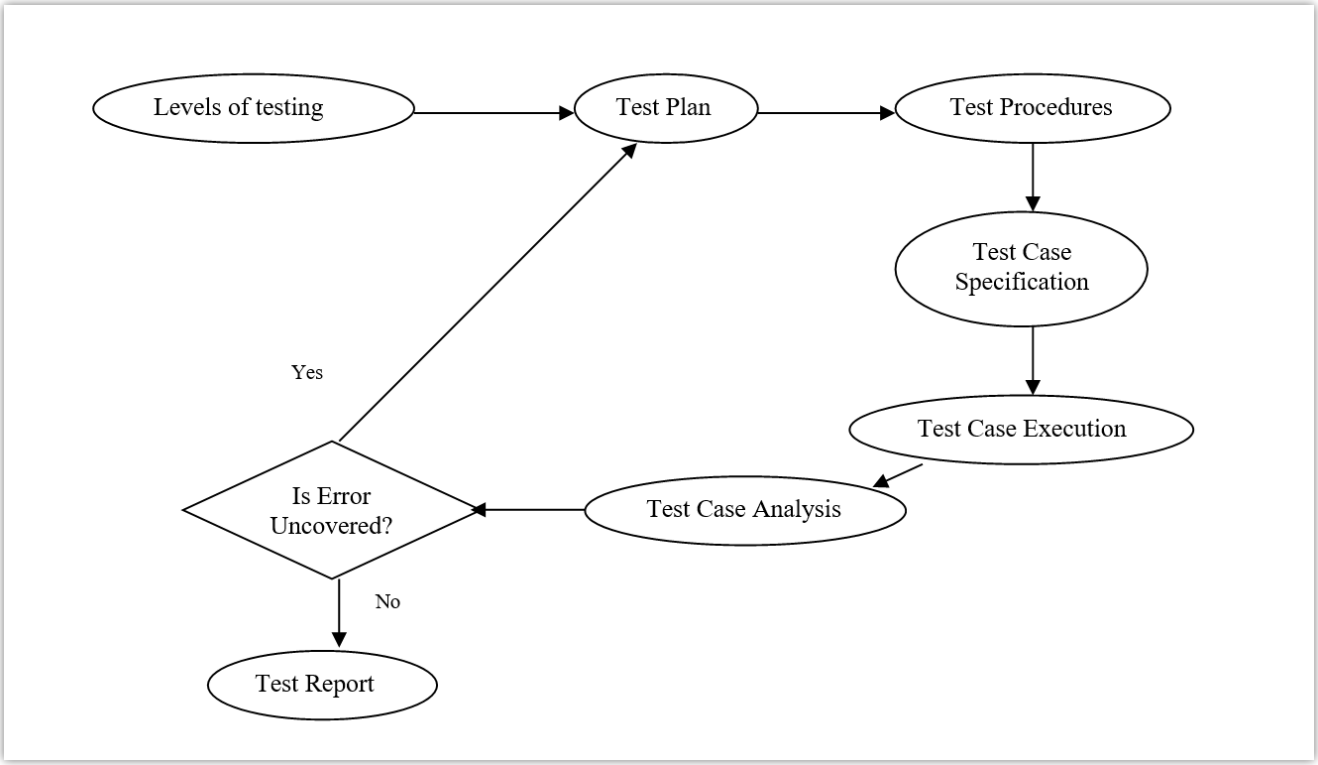
- Processor: Pentium 4 or more for optimum performance
- RAM: Recommended 256MB
- Hard Disk: Minimum 100MB

SOFTWARE REQUIREMENTS:

- Operating System - Certified Distribution of Windows or macOS
- Python 3.4 or above
- Libraries:
  - Tkinter
  - Pymysql
  - Sqlite3

Testing Process Design

The testing process can be shown as:



Source Code

Windows_Version.py:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

__version__ = "0.1.0 for Windows 10"
__author__  ="Rishi Mule, Shubham Mulik, Gaurav Gend, Vedant Padwal"
__license__ = 'MIT'

from tkinterimport *
from tkinterimport ttk
from tkinterimport messagebox
import pymysql
import sqlite3

def create_database():
"""Function to create a Database"""
    conn = sqlite3.connect('inventory.db')
    cur = conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS dmce_inventory (product_id text PRIMARY KEY
, product_type text , model_no text , manufacturer text , department text ,
location text ,incharge text, comment text)")
conn.commit()
conn.close()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------

class Inventory():
"""Creating a main window on Inventory"""

def __init__(self, root):
"""Default __INIT__ Function"""

self.root=root
self.root.title("I.T.S")
self.root.geometry("1200x660+30+0")
self.root.resizable(0, 0)

self.product_id_var = StringVar()
self.product_type_var = StringVar()
self.model_no_var = StringVar()
self.manufacturer_var = StringVar()
self.department_var = StringVar()
self.location_var = StringVar()
self.incharge_var = StringVar()

self.search_by_var = StringVar()
self.search_txt_var = StringVar()
```

```
head_title = Label(self.root,text="Inventory Management System",bd=10,
relief=GROOVE, font=("ariel", 20 , "bold"), bg="RED", fg="white")
head_title.pack(side="top", pady=20, padx=10, fill=X)

#=============================================================================================
=================================================
#================================MANAGE_FRAME=====================================
=================================================
#=============================================================================================
=================================================

Manage_Frame=Frame(self.root, bd=5, relief=RIDGE, bg="crimson")
Manage_Frame.place(x=10, y=80, width=350, height=570)

m_title=Label(Manage_Frame,text="Manage Inventory", font=("", 20 , "bold"),
bg="crimson", fg="white")
m_title.grid(row=0, columnspan=2, pady=20)

#---------------------------------------------------------------------------------------------
-------------------------------------------------

def caps(event):
"""Function to Convert Text To UPPERCAP"""
self.product_id_var.set(self.product_id_var.get().upper())
self.product_type_var.set(self.product_type_var.get().upper())
self.model_no_var.set(self.model_no_var.get().upper())
self.manufacturer_var.set(self.manufacturer_var.get().upper())
self.location_var.set(self.location_var.get().upper())
self.incharge_var.set(self.incharge_var.get().upper())
self.search_txt_var.set(self.search_txt_var.get().upper())

#---------------------------------------------------------------------------------------------
-------------------------------------------------

lbl_product_id=Label(Manage_Frame,text="Product ID : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_product_id.grid(row=1, column=0, padx=10, pady=10,sticky ="w")

txt_product_id=Entry(Manage_Frame, font=("times new roman", 13 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.product_id_var)
txt_product_id.bind("<KeyRelease>", caps)
txt_product_id.grid(row=1, column=1, padx=10, pady=10, sticky ="w")

#---------------------------------------------------------------------------------------------
-------------------------------------------------

lbl_type=Label(Manage_Frame,text="Product Type : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_type.grid(row=2, column=0, padx=10, pady=10,sticky ="w")

txt_type=Entry(Manage_Frame, font=("times new roman", 13 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.product_type_var)
txt_type.bind("<KeyRelease>", caps)
txt_type.grid(row=2, column=1, padx=10, pady=10, sticky ="w")

#---------------------------------------------------------------------------------------------
-------------------------------------------------

lbl_model_no=Label(Manage_Frame,text="Model No : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
```

```
lbl_model_no.grid(row=3, column=0, padx=10, pady=10,sticky ="w")

txt_model_id=Entry(Manage_Frame, font=("times new roman", 13 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.model_no_var)
txt_model_id.bind("<KeyRelease>", caps)
txt_model_id.grid(row=3, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_manufacturer=Label(Manage_Frame,text="Manufacturer : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_manufacturer.grid(row=4, column=0, padx=10, pady=10,sticky ="w")

txt_manufacturer=Entry(Manage_Frame, font=("times new roman", 13 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.manufacturer_var)
txt_manufacturer.bind("<KeyRelease>", caps)
txt_manufacturer.grid(row=4, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_department=Label(Manage_Frame,text="Department : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_department.grid(row=5, column=0, padx=10, pady=10,sticky ="w")

combo_department=ttk.Combobox(Manage_Frame, width=18, font=("", 13, "bold" ),
state="readonly" , textvariable=self.department_var)

combo_department["values"]=("COMPUTER","ELECTRICAL","CIVIL","MECHANICAL","CHEMICAL"
,"I.T.")
combo_department.current(0)
combo_department.grid(row=5, column=1, padx=10, pady=10,sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_location=Label(Manage_Frame,text="Location : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_location.grid(row=6, column=0, padx=10, pady=10,sticky ="w")

txt_location=Entry(Manage_Frame, font=("times new roman", 13 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.location_var)
txt_location.bind("<KeyRelease>", caps)
txt_location.grid(row=6, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_incharge=Label(Manage_Frame,text="Incharge : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_incharge.grid(row=7, column=0, padx=10, pady=10,sticky ="w")

txt_incharge=Entry(Manage_Frame, font=("times new roman", 13 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.incharge_var)
txt_incharge.bind("<KeyRelease>", caps)
txt_incharge.grid(row=7, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------
```

```
lbl_comment=Label(Manage_Frame,text="Comment : ", font=("", 10 , "bold"),
bg="crimson", fg="white")
lbl_comment.grid(row=8, column=0, padx=10, pady=10,sticky ="w")

self.txt_comment=Text(Manage_Frame, width=20, height=3, bd=2, relief=GROOVE,
font=("times new roman", 13 , ""))
self.txt_comment.grid(row=8, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------




#===============================================================================
==============================================
#=========================BUTTON_FRAME=========================================
==============================================
#===============================================================================
==============================================

Button_Frame=Frame(Manage_Frame, bd=4, relief=RIDGE, bg="yellow")
Button_Frame.place(x=5, y=500, width=330, height=50)

#-------------------------------------------------------------------------------
-----------------------------------------------
add_button=Button(Button_Frame, text="Add", width=8, highlightbackground="yellow",
command=self.add_items)
add_button.grid(row=0, column=0, padx=5, pady=7)

#-------------------------------------------------------------------------------
-----------------------------------------------
update_button=Button(Button_Frame, text="Update", width=8,
highlightbackground="yellow", command=self.update_data)
update_button.grid(row=0, column=1, padx=5, pady=7)

#-------------------------------------------------------------------------------
-----------------------------------------------
delete_button=Button(Button_Frame, text="Delete", width=8,
highlightbackground="yellow", command=self.delete_data)
delete_button.grid(row=0, column=2, padx=5, pady=7)

#-------------------------------------------------------------------------------
-----------------------------------------------
clear_button=Button(Button_Frame, text="Clear", width=10,
highlightbackground="yellow", command=self.clear)
clear_button.grid(row=0, column=3, padx=5, pady=7)



#===============================================================================
==============================================
#=========================DETAIL_FRAME=========================================
==============================================
#===============================================================================
==============================================

Detail_Frame=Frame(self.root, bd=4, relief=RIDGE, bg="crimson")
Detail_Frame.place(x=370, y=80, width=820, height=570)
```

```
#============================================================================================
================================================
#=========================SEARCH_FRAME=============================================
================================================
#============================================================================================
================================================

Search_Frame=Frame(Detail_Frame, bd=4, relief=RIDGE, bg="yellow")
Search_Frame.place(x=10, y=10, width=792, height=60)

lbl_search=Label(Search_Frame,text="Search By : ", font=("", 13 , "bold"),
bg="yellow", fg="red")
lbl_search.grid(row=0, column=0, padx=10, pady=10,sticky ="w")

combo_search_by=ttk.Combobox(Search_Frame, width=13, font=("", 13, "" ),
state="readonly", textvariable=self.search_by_var)
combo_search_by["values"]=("All","ProductID.","ProductType","Model
No","Manufacturer","Department","Location","Incharge")
combo_search_by.current(0)
combo_search_by.grid(row=0, column=1, padx=2, pady=10,sticky ="w")



txt_search=Entry(Search_Frame, width=30,font=("times new roman", 15 ) ,bd=2,
relief=GROOVE, textvariable=self.search_txt_var)
txt_search.bind("<KeyRelease>", caps)
txt_search.grid(row=0, column=2, padx=20, pady=10, sticky ="w")

search_button=Button(Search_Frame, text="Search", width=8,
highlightbackground="yellow", command=self.search_data)
search_button.grid(row=0, column=3, padx=4, pady=5)

view_button=Button(Search_Frame, text="View All", width=8,
highlightbackground="yellow", command=self.view_data)
view_button.grid(row=0, column=4, padx=9, pady=5)

#============================================================================================
================================================
#=========================TABLE_FRAME=============================================
================================================
#============================================================================================
================================================

Table_Frame=Frame(Detail_Frame, bd=4, relief=RIDGE, bg="yellow")
Table_Frame.place(x=10, y=80, width=792, height=472)

scroll_x=Scrollbar(Table_Frame, orient=HORIZONTAL)
scroll_y=Scrollbar(Table_Frame, orient=VERTICAL)
self.View_Table=ttk.Treeview(Table_Frame,
columns=("pid","ptype","mno","manufacturer","department","location","incharge","com
ment"), xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
scroll_x.pack(side=BOTTOM, fill=X)
scroll_y.pack(side=RIGHT, fill=Y)
scroll_x.config(command=self.View_Table.xview)
scroll_y.config(command=self.View_Table.yview)

self.View_Table.heading("pid", text="Product ID.")
self.View_Table.heading("ptype", text="Product Type")
self.View_Table.heading("mno", text="Model No")
self.View_Table.heading("manufacturer", text="Manufacturer")
self.View_Table.heading("department", text="Department")
```

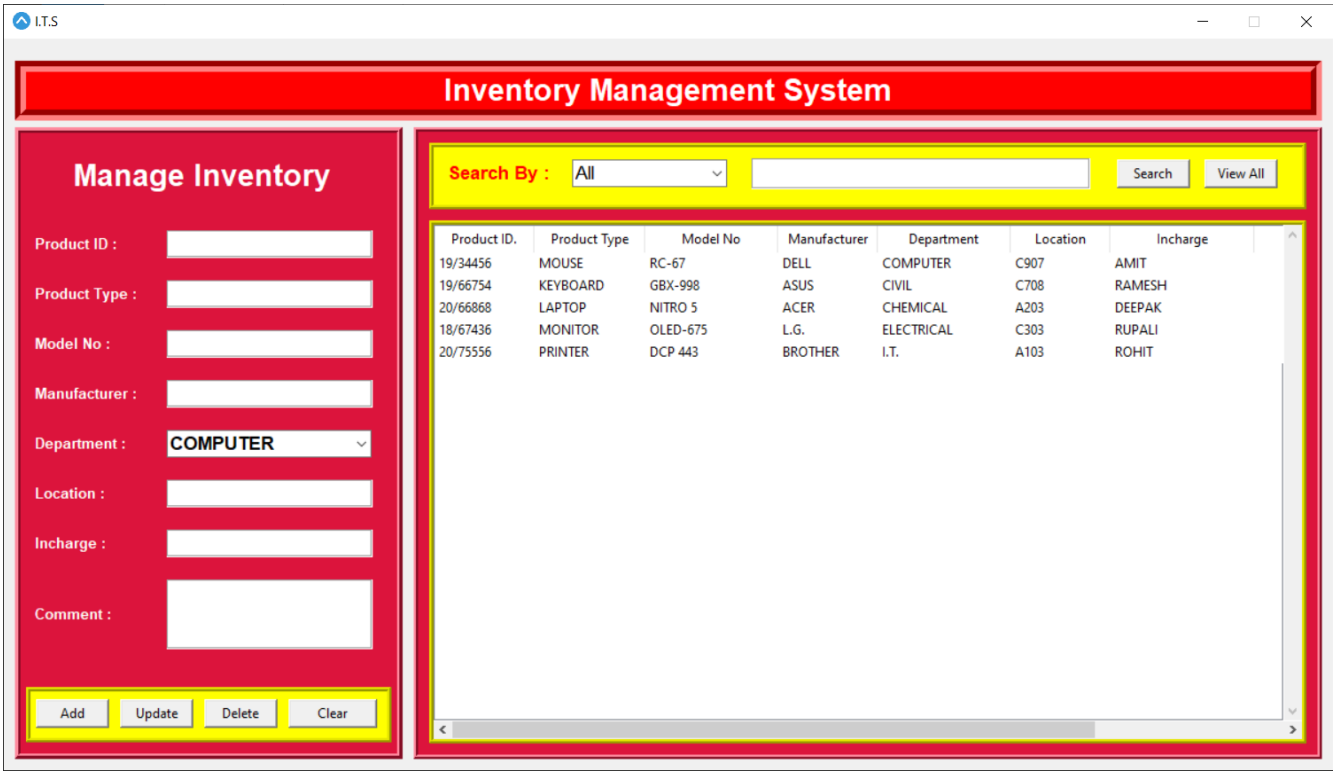
```
self.View_Table.heading("location", text="Location")
self.View_Table.heading("incharge", text="Incharge")
self.View_Table.heading("comment", text="Comment")

self.View_Table.column("pid", width=90)
self.View_Table.column("ptype", width=100)
self.View_Table.column("mno", width=120)
self.View_Table.column("manufacturer", width=90)
self.View_Table.column("department", width=120)
self.View_Table.column("location", width=90)
self.View_Table.column("incharge", width=130)
self.View_Table.column("comment", width=250)


self.View_Table["show"]="headings"
self.View_Table.pack(fill=BOTH, expand=1)
self.View_Table.bind("<ButtonRelease-1>", self.get_cursor)


self.view_data()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------

def add_items(self):
"""Function to ADD item to Database"""

if self.product_id_var.get()=="":
messagebox.showerror("Error","Product ID. cannot be blank!!!")

else:
try:
                con=sqlite3.connect('inventory.db')
                cur=con.cursor()
cur.execute(" insert into dmce_inventory values (?,?,?,?,?,?,?,?)",(
self.product_id_var.get(),
self.product_type_var.get(),
self.model_no_var.get(),
self.manufacturer_var.get(),
self.department_var.get(),
self.location_var.get(),
self.incharge_var.get(),
self.txt_comment.get('1.0',END),

))


con.commit()
self.view_data()
con.close()

except:
pass

else:
self.clear()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------
```

```
def view_data(self):
"""Function to VIEW data into Table"""

        con=sqlite3.connect('inventory.db')
        cur=con.cursor()
cur.execute("select * from dmce_inventory")
        rows=cur.fetchall()
self.View_Table.delete(*self.View_Table.get_children())
if len(rows)!=0:
for row in rows:
self.View_Table.insert("", END, values=row)
con.commit()
con.close()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------

def clear(self):
"""Function to CLEAR all Input Fields"""

self.product_id_var.set("")
self.product_type_var.set("")
self.model_no_var.set("")
self.manufacturer_var.set("")
self.location_var.set("")
self.incharge_var.set("")
self.txt_comment.delete("1.0", END)

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------

def get_cursor(self,event):
"""Function to SELECT a particular item"""

try:
cursor_row=self.View_Table.focus()
            contents=self.View_Table.item(cursor_row)
            row=contents["values"]

self.product_id_var.set(row[0])
self.product_type_var.set(row[1])
self.model_no_var.set(row[2])
self.manufacturer_var.set(row[3])
self.department_var.set(row[4])
self.location_var.set(row[5])
self.incharge_var.set(row[6])
self.txt_comment.delete("1.0", END)
self.txt_comment.insert(END, row[7])

except:
pass

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------

def update_data(self):
"""Function to UPDATE an item of Database"""
```

```
        con=sqlite3.connect('inventory.db')
        cur=con.cursor()
cur.execute("update dmce_inventory set  product_type=? , model_no=? ,
manufacturer=? , department=? , location=? ,incharge=?, comment=? where
product_id=?",(
self.product_type_var.get(),
self.model_no_var.get(),
self.manufacturer_var.get(),
self.department_var.get(),
self.location_var.get(),
self.incharge_var.get(),
self.txt_comment.get('1.0',END),
self.product_id_var.get()

))
con.commit()
self.view_data()
self.clear()
con.close()

#-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------------------------------------------

def delete_data(self):
"""Function to DELETE an item from the Database"""

        con=sqlite3.connect('inventory.db')
        cur=con.cursor()
cur.execute("delete from dmce_inventory where
product_id=?",(self.product_id_var.get(),))
con.commit()
self.view_data()
self.clear()
con.close()

#-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------------------------------------------

def search_data(self):
"""Function to Search for items in Database"""

        con=sqlite3.connect('inventory.db')
        cur=con.cursor()

if self.search_by_var.get()=="Product ID.":
cur.execute("select * from dmce_inventory where product_id=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Product Type":
cur.execute("select * from dmce_inventory where product_type=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Model No":
cur.execute("select * from dmce_inventory where model_no=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()
```

```
elifself.search_by_var.get()=="Manufacturer":
cur.execute("select * from dmce_inventory where manufacturer=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Department":
cur.execute("select * from dmce_inventory where department=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Location":
cur.execute("select * from dmce_inventory where location=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Incharge":
cur.execute("select * from dmce_inventory where incharge=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

else:
cur.execute("select * from dmce_inventory where product_id=? OR product_type=? OR
model_no=? OR manufacturer=? OR department=? OR location=? OR incharge=?", (
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),

))
            rows=cur.fetchall()


self.View_Table.delete(*self.View_Table.get_children())
if len(rows)!=0:
for row in rows:
self.View_Table.insert("", END, values=row)
con.commit()
con.close()

#---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
-------------------------------------------------


if __name__ == '__main__':
"""START THE PROGRAM"""

create_database()
    root = Tk()
root.title("Inventory Tracking System")
root.iconbitmap("its_icon.ico")
ob = Inventory(root)
root.mainloop()
```

Output


I.T.S
Inventory Management System
Manage Inventory
Product ID :
Product Type :
Model No :
Manufacturer :
Department :
COMPUTER
Location :
Incharge :
Comment :
Add
Update
Delete
Clear
Search By :
All
Search
View All
Product ID.
Product Type
Model No
Manufacturer
Department
Location
Incharge
19/34456 MOUSE RC-67 DELL COMPUTER C907 AMIT
19/66754 KEYBOARD GBX-998 ASUS CIVIL C708 RAMESH
20/66868 LAPTOP NITRO 5 ACER CHEMICAL A203 DEEPAK
18/67436 MONITOR OLED-675 L.G. ELECTRICAL C303 RUPALI
20/75556 PRINTER DCP 443 BROTHER I.T. A103 ROHIT

MacOS_Version.py:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

__version__ = "0.1.0 for MacOS"
__author__  ="Rishi Mule, Shubham Mulik, Gaurav Gend, Vedant Padwal"
__license__ = 'MIT'

from tkinterimport *
from tkinterimport ttk
from tkinterimport messagebox
import pymysql
import sqlite3

def create_database():
"""Function to create a Database"""
    conn = sqlite3.connect('inventory.db')
    cur = conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS dmce_inventory (product_id text PRIMARY KEY
, product_type text , model_no text , manufacturer text , department text ,
location text ,incharge text, comment text)")
conn.commit()
conn.close()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------

class Inventory():
"""Creating a main window on Inventory"""

def __init__(self, root):
"""Default __INIT__ Function"""

self.root=root
self.root.title("I.T.S")
self.root.geometry("1200x660+30+0")
self.root.resizable(0, 0)

self.product_id_var = StringVar()
self.product_type_var = StringVar()
self.model_no_var = StringVar()
self.manufacturer_var = StringVar()
self.department_var = StringVar()
self.location_var = StringVar()
self.incharge_var = StringVar()

self.search_by_var = StringVar()
self.search_txt_var = StringVar()



head_title = Label(self.root,text="Inventory Management System",bd=10,
relief=GROOVE, font=("ariel", 30 , "bold"), bg="RED", fg="white")
head_title.pack(side="top", pady=10, padx=10, fill=X)
```

```
#=============================================================================
==================================================
#==============================MANAGE_FRAME=======================================
==================================================
#=============================================================================
==================================================

Manage_Frame=Frame(self.root, bd=5, relief=RIDGE, bg="crimson")
Manage_Frame.place(x=10, y=80, width=350, height=570)

m_title=Label(Manage_Frame,text="Manage Inventory", font=("", 20 , "bold"),
bg="crimson", fg="white")
m_title.grid(row=0, columnspan=2, pady=20)

#-----------------------------------------------------------------------------
-------------------------------------------------

def caps(event):
"""Function to Convert Text To UPPERCAP"""
self.product_id_var.set(self.product_id_var.get().upper())
self.product_type_var.set(self.product_type_var.get().upper())
self.model_no_var.set(self.model_no_var.get().upper())
self.manufacturer_var.set(self.manufacturer_var.get().upper())
self.location_var.set(self.location_var.get().upper())
self.incharge_var.set(self.incharge_var.get().upper())
self.search_txt_var.set(self.search_txt_var.get().upper())

#-----------------------------------------------------------------------------
-------------------------------------------------

lbl_product_id=Label(Manage_Frame,text="Product ID : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_product_id.grid(row=1, column=0, padx=10, pady=10,sticky ="w")

txt_product_id=Entry(Manage_Frame, font=("times new roman", 15 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.product_id_var)
txt_product_id.bind("<KeyRelease>", caps)
txt_product_id.grid(row=1, column=1, padx=10, pady=10, sticky ="w")

#-----------------------------------------------------------------------------
-------------------------------------------------

lbl_type=Label(Manage_Frame,text="Product Type : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_type.grid(row=2, column=0, padx=10, pady=10,sticky ="w")

txt_type=Entry(Manage_Frame, font=("times new roman", 15 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.product_type_var)
txt_type.bind("<KeyRelease>", caps)
txt_type.grid(row=2, column=1, padx=10, pady=10, sticky ="w")

#-----------------------------------------------------------------------------
-------------------------------------------------

lbl_model_no=Label(Manage_Frame,text="Model No : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_model_no.grid(row=3, column=0, padx=10, pady=10,sticky ="w")

txt_model_id=Entry(Manage_Frame, font=("times new roman", 15 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.model_no_var)
```

```
txt_model_id.bind("<KeyRelease>", caps)
txt_model_id.grid(row=3, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_manufacturer=Label(Manage_Frame,text="Manufacturer : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_manufacturer.grid(row=4, column=0, padx=10, pady=10,sticky ="w")

txt_manufacturer=Entry(Manage_Frame, font=("times new roman", 15 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.manufacturer_var)
txt_manufacturer.bind("<KeyRelease>", caps)
txt_manufacturer.grid(row=4, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_department=Label(Manage_Frame,text="Department : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_department.grid(row=5, column=0, padx=10, pady=10,sticky ="w")

combo_department=ttk.Combobox(Manage_Frame, width=15, font=("", 15, "" ),
state="readonly", textvariable=self.department_var)

combo_department["values"]=("COMPUTER","ELECTRICAL","CIVIL","MECHANICAL","CHEMICAL"
,"I.T.")
combo_department.current(0)
combo_department.grid(row=5, column=1, padx=10, pady=10,sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_location=Label(Manage_Frame,text="Location : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_location.grid(row=6, column=0, padx=10, pady=10,sticky ="w")

txt_location=Entry(Manage_Frame, font=("times new roman", 15 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.location_var)
txt_location.bind("<KeyRelease>", caps)
txt_location.grid(row=6, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_incharge=Label(Manage_Frame,text="Incharge : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_incharge.grid(row=7, column=0, padx=10, pady=10,sticky ="w")

txt_incharge=Entry(Manage_Frame, font=("times new roman", 15 , "bold") ,bd=2,
relief=GROOVE, textvariable=self.incharge_var)
txt_incharge.bind("<KeyRelease>", caps)
txt_incharge.grid(row=7, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------

lbl_comment=Label(Manage_Frame,text="Comment : ", font=("", 15 , "bold"),
bg="crimson", fg="white")
lbl_comment.grid(row=8, column=0, padx=10, pady=10,sticky ="w")
```

```
self.txt_comment=Text(Manage_Frame, width=20, height=3, bd=2, relief=GROOVE,
font=("times new roman", 15 , ""))
self.txt_comment.grid(row=8, column=1, padx=10, pady=10, sticky ="w")

#-------------------------------------------------------------------------------
-----------------------------------------------




#===============================================================================
==============================================
#=========================BUTTON_FRAME==========================================
==============================================
#===============================================================================
==============================================

Button_Frame=Frame(Manage_Frame, bd=4, relief=RIDGE, bg="yellow")
Button_Frame.place(x=5, y=500, width=330, height=50)

#-------------------------------------------------------------------------------
-----------------------------------------------
add_button=Button(Button_Frame, text="Add", width=5, highlightbackground="yellow",
command=self.add_items)
add_button.grid(row=0, column=0, padx=0, pady=7)

#-------------------------------------------------------------------------------
-----------------------------------------------
update_button=Button(Button_Frame, text="Update", width=5,
highlightbackground="yellow", command=self.update_data)
update_button.grid(row=0, column=1, padx=0, pady=7)

#-------------------------------------------------------------------------------
-----------------------------------------------
delete_button=Button(Button_Frame, text="Delete", width=5,
highlightbackground="yellow", command=self.delete_data)
delete_button.grid(row=0, column=2, padx=0, pady=7)

#-------------------------------------------------------------------------------
-----------------------------------------------
clear_button=Button(Button_Frame, text="Clear", width=5,
highlightbackground="yellow", command=self.clear)
clear_button.grid(row=0, column=3, padx=0, pady=7)



#===============================================================================
==============================================
#=========================DETAIL_FRAME==========================================
==============================================
#===============================================================================
==============================================

Detail_Frame=Frame(self.root, bd=4, relief=RIDGE, bg="crimson")
Detail_Frame.place(x=370, y=80, width=820, height=570)


#===============================================================================
==============================================
#=========================SEARCH_FRAME==========================================
==============================================
```

```
#============================================================================================
================================================

Search_Frame=Frame(Detail_Frame, bd=4, relief=RIDGE, bg="yellow")
Search_Frame.place(x=10, y=10, width=792, height=60)

lbl_search=Label(Search_Frame,text="Search By : ", font=("", 15 , "bold"),
bg="yellow", fg="red")
lbl_search.grid(row=0, column=0, padx=10, pady=10,sticky ="w")

combo_search_by=ttk.Combobox(Search_Frame, width=10, font=("", 15, "" ),
state="readonly", textvariable=self.search_by_var)
combo_search_by["values"]=("ALL","ProductID.","ProductType","Model
No","Manufacturer","Department","Location","Incharge")
combo_search_by.current(0)
combo_search_by.grid(row=0, column=1, padx=2, pady=10,sticky ="w")



txt_search=Entry(Search_Frame, width=35,font=("times new roman", 15 ) ,bd=2,
relief=GROOVE, textvariable=self.search_txt_var)
txt_search.bind("<KeyRelease>", caps)
txt_search.grid(row=0, column=2, padx=20, pady=10, sticky ="w")

search_button=Button(Search_Frame, text="Search", width=5,
highlightbackground="yellow", command=self.search_data)
search_button.grid(row=0, column=3, padx=5, pady=5)

view_button=Button(Search_Frame, text="View All", width=8,
highlightbackground="yellow", command=self.view_data)
view_button.grid(row=0, column=4, padx=6, pady=5)

#============================================================================================
================================================
#=========================TABLE_FRAME=========================================
================================================
#============================================================================================
================================================

Table_Frame=Frame(Detail_Frame, bd=4, relief=RIDGE, bg="yellow")
Table_Frame.place(x=10, y=80, width=792, height=472)

scroll_x=Scrollbar(Table_Frame, orient=HORIZONTAL)
scroll_y=Scrollbar(Table_Frame, orient=VERTICAL)
self.View_Table=ttk.Treeview(Table_Frame,
columns=("pid","ptype","mno","manufacturer","department","location","incharge","com
ment"), xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
scroll_x.pack(side=BOTTOM, fill=X)
scroll_y.pack(side=RIGHT, fill=Y)
scroll_x.config(command=self.View_Table.xview)
scroll_y.config(command=self.View_Table.yview)

self.View_Table.heading("pid", text="Product ID.")
self.View_Table.heading("ptype", text="Product Type")
self.View_Table.heading("mno", text="Model No")
self.View_Table.heading("manufacturer", text="Manufacturer")
self.View_Table.heading("department", text="Department")
self.View_Table.heading("location", text="Location")
self.View_Table.heading("incharge", text="Incharge")
self.View_Table.heading("comment", text="Comment")
```

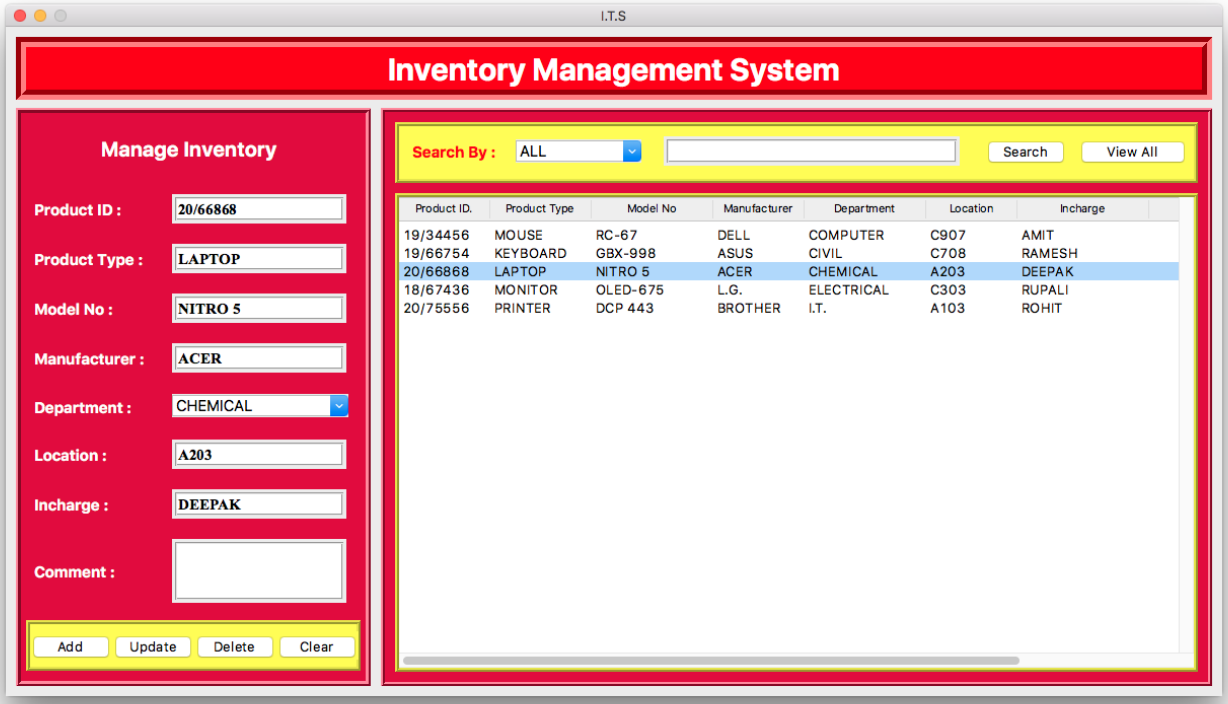
```
self.View_Table.column("pid", width=90)
self.View_Table.column("ptype", width=100)
self.View_Table.column("mno", width=120)
self.View_Table.column("manufacturer", width=90)
self.View_Table.column("department", width=120)
self.View_Table.column("location", width=90)
self.View_Table.column("incharge", width=130)
self.View_Table.column("comment", width=250)


self.View_Table["show"]="headings"
self.View_Table.pack(fill=BOTH, expand=1)
self.View_Table.bind("<ButtonRelease-1>", self.get_cursor)


self.view_data()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------

def add_items(self):
"""Function to ADD item to Database"""

if self.product_id_var.get()=="":
messagebox.showerror("Error","Product ID. cannot be blank!!!")

else:
try:
                con=sqlite3.connect('inventory.db')
                cur=con.cursor()
cur.execute(" insert into dmce_inventory values (?,?,?,?,?,?,?,?)",(
self.product_id_var.get(),
self.product_type_var.get(),
self.model_no_var.get(),
self.manufacturer_var.get(),
self.department_var.get(),
self.location_var.get(),
self.incharge_var.get(),
self.txt_comment.get('1.0',END),

))


con.commit()
self.view_data()
con.close()

except:
pass

else:
self.clear()

#-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------

def view_data(self):
"""Function to VIEW data into Table"""
```

```
        con=sqlite3.connect('inventory.db')
        cur=con.cursor()
cur.execute("select * from dmce_inventory")
        rows=cur.fetchall()
self.View_Table.delete(*self.View_Table.get_children())
if len(rows)!=0:
for row in rows:
self.View_Table.insert("", END, values=row)
con.commit()
con.close()

#---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
-------------------------------------------------

def clear(self):
"""Function to CLEAR all Input Fields"""

self.product_id_var.set("")
self.product_type_var.set("")
self.model_no_var.set("")
self.manufacturer_var.set("")
self.location_var.set("")
self.incharge_var.set("")
self.txt_comment.delete("1.0", END)

#---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
-------------------------------------------------

def get_cursor(self,event):
"""Function to SELECT a particular item"""

try:
cursor_row=self.View_Table.focus()
            contents=self.View_Table.item(cursor_row)
            row=contents["values"]

self.product_id_var.set(row[0])
self.product_type_var.set(row[1])
self.model_no_var.set(row[2])
self.manufacturer_var.set(row[3])
self.department_var.set(row[4])
self.location_var.set(row[5])
self.incharge_var.set(row[6])
self.txt_comment.delete("1.0", END)
self.txt_comment.insert(END, row[7])

except:
pass

#---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
-------------------------------------------------

def update_data(self):
"""Function to UPDATE an item of Database"""

        con=sqlite3.connect('inventory.db')
        cur=con.cursor()
```

```
cur.execute("update dmce_inventory set  product_type=? , model_no=? ,
manufacturer=? , department=? , location=? ,incharge=?, comment=? where
product_id=?",(
self.product_type_var.get(),
self.model_no_var.get(),
self.manufacturer_var.get(),
self.department_var.get(),
self.location_var.get(),
self.incharge_var.get(),
self.txt_comment.get('1.0',END),
self.product_id_var.get()

))
con.commit()
self.view_data()
self.clear()
con.close()

#-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------------------------------

def delete_data(self):
"""Function to DELETE an item from the Database"""

        con=sqlite3.connect('inventory.db')
        cur=con.cursor()
cur.execute("delete from dmce_inventory where
product_id=?",(self.product_id_var.get(),))
con.commit()
self.view_data()
self.clear()
con.close()

#-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------------------------------

def search_data(self):
"""Function to Search for items in Database"""

        con=sqlite3.connect('inventory.db')
        cur=con.cursor()

if self.search_by_var.get()=="Product ID.":
cur.execute("select * from dmce_inventory where product_id=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Product Type":
cur.execute("select * from dmce_inventory where product_type=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Model No":
cur.execute("select * from dmce_inventory where model_no=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Manufacturer":
```

```
cur.execute("select * from dmce_inventory where manufacturer=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Department":
cur.execute("select * from dmce_inventory where department=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Location":
cur.execute("select * from dmce_inventory where location=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

elifself.search_by_var.get()=="Incharge":
cur.execute("select * from dmce_inventory where incharge=?", (
self.search_txt_var.get(),))
            rows=cur.fetchall()

else:
cur.execute("select * from dmce_inventory where product_id=? OR product_type=? OR
model_no=? OR manufacturer=? OR department=? OR location=? OR incharge=?", (
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),
self.search_txt_var.get(),

))
            rows=cur.fetchall()


self.View_Table.delete(*self.View_Table.get_children())
if len(rows)!=0:
for row in rows:
self.View_Table.insert("", END, values=row)
con.commit()
con.close()

#---------------------------------------------------------------------------
---------------------------------------------------------------------------
-------------------------------------------------

if __name__ == '__main__':
"""START THE PROGRAM"""

create_database()
    root = Tk()
root.title("Inventory Tracking System")
root.iconbitmap("its_icon.ico")
ob = Inventory(root)
root.mainloop()
```

Output


I.T.S
Inventory Management System
Manage Inventory
Product ID :
20/66868
Product Type :
LAPTOP
Model No :
NITRO 5
Manufacturer :
ACER
Department :
CHEMICAL
Location :
A203
Incharge :
DEEPAK
Comment :
Add
Update
Delete
Clear
Search By :
ALL
Search
View All
Product ID.
Product Type
Model No
Manufacturer
Department
Location
Incharge
19/34456 MOUSE RC-67 DELL COMPUTER C907 AMIT
19/66754 KEYBOARD GBX-998 ASUS CIVIL C708 RAMESH
20/66868 LAPTOP NITRO 5 ACER CHEMICAL A203 DEEPAK
18/67436 MONITOR OLED-675 L.G. ELECTRICAL C303 RUPALI
20/75556 PRINTER DCP 443 BROTHER I.T. A103 ROHIT

Conclusion

While developing the system a conscious effort has been made to create and develop a software package, making use of available tools, techniques, and resources – that would generate a proper System

While making the system, an eye has been kept on making it as user-friendly, as cost-effective, and as flexible as possible.

As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs.

As in case of any system development processes where there are several shortcomings, there have been some shortcomings in the development of this system also. The project is still under modification.

References

- http://www.dreamincode.net
- http://www.a1vbcode.com
- “Introduction to Programming” by Gary J. Bronson.