Stock Management system

|  |  |
| --- | --- |
| Members: | Roll No: |
| Utkarsh Agarwal | 3 |
| Pratham Bhanshali | 6 |
| Abhishek Saraf | 35 |



INDEX

|  |  |  |
| --- | --- | --- |
| SR. NO. | TOPIC | PAGE |
| 1 | Background of the Company | 4 |
| 2 | Scope of The Project | 5 |
| 3 | Objectives | 6 |
| 4 | Existing System | 7 |
| 5 | Problems in the Existing System | 8 |
| 6 | Proposed System | 9 |
| 7 | Market potential | 10 |
| 8 | Usefulness | 11 |
| 9 | Technical Details | 12 |
| 10 | Modules Imported | 14 |
| 11 | Back End | 16 |
| 12 | Source Code | 18 |
| 13 | Flow of Control | 34 |
| 14 | Output | 35 |
| 15 | Validation | 40 |
| 16 | Limitations | 41 |
| 17 | Acknowledgment | 42 |
| 18 | Bibliography | 43 |

Background of the Company

**Name:** Reliable Paper Co. **State:** West Bengal

**Incorporation Date:** 14/07/1964 **Industry:** Paper Manufacturing

Reliable Paper Co. is a small-scale paper manufacturer. It supplies paper to other small businesses at highly competitive prices. The main office is located in Kolkata and the manufacturing unit is in a town close to Haldia. It also has a printing unit, where the manufactured paper is used to print posters and templates according to the client’s requirements. They sell paper in many different sizes from large billboards to small business cards. The owner of the organization claims that due to technological advancements and increasing reliance on digital technologies to store and access data the paper business is slowly declining.

# Scope of the project

The software's present scope is to allow users to obtain and change information from a centralised MySQL database. Due to its limited capability and simplicity, users do not require much training time.

This project will help local company owners provide more timely and efficient shop information. This programme was created to automate tasks that would otherwise take more time if done manually.

# Objectives of the Project

The objectives of inventory management are operational and financial. In operational, materials and stock should be available in sufficient amount whereas, in functional, the minimum working capital should be locked in. The basic principle of inventory management is to hold costs. The investment put in inventory is very high, especially for those businesses that deal in manufacturing, wholesale, and retail trade. For a company, keeping record and track of inventory, updating it and using the same data to track profits, needs and sales are very important but for a small business inventory management system can make or break the ability to keep up with sales and demand.

* To ensure a continuous supply of materials and stock so that production should not suffer at the time of customers’ demand.
* To maintain the availability of materials whenever and wherever required in enough quantity.
* To minimize loss through deterioration, pilferage, wastages, and damages.
* To eliminate duplication in ordering stocks.
* To facilitate furnishing of data for short and long-term planning with a controlled inventory.

# Existing System

The organization currently uses a paid software for their inventory management. The software offers a range of features: -

* Inventory Management Solution
* eCommerce Management
* Expense Management
* Tracking delivery orders
* GST invoice generation
* Vendor credit tracker

# Problems in the existing system

* The unintuitive UI of the current software has led to human errors in multiple occasions because of which the company had to incur huge losses;
* The current software is expensive, and the company can’t afford to pay for such expensive services;
* The software does provide a range of features, but most of them are irrelevant for the company as it needs the software only for warehouse inventory management;
* The company wants a more personalized software catering to their demands and needs
* Due to the unintuitive nature of the UI, it is difficult to store information related to the goods. When there is too much information, accessing any of it becomes a complex and time-consuming job.
* Storing and retrieving information not only a challenging process, it also requires a lot of time and effort.

# Proposed System

In the proposed system, the administrator can access and write information and records. They may also add, edit and delete the records according to their needs.

This project will help in the smoother management of products. Because of its simplistic design and use, this software does not need extensive user training. Modules or functionalities can be created and incorporated to the system to meet the demands of future users. Without much effort, every element of the software and reports may be updated separately.

This software project aims to improve the present record-keeping system by assisting local company owners in retrieving up-to-date information in the correct format at the right time and in a user-friendly environment.

* Computerization - All aspects of the management system will be computerized.
* Automation - This management system's automation function will make the task of keeping records in the papers easier.

# Market Potential

The software is designed specifically for The Reliable Paper Company. So, there is little chance that it can fulfil the needs of other businesses. However, if an organization needs a simple software for warehouse resource management, this might fulfil their requirements. The simple and intuitive UI can be used by anyone even if they are not technologically sound.

# Usefulness

Inventory management is important, especially to small businesses, as it helps them prevent stockouts, manage multiple locations, and ensure accurate recordkeeping. An inventory solution makes these processes easier than trying to do them all manually.

* **Optimization solutions drive actionable results for users and stakeholders:**
* **Management:** Better customer service and order completion boosts the reputation of the company and aids executive management goals to grow the brand and market share.
* **Finance:** Financial managers are better equipped to achieve strategic financial goals owing to reduced carrying costs and reduced working capital tied up in stock.
* **Operations:** Operational managers will benefit from reduced staffing costs and an increase in efficiency and ease of operations, owing to the increased automation of the system. It also reduces the element of human error, which can drastically increase profit margins annually.
* **Enhanced features and functionality for inventory management:**
* Intelligent monitoring of stock enables accurate stock procurement and the elimination of obsolescence. At the same time, stock is always available, while service rates remain high.
* A centralized system enables inventory to be monitored across the business so that stock can be redistributed as necessary between stocking locations.
* Inventory replenishment becomes automated
* **Quick return on investment:**
* Since deployment of optimization solutions can be completed within weeks, businesses can actionably start cutting costs to drive results within the first quarter of launch. This process results in a quick reduction of inventory levels and carrying costs while boosting up profit margins.

# Technical details

**Hardware used:**

**Processor** - Intel core i5 10th gen

**Hard disk -** 512 GB

**Ram -** 4 GB

**System type -** 64-bit Operating System

**Software used:**

**Front End -** Python 3.7

**Back End -** MySQL

**Operating system -** Microsoft Windows 10

**For Python-MySQL connectivity:**

**Host -** localhost

**User -** root

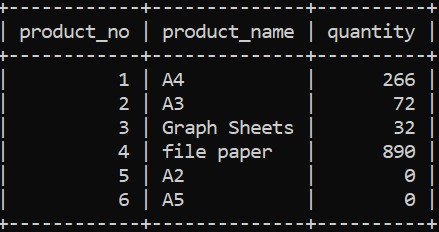
**Password -** Utkarsh123#

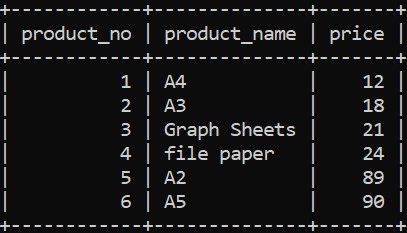
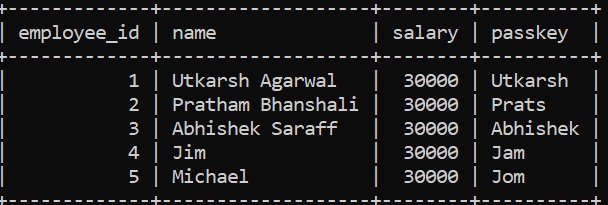
**Database -** project

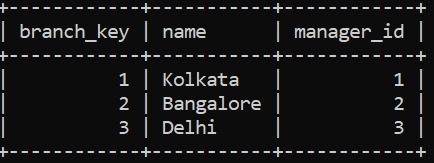
|  |  |  |
| --- | --- | --- |
| Technical Design | | |
| Main.py | | |
| FUNCTIONS | PURPOSE | |
| nextpage() | Going from welcome to login page | |
| connect() | For connecting Python file to database | |
| login( *username, password* ) | Searching for the details of the user in the database | |
| manager( *username* ) | To distinguish between manager and employee | |
| login\_page() | Front end of login page | |
| loginnow() | Verifying login credentials | |
| sidebar\_buttons() | Front end of various program functions | |
| GetValue( *event* ) | To autofill entry boxes from the MySQL database | |
| employee\_box() | To display MySQL employee table | |
| del\_employees() | Delete data of the given employee | |
| input\_employees() | Entering data of a new employee | |
| item\_update() | Updating record of a given employee | |
| display() | Front end for create, read, update and delete functions for employees table | |
| add\_TO\_items() | Add a new item into the inventory | |
| add\_TO\_inventory() | Add more quantity for an existing item | |
| find\_inventory( *product* ) | Finding details of the product from the inventory | |
| update\_quan( *inventory\_alter,quan\_added* ) | Updates the quantity in the database | |
| add\_inventory() | Front end for adding goods to inventory | |
| remove\_FROM\_inventory() | Removes quantity for an existing item | |
| salary(*username,product\_name,item\_quantityS*) | Adds commission to salary of an employee on the basis of their sales | |
| delete\_inventory() | Front end for deleting good from the inventory | |
| MODULES IMPORTED | | |
| tkinter | | Python GUI interface |
| mysql.connector | | Connector MySQL database to python |
| PIL | | Image processing in tkinter |

|  |  |
| --- | --- |
| Stock Management.py | |
| create\_table() | Creates table and speed up the process of making table by automating it |
| input\_branch() | Speeds up the process of entering values by automating it |
| input\_inventory() | Speeds up the process of entering values by automating it |

Back End







**Source Code**

**Stock Management.py**

import mysql.connector as con

def connect():

db = con.connect(

database='project',

host="localhost",

user="root",

password="Utkarsh123#"

)

return db

employees\_table='''create table if not exists employees (

employee\_id INT primary key auto\_increment,

name varchar(20) not null ,

salary int,

passkey varchar(9)

);

'''

branch\_table='''create table if not exists branch (

branch\_key int primary key auto\_increment,

name varchar(20) not null,

manager\_id int default null,

FOREIGN KEY (manager\_id) REFERENCES employees(employee\_id)

);

'''

items\_table='''create table if not exists items (

product\_no int primary key auto\_increment,

product\_name varchar(20) unique,

price int

);'''

inventory\_table='''create table if not exists inventory (

product\_no int primary key auto\_increment,

product\_name varchar(20) unique,

quantity int,

FOREIGN KEY (product\_no) REFERENCES items(product\_no),

FOREIGN KEY (product\_name) REFERENCES items(product\_name)

);'''

def create\_table(a):

db=connect()

cur=db.cursor()

cur.execute(a)

db.commit()

db.close()

list\_table=[employees\_table,branch\_table,items\_table,inventory\_table]

for table\_name in list\_table:

create\_table(table\_name)

def input\_employees():

db=connect()

cur=db.cursor()

sql = "INSERT INTO employees(name,salary,passkey) VALUES (%s,%s, %s)"

name,salary,passkey=input('name of employee'),input('salary'),input('password')

val = (name,salary,passkey)

cur.execute(sql, val)

db.commit()

def input\_branch():

db=connect()

cur=db.cursor()

sql = "INSERT INTO branch(name) VALUES (%s)"

name=input('name of branch')

val = (name)

cur.execute(sql, val)

db.commit()

def input\_inventory():

db=connect()

cur=db.cursor()

sql = "INSERT INTO items(product\_name,price) VALUES (%s, %s)"

product\_name,product\_price=input('name of product'),input('price of product')

val = (product\_name,product\_price)

cur.execute(sql, val)

sql="insert into inventory(product\_name,quantity) value (%s,%s)"

val=(product\_name,0)

cur.execute(sql,val)

db.commit()

def input\_items():

db=connect()

cur=db.cursor()

sql = "INSERT INTO items(product\_name,price) VALUES (%s, %s)"

product\_name,product\_price=input('name of product'),input('price of product')

val = (product\_name,product\_price)

cur.execute(sql, val)

sql="insert into inventory(product\_name,quantity) value (%s,%s)"

val=(product\_name,0)

cur.execute(sql,val)

db.commit()

**main.py**

from tkinter import \*

import tkinter as tk

from tkinter import ttk,messagebox

from PIL import ImageTk,Image

import mysql.connector as con

root=Tk()

app\_width = 600

app\_height = 600

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

x = (screen\_width / 2) - (app\_width / 2)

y = (screen\_height / 2 ) - (app\_height / 2)

root.geometry(f'{app\_width}x{app\_height}+{int(x)}+{int(y)}')

#root.configure(bg='#F9F6EE')

frame\_main = Frame(root,bg='#F9F6EE',width=app\_width,height=app\_height)

frame\_main.place(x=0,y=0)

my\_img=Image.open("hello2.jpeg")

my\_img=my\_img.resize((400,400), Image.ANTIALIAS)

my\_img1 = ImageTk.PhotoImage(my\_img)

my\_label = Label(frame\_main,image=my\_img1,bg="red")

my\_label.place(x= 100,y=50)

def nextpage():

for widgets in frame\_main.winfo\_children():

widgets.destroy()

def connect():

db = con.connect(

database='project',

host="localhost",

user="root",

password="Utkarsh123#"

)

return db

def login(username, password):

db=connect()

cur=db.cursor()

sql='select passkey from employees where employee\_id = %s'

cur.execute(sql,(username,))

myresult=cur.fetchone()

for i in myresult:

if i==password:

return myresult

def manager(username):

db=connect()

cur=db.cursor()

sql='select \* from branch where branch\_key=%i'%username

cur.execute(sql)

myresult=cur.fetchone()

return myresult

def login\_page():

for widgets in frame\_main.winfo\_children():

widgets.destroy()

my\_img2=Image.open("login1.jpg")

my\_img2=my\_img2.resize((200,100), Image.ANTIALIAS)

my\_img3 = ImageTk.PhotoImage(my\_img2)

my\_label = Label(frame\_main,image=my\_img3,bg="#F9F6EE")

my\_label.place(x= 200,y=50)

label\_us=Label(frame\_main,text='Username',font=("Times New Roman",20),fg="black")

label\_us.config(bg='#F9F6EE')

label\_us.place(x=150,y=160)

entry\_us=Entry(frame\_main,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

entry\_us.place(x=280,y=160)

label\_pass=Label(frame\_main,text='Password',font=("Times New Roman",20),fg="black")

label\_pass.config(bg='#F9F6EE')

label\_pass.place(x=150,y=200)

entry\_pass=Entry(frame\_main,show="\*",highlightcolor="black",highlightbackground = "black",highlightthickness=1)

entry\_pass.place(x=280,y=200)

def loginnow():

if entry\_us.get().strip()=="" or entry\_pass.get().strip()=="":

messagebox.showwarning("showwarning", "All Fields are Important")

else:

try:

username=int(entry\_us.get())

except ValueError:

username=entry\_us.get()

password=entry\_pass.get()

if login(username, password)==None:

label\_pass=Label(frame\_main,text='\*Incorrect Username or Password\*',font=("Times New Roman",20),fg="red")

label\_pass.config(bg='#F9F6EE')

label\_pass.place(x=150,y=260)

entry\_pass.delete(0,END)

else:

def sidebar\_buttons():

global items

global plus

global minus

global items\_b

global add\_b

global minus\_b

global logout\_b

items = ImageTk.PhotoImage(Image.open('items.png').resize((40,40),Image.ANTIALIAS))

plus = ImageTk.PhotoImage(Image.open('plus.png').resize((40,40),Image.ANTIALIAS))

minus = ImageTk.PhotoImage(Image.open('minus.png').resize((40,40),Image.ANTIALIAS))

items\_b = Button(frame\_sidebar,image=items,command=display,bg='pink')

add\_b = Button(frame\_sidebar,image=plus,command=add\_inventory,bg='pink')

minus\_b = Button(frame\_sidebar,image=minus,command=delete\_inventory,bg='pink')

logout\_b=Button(frame\_sidebar,text="Exit",command=login\_page,bg='pink')

def GetValue(event):

e1.delete(0, END)

e2.delete(0, END)

e3.delete(0, END)

row\_id = Display\_table.selection()[0]

select = Display\_table.set(row\_id)

e1.insert(0,select['name'])

e2.insert(0,select['salary'])

e3.insert(0,select['passkey'])

def GetValue2(event):

e5.delete(0, END)

e6.delete(0, END)

row\_id = Display\_table2.selection()[0]

select = Display\_table2.set(row\_id)

e5.insert(0,select['product\_name'])

e6.insert(0,select['quantity'])

def GetValue3(event):

e8.delete(0, END)

e9.delete(0, END)

row\_id = Display\_table3.selection()[0]

select = Display\_table3.set(row\_id)

e8.insert(0,select['product\_name'])

e9.insert(0,select['quantity'])

def employee\_box():

global Display\_table

db=connect()

cur=db.cursor()

col\_names = ('employee\_id','name','salary','passkey')

col\_text=['password','salary','name','employee\_id']

Display\_table = ttk.Treeview(frame\_user, columns=col\_names, show='headings')

for column1 in col\_names:

Display\_table.column(column1,anchor=CENTER, stretch=NO, width=120)

Display\_table.heading(column1, text=col\_text.pop())

Display\_table.place(x=10,y=300)

cur.execute("SELECT \* FROM employees")

records = cur.fetchall()

for i, (employee\_id,name,salary,passkey) in enumerate(records):

Display\_table.insert("", "end", values=(employee\_id,name,salary,passkey))

Display\_table.bind('<Double-Button-1>',GetValue)

db.close()

def del\_employees():

db=connect()

cur=db.cursor()

def find\_emp(pname):

db=connect()

cur=db.cursor()

val=(pname,)

sql='select \* from employees where name=%s'

cur.execute(sql,val)

myresult=cur.fetchone()

return myresult[0]

if e1.get().strip()=="" or e2.get().strip()=="" or e3.get().strip()=="":

messagebox.showwarning("showwarning", "All Fields are Important")

else:

pname = e1.get()

psalary = e2.get()

ppasskey = e3.get()

val=(pname,psalary,ppasskey)

if manager(find\_emp(pname))==None:

val=(pname,psalary,ppasskey)

sql='delete from employees where name=%s and salary=%s and passkey=%s'

cur.execute(sql,val)

db.commit()

# messagebox.showinfo("Informantion", "Record Deleted Successfully...")

else:

r=manager(find\_emp(pname))

print(r)

val=(int(r[2]),)

sql='delete from branch where manager\_id=%i'%val

cur.execute(sql)

db.commit()

sql='delete from employees where employee\_id=%i'%val

cur.execute(sql)

db.commit()

# messagebox.showinfo("Informantion", "Record Deleted Successfully...")

messagebox.showinfo("Informantion", "Record Deleted Successfully...")

employee\_box()

e1.delete(0,END)

e2.delete(0,END)

e3.delete(0,END)

e1.focus\_set()

def input\_employees():

db=connect()

cur=db.cursor()

if e1.get().strip()=="" or e2.get().strip()=="" or e3.get().strip()=="":

messagebox.showwarning("showwarning", "All Fields are Important")

else:

pname = e1.get()

psalary= e2.get()

ppasskey = e3.get()

sql = "INSERT INTO employees(name,salary,passkey) VALUES (%s,%s, %s)"

val = (pname,psalary,ppasskey)

cur.execute(sql, val)

db.commit()

messagebox.showinfo("Informantion", "Record Inserted Successfully...")

employee\_box()

e1.delete(0,END)

e2.delete(0,END)

e3.delete(0,END)

e1.focus\_set()

def item\_update():

if e1.get().strip()=="" or e2.get().strip()=="" or e3.get().strip()=="":

messagebox.showwarning("showwarning", "All Fields are Important")

else:

pname = e1.get()

psalary = e2.get()

ppasskey = e3.get()

val=(ppasskey,psalary,pname)

db=connect()

cur=db.cursor()

sql = "Update employees set passkey= %s,salary= %s where name= %s"

cur.execute(sql, val)

db.commit()

messagebox.showinfo("information", "Record Updatd successfully...")

e1.delete(0, END)

e2.delete(0, END)

e3.delete(0, END)

e1.focus\_set()

employee\_box()

def display():

for widgets in frame\_user.winfo\_children():

widgets.destroy()

frame\_user.update()

global e1

global e2

global e3

db=connect()

cur=db.cursor()

h1=Label(frame\_user, text="EMPLOYEE", fg="pink", font=("Times New Roman", 24))

h1.config(bg='#F9F6EE')

h1.place(relx=0.5,y=20,anchor=CENTER)

sub1=Label(frame\_user, text="Employee Name",font=("Times New Roman", 14))

sub1.config(bg='#F9F6EE')

sub1.place(x=70, y=80)

sub2=Label(frame\_user, text="Salary",font=("Times New Roman", 14))

sub2.config(bg='#F9F6EE')

sub2.place(x=70, y=110)

sub3=Label(frame\_user, text="Password",font=("Times New Roman", 14))

sub3.config(bg='#F9F6EE')

sub3.place(x=70, y=140)

e1 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e1.place(x=200, y=80)

e2 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e2.place(x=200, y=110)

e3 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e3.place(x=200, y=140)

add\_button=Button(frame\_user, text="Add",command=input\_employees,height=3, width= 13)

add\_button.place(x=70, y=200)

update\_button=Button(frame\_user, text="Update",command=item\_update,height=3, width= 13)

update\_button.place(x=200, y=200)

delete\_button=Button(frame\_user, text="Delete",command=del\_employees,height=3, width= 13)

delete\_button.place(x=330, y=200)

db.close()

employee\_box()

def add\_TO\_items():

global item\_add

db=connect()

cur=db.cursor()

sql = "INSERT INTO items(product\_name,price) VALUES (%s, %s)"

product\_name,product\_price=item\_add,int(e10.get())

val = (product\_name,product\_price)

cur.execute(sql, val)

sql="insert into inventory(product\_name,quantity) value (%s,%s)"

val=(product\_name,0)

cur.execute(sql,val)

db.commit()

e10.delete(0,END)

for widgets in frame\_user.winfo\_children():

widgets.destroy()

frame\_user.update()

add\_inventory()

def add\_TO\_inventory():

def find\_inventory(product):

db=connect()

cur=db.cursor()

sql='select \* from inventory where product\_name=%s'

cur.execute(sql,(product,))

myresult=cur.fetchone()

return myresult

def update\_quan(inventory\_alter,quan\_added):

db=connect()

cur=db.cursor()

product\_no=inventory\_alter[0]

inventory\_quan=inventory\_alter[2]+quan\_added

val=(inventory\_quan,product\_no)

sql='''update inventory

set quantity=%s

where product\_no=%i

'''%val

cur.execute(sql)

db.commit()

inventory\_box()

if e5.get().strip()=="" or e6.get().strip()=="":

messagebox.showwarning("showwarning", "All Fields are Important")

e5.delete(0, END)

e6.delete(0, END)

e5.focus\_set()

inventory\_box()

else:

global item\_add

global item\_quantityA

item\_add=e5.get()

item\_quantityA=int(e6.get())

inventory\_alter=find\_inventory(item\_add)

e5.delete(0, END)

e6.delete(0, END)

e5.focus\_set()

inventory\_box()

if inventory\_alter==None:

response=messagebox.askyesno("Information", "Do you wish to enter a new record?")

if response==1:

for widgets in frame\_user.winfo\_children():

widgets.destroy()

frame\_user.update()

global e10

db=connect()

cur=db.cursor()

h3=Label(frame\_user, text="ADD NEW ITEM", fg="pink", font=("Times New Roman", 24))

h3.config(bg='#F9F6EE')

h3.place(relx=0.5,y=20,anchor=CENTER)

sub10=Label(frame\_user, text="Price",font=("Times New Roman", 14))

sub10.config(bg='#F9F6EE')

sub10.place(x=70, y=110)

e10 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e10.place(x=200, y=110)

Button(frame\_user, text="Add",command=add\_TO\_items,height=3, width= 13).place(x=150, y=200)

db.close()

inventory\_box()

else:

update\_quan(inventory\_alter,item\_quantityA)

def inventory\_box():

db=connect()

cur=db.cursor()

global Display\_table2

col\_names = ('product\_no', 'product\_name', 'quantity')

col\_text=["Quantity","Product Name","Product Number"]

Display\_table2 = ttk.Treeview(frame\_user, columns=col\_names, show='headings')

for column1 in col\_names:

Display\_table2.column(column1,anchor=CENTER, stretch=NO, width=100)

Display\_table2.heading(column1, text=col\_text.pop())

Display\_table2.place(x=70,y=300)

cur.execute("SELECT \* FROM inventory")

records = cur.fetchall()

for i, (product\_no,product\_name,quantity) in enumerate(records):

Display\_table2.insert("", "end", values=(product\_no,product\_name,quantity))

db.close()

Display\_table2.bind('<Double-Button-1>',GetValue2)

def inventory\_box2():

db=connect()

cur=db.cursor()

global Display\_table3

col\_names = ('product\_no', 'product\_name', 'quantity')

col\_text=["Quantity","Product Name","Product Number"]

Display\_table3 = ttk.Treeview(frame\_user, columns=col\_names, show='headings')

for column1 in col\_names:

Display\_table3.column(column1,anchor=CENTER, stretch=NO, width=100)

Display\_table3.heading(column1, text=col\_text.pop())

Display\_table3.place(x=70,y=300)

cur.execute("SELECT \* FROM inventory")

records = cur.fetchall()

for i, (product\_no,product\_name,quantity) in enumerate(records):

Display\_table3.insert("", "end", values=(product\_no,product\_name,quantity))

db.close()

Display\_table3.bind('<Double-Button-1>',GetValue3)

def add\_inventory():

for widgets in frame\_user.winfo\_children():

widgets.destroy()

frame\_user.update()

global e5

global e6

db=connect()

cur=db.cursor()

h2=Label(frame\_user, text="ADD TO INVENTORY", fg="pink", font=("Times New Roman", 24))

h2.config(bg='#F9F6EE')

h2.place(relx=0.5,y=20,anchor=CENTER)

sub5=Label(frame\_user, text="Product Name",font=("Times New Roman", 14))

sub5.config(bg='#F9F6EE')

sub5.place(x=70, y=110)

sub6=Label(frame\_user, text="Quantity",font=("Times New Roman", 14))

sub6.config(bg='#F9F6EE')

sub6.place(x=70, y=140)

e5 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e5.place(x=200, y=110)

e6 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e6.place(x=200, y=140)

Button(frame\_user, text="Add",command=add\_TO\_inventory,height=3, width= 13).place(x=150, y=200)

db.close()

inventory\_box()

def remove\_FROM\_inventory():

def find\_inventory(product):

# product=input('enter')

db=connect()

cur=db.cursor()

sql='select \* from inventory where product\_name=%s'

cur.execute(sql,(product,))

myresult=cur.fetchone()

return myresult

def update\_quan(inventory\_alter,quan\_added):

db=connect()

cur=db.cursor()

product\_no=inventory\_alter[0]

inventory\_quan=inventory\_alter[2]+quan\_added

if inventory\_quan<0:

messagebox.showwarning("showwarning", "Insufficient Quantity in Inventory")

else:

val=(inventory\_quan,product\_no)

sql='''update inventory

set quantity=%s

where product\_no=%i

'''%val

cur.execute(sql)

db.commit()

inventory\_box2()

def salary(username,product\_name,item\_quantityS):

db=connect()

cur=db.cursor()

sql='select \* from items where product\_name=%s'

cur.execute(sql,(product\_name,))

item\_details=cur.fetchone()

item\_price=item\_details[2]\*abs(item\_quantityS)

employee\_commission=(15/100)\*item\_price

sql='select \* from employees where employee\_id=%i'%username

cur.execute(sql)

employee\_details=cur.fetchone()

employee\_salary=employee\_details[2]+employee\_commission

val=(employee\_salary,username)

sql='''update employees

set salary=%i

where employee\_id=%i

'''%val

cur.execute(sql)

db.commit()

if e8.get().strip()=="" or e9.get().strip()=="":

messagebox.showwarning("showwarning", "All Fields are Important")

e8.delete(0, END)

e9.delete(0, END)

e8.focus\_set()

inventory\_box2()

else:

item\_subtract=e8.get()

item\_quantityS=-(int(e9.get()))

inventory\_alter=find\_inventory(item\_subtract)

update\_quan(inventory\_alter,item\_quantityS)

salary(username,item\_subtract,item\_quantityS)

e8.delete(0, END)

e9.delete(0, END)

e8.focus\_set()

inventory\_box2()

def delete\_inventory():

for widgets in frame\_user.winfo\_children():

widgets.destroy()

frame\_user.update()

global e8

global e9

db=connect()

cur=db.cursor()

h3=Label(frame\_user, text="REMOVE FROM INVENTORY", fg="pink", font=("Times New Roman", 24))

h3.config(bg='#F9F6EE')

h3.place(relx=0.5,y=20,anchor=CENTER)

sub8=Label(frame\_user, text="Product Name",font=("Times New Roman", 14))

sub8.config(bg='#F9F6EE')

sub8.place(x=70, y=110)

sub9=Label(frame\_user, text="Quantity",font=("Times New Roman", 14))

sub9.config(bg='#F9F6EE')

sub9.place(x=70, y=140)

e8 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e8.place(x=200, y=110)

e9 = Entry(frame\_user,highlightcolor="black",highlightbackground = "black",highlightthickness=1)

e9.place(x=200, y=140)

Button(frame\_user, text="Delete",command=remove\_FROM\_inventory,height=3, width= 13).place(x=150, y=200)

db.close()

inventory\_box2()

if manager(username)==None:

for widgets in frame\_main.winfo\_children():

widgets.destroy()

frame\_user = Frame(frame\_main,bg='#F9F6EE',width=app\_width-60,height=app\_height)

frame\_user.place(x=60,y=0)

frame\_user.grid\_propagate(False)

add\_inventory()

frame\_sidebar = Frame(frame\_main,bg='pink',width=60,height=app\_height)

frame\_sidebar.place(x=0,y=0)

frame\_sidebar.grid\_propagate(False)

sidebar\_buttons()

add\_b.grid(row=0,column=0,padx=10,pady=10)

minus\_b.grid(row=1,column=0,pady=10)

logout\_b.grid(row=2,column=0,pady=400)

else:

for widgets in frame\_main.winfo\_children():

widgets.destroy()

frame\_user = Frame(frame\_main,bg='#F9F6EE',width=app\_width-60,height=app\_height)

frame\_user.place(x=60,y=0)

frame\_user.grid\_propagate(False)

display()

frame\_sidebar = Frame(frame\_main,bg='pink',width=60,height=app\_height)

frame\_sidebar.place(x=0,y=0)

frame\_sidebar.grid\_propagate(False)

sidebar\_buttons()

items\_b.grid(row=0,column=0,pady=10,padx=10)

add\_b.grid(row=1,column=0,pady=10)

minus\_b.grid(row=2,column=0,pady=10)

logout\_b.grid(row=3,column=0,pady=350)

Button(frame\_main, text="Login Now", command=loginnow,background="pink").place(x=250,y=240)

root.resizable(False, False)

root.mainloop()

login\_page()

Button(frame\_main, text="Continue", command=nextpage,background="pink",width=10,height=1,font=("Times New Roman",20)).place(x=250,y=500)

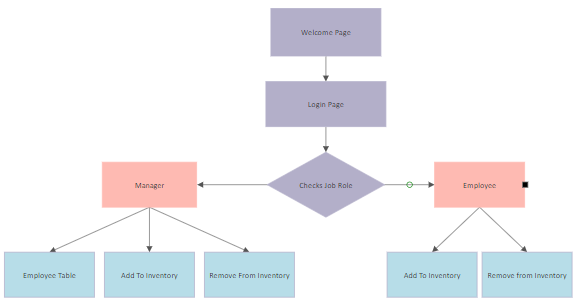
root.update\_idletasks()

root.resizable(False, False)

frame\_main.grid\_propagate(False)

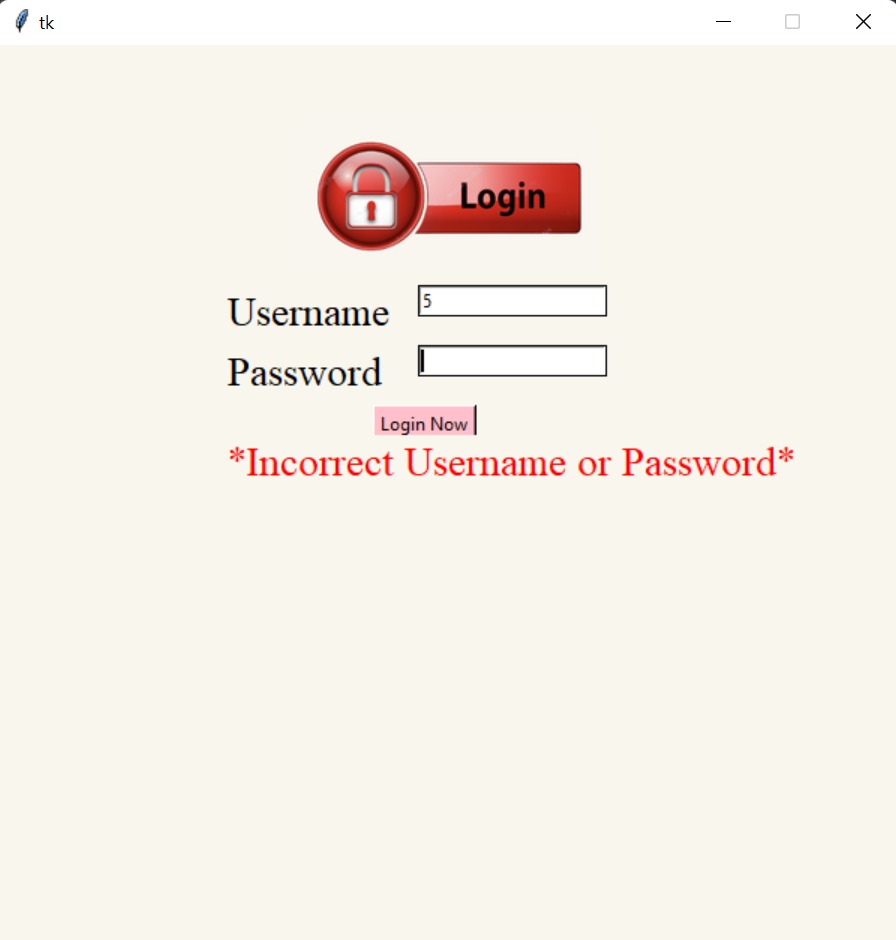
root.mainloop()

Flow of Control

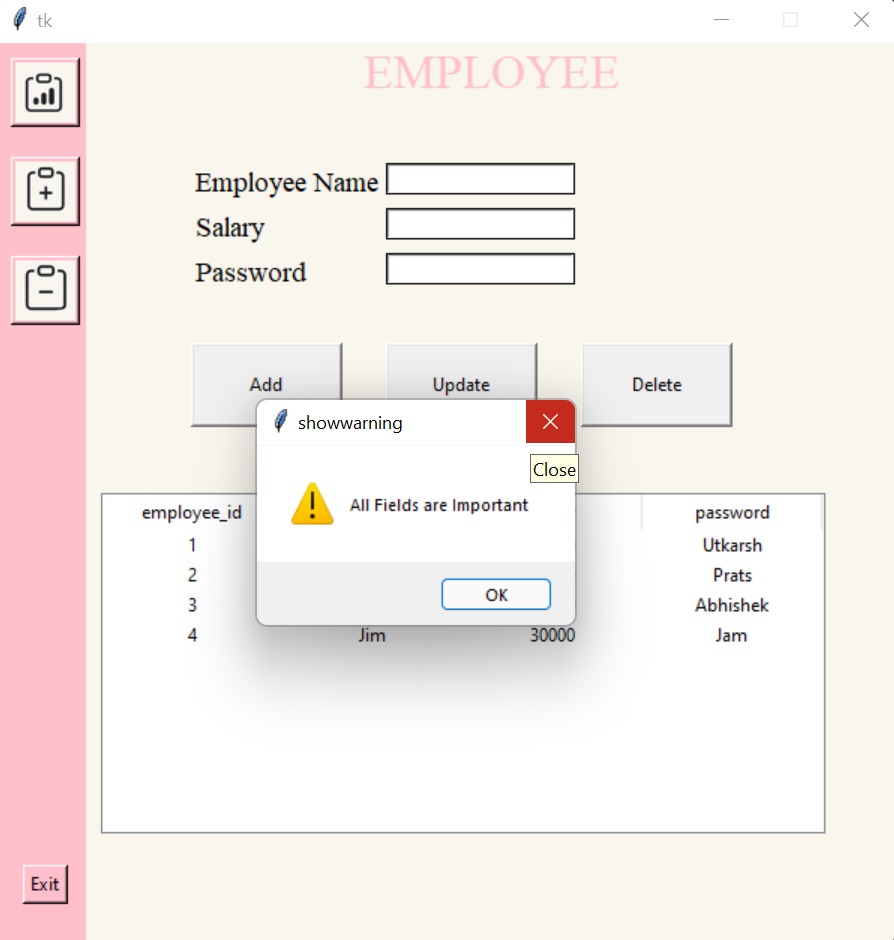


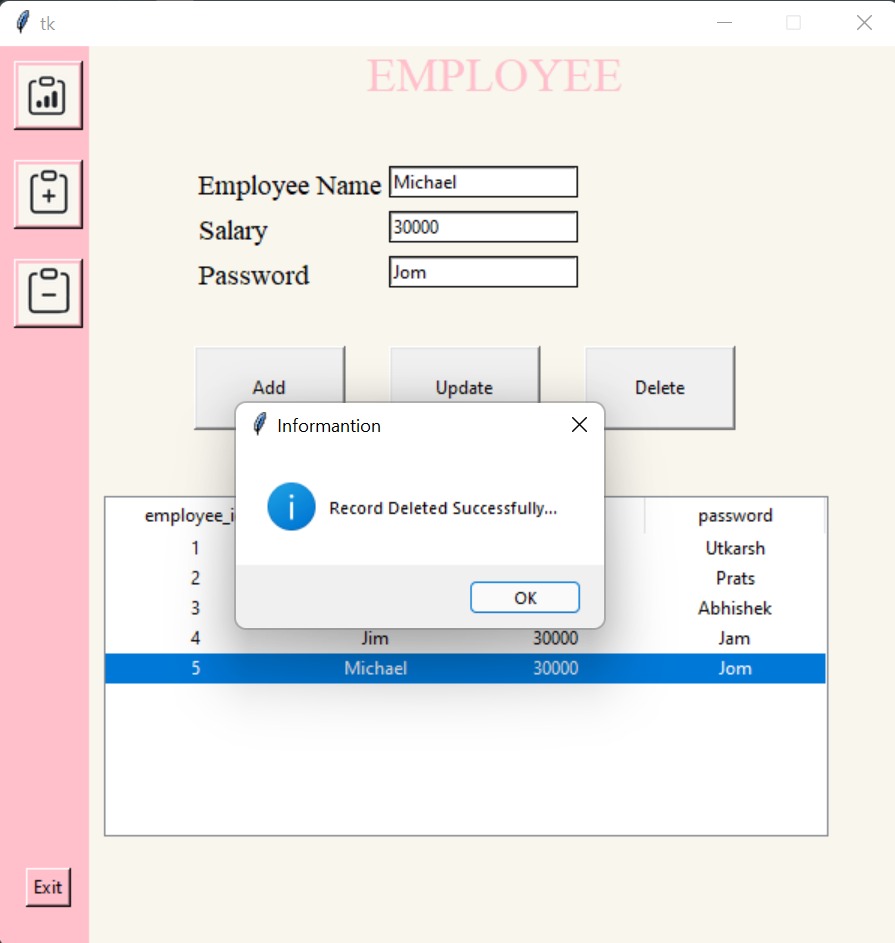
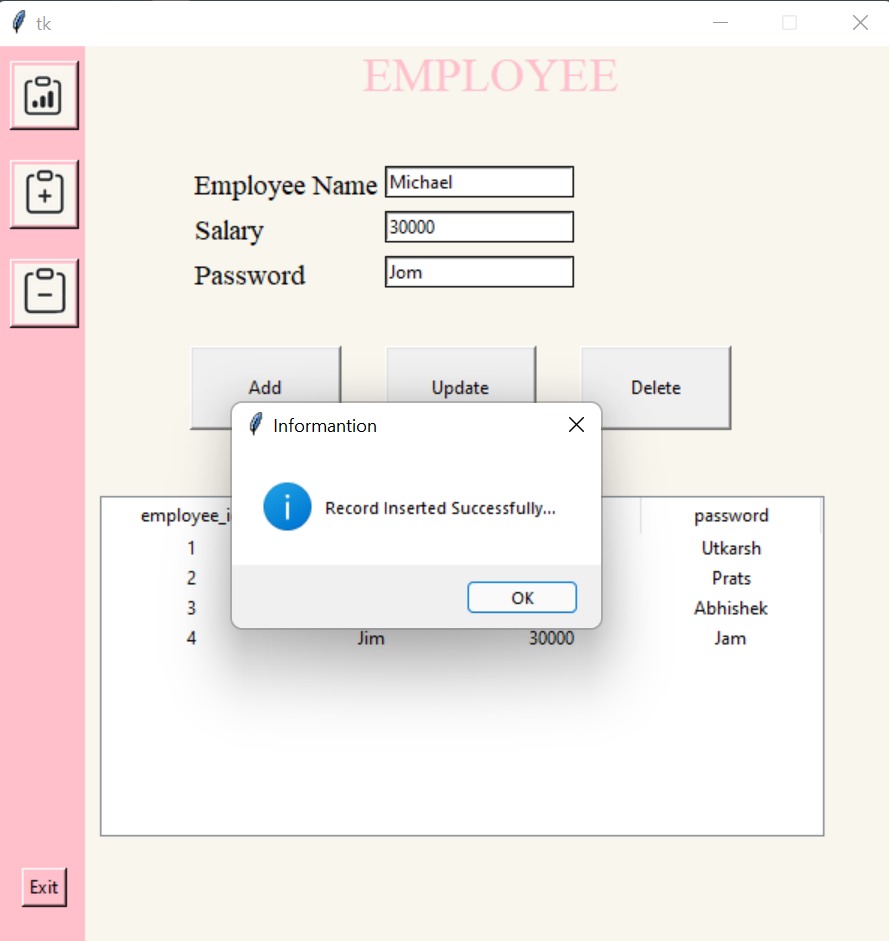
Output

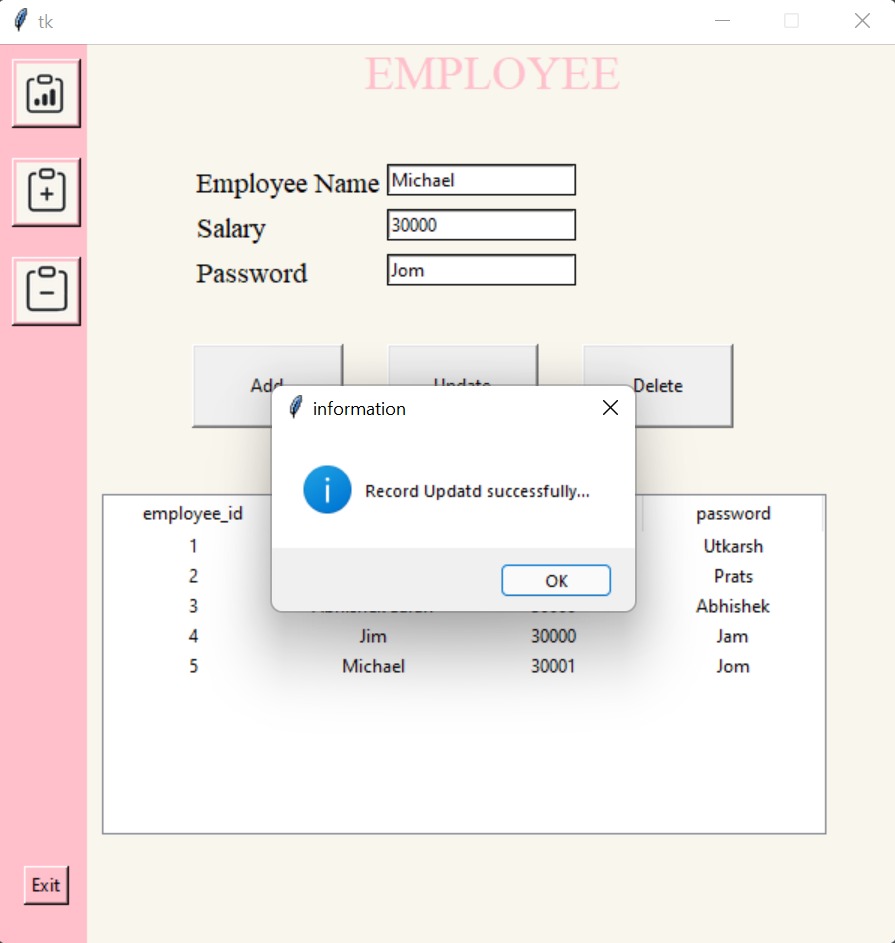




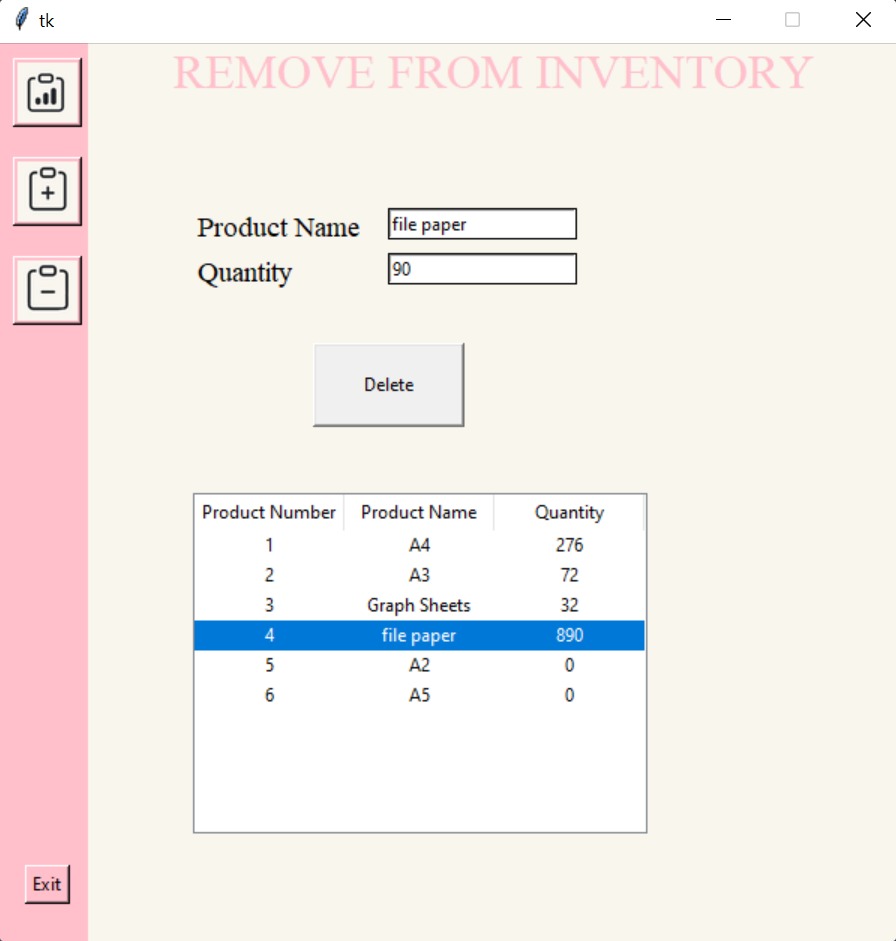




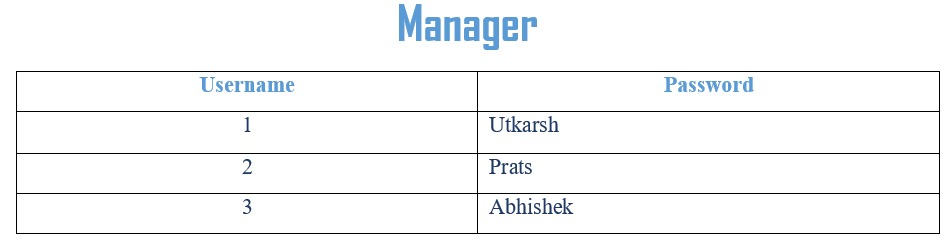


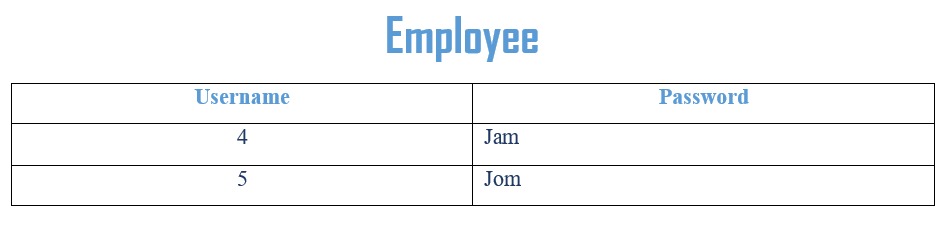






Validations





Limitations

Our stock management system has very few features compared to other industry leaders. We hope in the near future we are able to add more features so that the software is sufficient to manage large and small businesses alike.

The UI though very user-friendly and simple, is boring and lacks the charm. We intend to integrate more technologies and make the software more interesting.

Cross-platform compatibility is limited and the UI becomes distorted when program is run on different OS.

Acknowledgement

I would like to express my sincere gratitude towards our computer teacher, Ms. S. Mukherjee, for her inspiration and support in the completion of this project.

Bibliography

Computer Science with Python by Sumita Arora

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/>

<https://realpython.com/python-gui-tkinter/>

<https://www.javatpoint.com/python-tkinter>

<https://stackoverflow.com/>

<https://pynative.com/python-mysql-database-connection/>