**KENDRIYA VIDYALAYA ONGC**

**JORHAT, ASSAM**



***Academic Year: 2022-23***

*A project report on*

*STATIONARY MANAGEMENT*

**Submitted By: Submitted to:**

*Dinku Hazarika Mahesh Chand Pal*

*Class XII*

*Roll no.- 16657584*

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| Sno. | Description | Page No. |
| 01 | ACKNOWLEDGEMENT | **3** |
| 02 | GETTING STARTED | **4** |
| 03 | CODE | **5-18** |
| 04 | OUTPUT | **19-28** |
| 05 | BIBLIOGRAPHY | **29** |

**ACKNOWLEDGEMENT**

*I am grateful and feel thankful to our Computer Science teacher, Mr. Mahesh Chand Pal. It was his unfailing push and constant inspiration that his work could take the present shape.*

*Turning aspirations into reality and guiding us in each step to accomplish the task and complete this project successfully.*

**GETTING STARTED**

**SOME REQUIREMENTS**

1. Pycharm latest version required.
2. MySql latest version required.
3. PrettyTable package need to be downloaded in pycharm. To install PrettyTable, Go to file, settings, project settings, project interpreter, click on the plus icon to install new package and just search an install ‘prettytable’ .
4. Use mono-spaced font for best experience.
5. Also go to File, Settings, Editor, Font, and enable ligature feature for better experience. This feature use several characters as a single ligature, and enhances the code readability.

**INITIALIZING PROJECT**

To create database, open create\_database.py and run it. It should create required databases to run the main program. It also add some stationary item data for the ease to check the program.

**CODE**

**Create\_database.py**

import pickle  
import mysql.connector as ms  
  
pswrd = input('Enter your mysql server password: ')  
  
mydb = ms.connect(host='localhost', user='root', password=pswrd)  
my\_cursor = mydb.cursor()  
file = open('rows.bin', 'rb')  
table\_rows = pickle.load(file)  
  
  
my\_cursor.execute('Create database stationary\_manager')  
my\_cursor.execute('use stationary\_manager')  
my\_cursor.execute('create table items(ItemNo int primary key Auto\_increment,'  
 'Item\_name varchar(30),'  
 'Stock int,price float(10, 2),'  
 'Added\_On timestamp default current\_timestamp)')  
my\_cursor.execute('create table users(UserId int primary key,'  
 'UserName varchar(30),'  
 'First\_Logged\_in timestamp default current\_timestamp(),'  
 'Last\_Logged\_in timestamp default current\_timestamp())')  
  
query = 'insert into items values(%s, %s, %s, %s, %s)'  
my\_cursor.executemany(query, table\_rows)  
mydb.commit()  
print('Database imported successfully!')  
  
  
file.close()

**MAIN.PY**

import functions

program\_finished = False

main\_menu\_op = (('Enter 1 to display all Items',),

('Enter 2 to search Item details',),

('Enter 3 to add Items',),

('Enter 4 to modify Items',),

('Enter 5 to remove Items',),

('Enter 6 to manage users',),

('Enter 7 to exit',))

user\_op = (('Enter 1 to display all users',),

('Enter 2 to search users',),

('Enter 3 to change username',),

('Enter 4 to delete Items',),

('Enter 5 to go back to main menu',))

username = functions.log\_in()

while not program\_finished:

print(functions.table\_display(main\_menu\_op, ['Main Menu']))

choice = input('-> ')

while type(choice) is str:

if choice.isdigit():

if int(choice) in range(1, 8):

choice = int(choice)

else:

print('Enter a valid choice')

choice = input()

else:

print('Enter a valid choice')

choice = input()

if choice == 1:

print(functions.display\_items())

input('Press Enter to go to Main Menu')

elif choice == 2:

print(functions.search())

input('Press Enter to go to Main Menu')

elif choice == 3:

functions.add\_items()

input('Press Enter to go to Main Menu')

elif choice == 4:

functions.modify()

input('Press Enter to go to Main Menu')

elif choice == 5:

functions.remove()

input('Press Enter to go to Main Menu')

elif choice == 6:

managing\_users = True

while managing\_users:

print(functions.table\_display(user\_op, ['Manage Users']))

user\_choice = input('-> ')

if user\_choice == '1':

print(functions.see\_users())

input('Enter to go back')

elif user\_choice == '2':

print(functions.search\_user())

input('Enter to go back')

elif user\_choice == '3':

functions.change\_username()

input('Enter to go back')

elif user\_choice == '4':

functions.delete\_users()

input('Enter to go back')

elif user\_choice == '5':

managing\_users = False

elif choice == 7:

program\_finished = True

print('Program exited...')

functions.close(username)

**FUNCTIONS.PY**

import mysql.connector as ms

from prettytable import PrettyTable

from datetime import datetime

def get\_pswrd():

password = input('Enter your mysql server password: ')

return password

mydb = ms.connect(host='localhost', user='root', password=get\_pswrd(), database='stationary\_manager')

my\_cursor = mydb.cursor()

user\_atr = ('UserId', 'Username', 'FirstLogged-in', 'LastLogged-in')

item\_atr = ('ItemNo', 'Item\_name', 'Stock', 'price(in Rs)', 'Added\_On')

def table\_display(row, atr=item\_atr):

"""Displays tubular data in ASCII table format"""

table = PrettyTable()

table.field\_names = atr

table.add\_rows(row)

return table

def display\_items(query='select \* from items', atr=item\_atr):

"""Display data in tabular form just like in mysql"""

my\_cursor.execute(query)

rows = my\_cursor.fetchall()

if len(rows) == 0:

return rows

return table\_display(rows, atr)

def search():

# select search method

print('''Enter 1 to search by Item Name

Enter 2 to search by price

Enter 3 to search by date added''')

choice = input()

if choice == '1':

item\_name = input('Enter the product name\n')

query = f'select \* from items where Item\_name="{item\_name}"'

output = display\_items(query)

if type(output) is list:

return f"No item named '{item\_name}' is available"

else:

return output

elif choice == '2':

print('''Enter 1 to enter a specific price

Enter 2 to enter price range''')

choice = int(input())

if choice == 1:

price = input('Enter the price\n')

query = f'select \* from items where price = {price}'

output = display\_items(query)

if type(output) is list:

return f'No items of price {price}'

else:

return display\_items(query)

elif choice == 2:

min\_price = input("Enter starting price\n")

max\_price = input("Enter max price\n")

query = f'select \* from items where price>={min\_price} and price<={max\_price}'

return display\_items(query)

elif choice == '3':

print('''Enter 1 to search items added on a day

Enter 2 to search items added on range of days''')

date\_method = int(input())

if date\_method == 1:

year = int(input('Enter year: '))

month = int(input('Enter month: '))

date = int(input('Enter date: '))

date2 = datetime(year, month, date + 1)

date = datetime(year, month, date)

output = display\_items(f'select \* from items where added\_on >"{date}" and added\_on <"{date2}"')

elif date\_method == 2:

year\_1 = int(input('Enter year\_1: '))

month\_1 = int(input('Enter month\_1: '))

date\_1 = int(input('Enter date\_1: '))

year\_2 = int(input('Enter year\_2: '))

month\_2 = int(input('Enter month\_2: '))

date\_2 = int(input('Enter date\_2: '))

date1 = datetime(year\_1, month\_1, date\_1)

date2 = datetime(year\_2, month\_2, date\_2+1)

output = display\_items(f'select \* from items where added\_on >"{date1}" and added\_on <"{date2}"')

if type(output) is list:

return f"No items found..."

else:

return output

def add\_items():

items = int(input('How many items you want to add?\n'))

val = []

item\_names = []

modification = False

for n in range(items):

modification = False

if items > 1:

print(f'Enter item {n+1}')

else:

print('Enter Item-Name')

item\_name = input()

if item\_name.lower() in eval((str(get\_column(column\_no=2)).lower())):

print(f'{item\_name} already exists. Do you want to update instead? Enter "y" or "n"')

to\_update = input()

if to\_update == 'y':

modify(item\_name)

modification = True

else:

stock = int(input('Enter stock\n'))

price = float(input('Enter the price\n'))

val.append((item\_name, stock, price))

item\_names.append(item\_name)

if modification is False:

query = 'insert into items(Item\_name, stock, price) values(%s, %s, %s)'

my\_cursor.executemany(query, val)

mydb.commit()

if len(item\_names) == 1:

item\_names = f'("{item\_names[0]}")'

else:

item\_names = tuple(item\_names)

print('The following items were added.')

print(display\_items(f'select \* from items where item\_name in {item\_names}'))

def modify(item\_name=None):

to\_continue1 = True

is\_modified = False

while to\_continue1:

if item\_name is None:

item\_name = input('Enter the item name you want to modify\n')

else:

print(f'Updating {item\_name}...\n')

if item\_name.lower() not in eval(str(get\_column(column\_no=2)).lower()):

print(f'No item named {item\_name} was found')

print("Do you want to add items? Enter 'y' or 'n'")

choice = input()

if choice == 'y':

add\_items()

break

else:

break

to\_continue2 = True

while to\_continue2:

print('''Enter 1 to change the price

Enter 2 to add to stock

Enter 3 to remove from stock''')

choice = int(input())

if choice == 1:

new\_price = input('Enter new price in Rs\n')

my\_cursor.execute(f'update items set price={new\_price} where item\_name="{item\_name}"')

is\_modified = True

elif choice == 2:

add\_stock = int(input('Enter the number of new items you are adding\n'))

old\_stock = get\_column('items', 3, item\_name)

stock = old\_stock[0] + add\_stock

my\_cursor.execute(f'update items set stock={stock} where item\_name = "{item\_name}"')

is\_modified = True

elif choice == 3:

old\_stock = get\_column('items', 3, item\_name)

if old\_stock[0] == 0:

print('This item is out of stock')

elif old\_stock[0] != 0:

sub\_stock = int(input('Enter the number of items to be removed: '))

if old\_stock[0] < sub\_stock:

stock = 0

my\_cursor.execute(f'update items set stock={stock} where item\_name = "{item\_name}"')

else:

stock = old\_stock[0] - sub\_stock

my\_cursor.execute(f'update items set stock={stock} where item\_name = "{item\_name}"')

is\_modified = True

if is\_modified:

print("Modification done successfully...")

print(display\_items(f'select \* from items where Item\_name="{item\_name}"'), '\n\n')

mydb.commit()

print(f'Do you want continue modifying the item "{item\_name}"? Enter "y" or "n".')

to\_continue = input().lower()

if to\_continue == 'n':

to\_continue2 = False

print(f'Do you want modifications to other items? enter "y" or "n"')

to\_continue = input().lower()

if to\_continue == 'n':

to\_continue1 = False

def remove():

item\_name = input("Enter the item name to be removed\n")

print('The following item was deleted...')

display\_items(f'select \* from items where item\_name="{item\_name}"')

my\_cursor.execute(f'delete from items where item\_name="{item\_name}"')

def get\_column(table\_name='items', column\_no=1, item\_name=None):

"""Returns the list of values of the passed column"""

if item\_name is not None:

my\_cursor.execute(f'select \* from {table\_name} where item\_name = "{item\_name}"')

else:

my\_cursor.execute(f'select \* from {table\_name}')

column = []

for row in my\_cursor:

column.append(row[column\_no-1])

return column

def see\_users():

my\_cursor.execute('select \* from users')

rows = my\_cursor.fetchall()

return table\_display(rows, user\_atr)

def search\_user():

username = input('Enter username: ')

my\_cursor.execute(f'select \* from users where username="{username}"')

users = my\_cursor.fetchall()

if len(users) > 0:

return table\_display(users, user\_atr)

else:

return f'No user named "{username}" was found...'

def change\_username():

old\_name = input('Enter the username you want to change: ')

if old\_name.lower() in eval(str(get\_column('users', 2, )).lower()):

new\_name = input('Enter the new username: ').title()

my\_cursor.execute(f'update users set username = "{new\_name}" where username = "{old\_name}"')

print(display\_items(f'select \* from users where username="{new\_name}"', user\_atr))

mydb.commit()

else:

print(f'No user named "{old\_name}" was found...')

def delete\_users():

username = input('Enter the username you want to delete: ')

if username.lower() in eval(str(get\_column('users', 2, )).lower()):

print('Following user was deleted...')

print(display\_items(f'select \* from users where username="{username}"', user\_atr))

my\_cursor.execute(f'delete from users where username="{username}"')

mydb.commit()

else:

print(f'No user named "{username}" was found...')

def sign\_up():

print('Looks like you are new, please login\n')

name\_valid = False

full\_name = ''

while not name\_valid:

f\_name = input('Enter your first name\n').capitalize()

l\_name = input('Enter your last name\n').capitalize()

full\_name = f'{f\_name} {l\_name}'

if full\_name in get\_column('users', 2):

print('Username already exists!')

else:

name\_valid = True

no\_of\_users = len(get\_column('users'))

if no\_of\_users == 0:

user\_id = 1

else:

user\_id = no\_of\_users+1

print(f'Your user id is {user\_id}')

q = 'insert into users(UserId, UserName) values(%s, %s)'

val = (user\_id, full\_name)

my\_cursor.execute(q, val)

mydb.commit()

print(f'signed\_in successfully! Welcome {full\_name}')

return full\_name

def log\_in():

user = input('Enter your username\n').title()

while True:

if user in get\_column('users', 2):

print(f'Hello {user}')

return user

else:

sign\_up()

break

def log\_out(username):

my\_cursor.execute(f'update users set Last\_Logged\_in=current\_timestamp() where username = "{username}"')

mydb.commit()

def close(username):

log\_out(username)

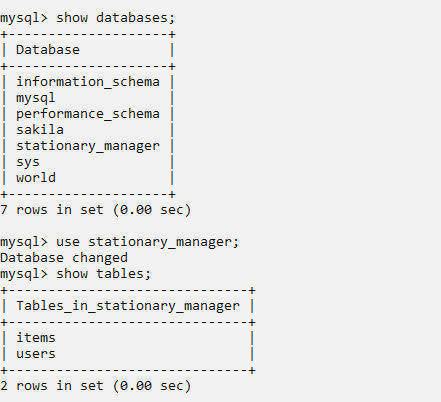
mydb.close()

**OUTPUT**

**Output of create\_database.py**

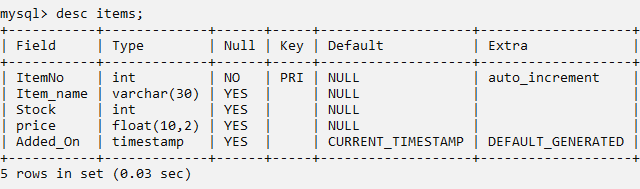
Enter your mysql server password: **students**

Database imported successfully!

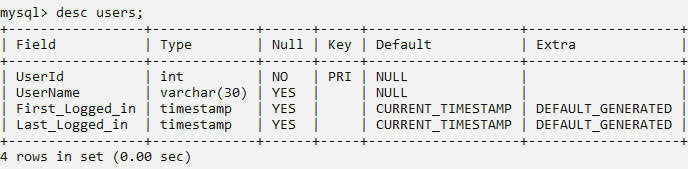


**Structure of tables**

Items table

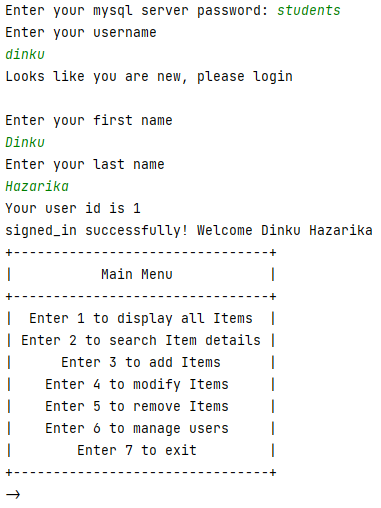


Users table

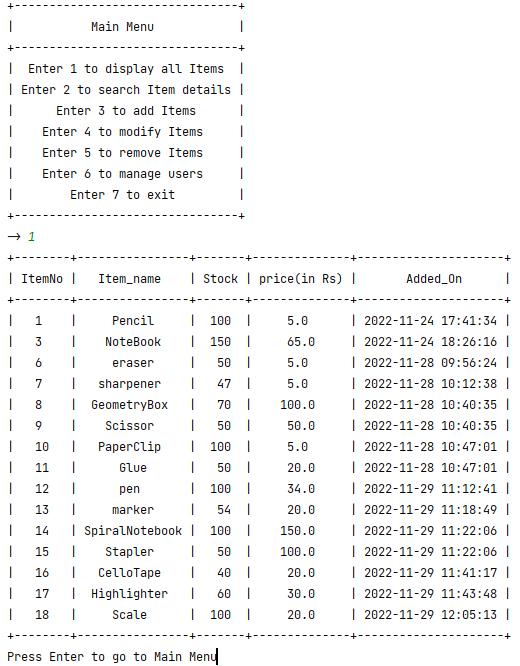


**Main Program**

Main menu

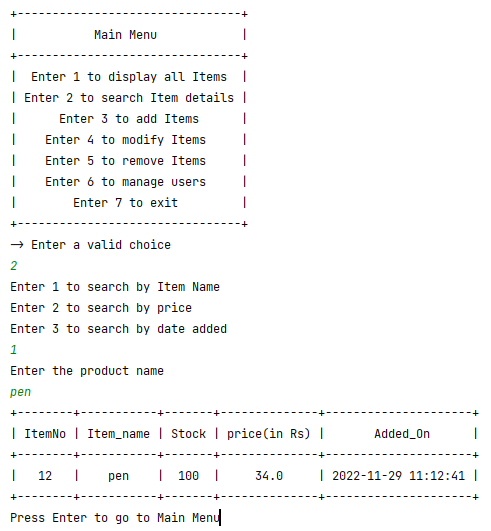


1. **Display all Items**

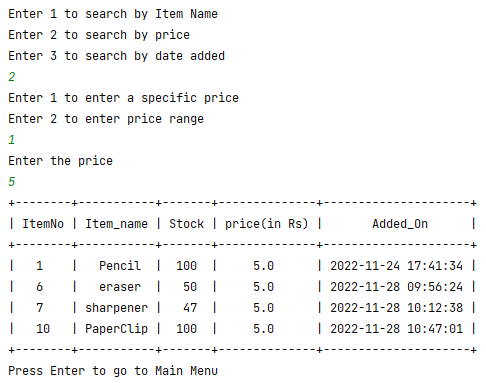


1. **Search Items**

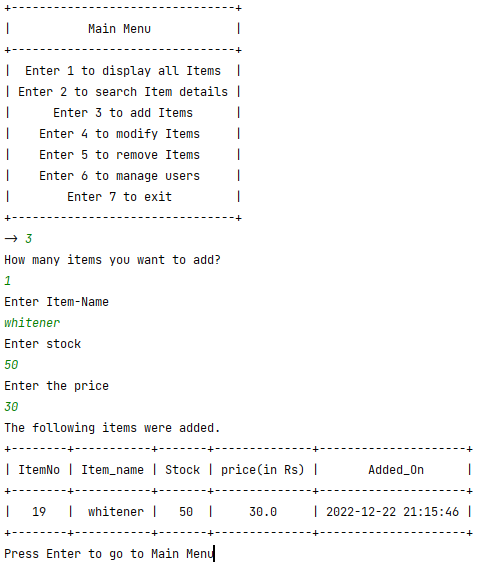
Search by name



Search by price

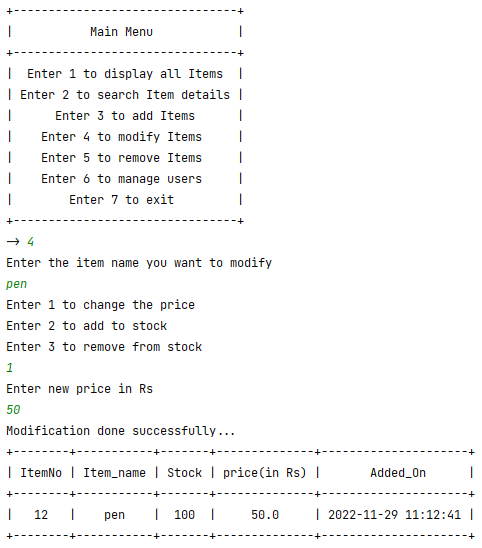


1. **Add Items**

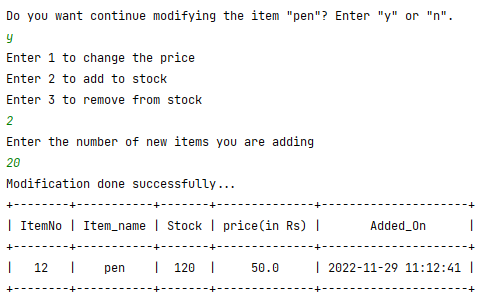


1. Modify Items

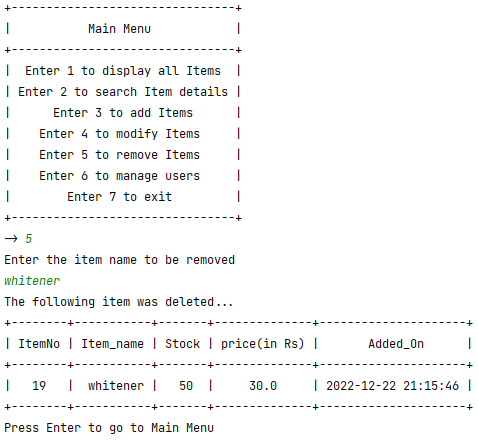
Change Price



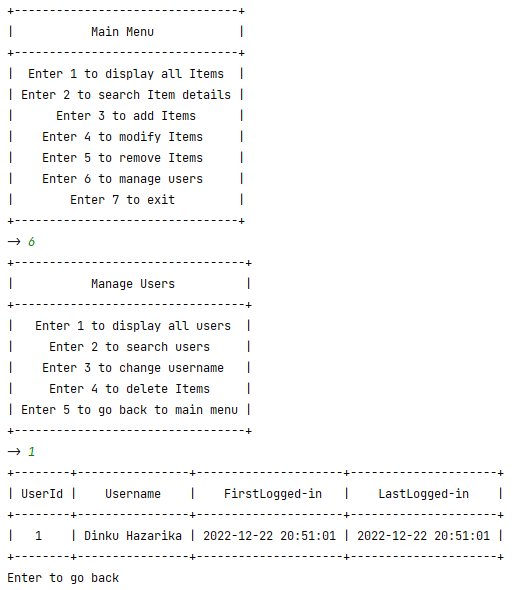
Add Stock

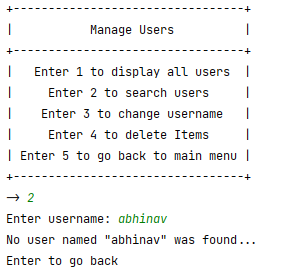


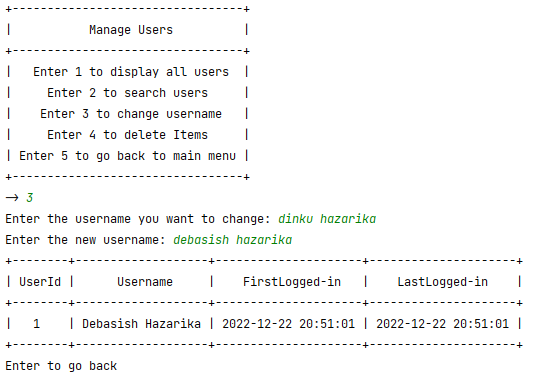
1. Remove Items

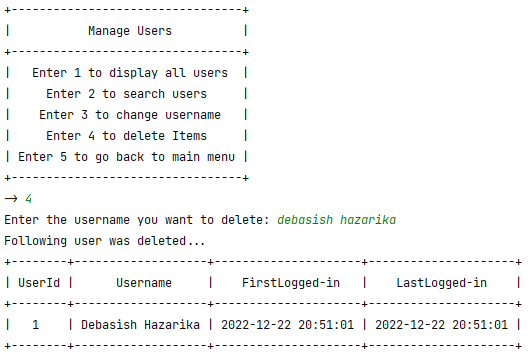


1. Managing Users









**BIBLIOGRAPHY**



