

## **E-Commerce Management System**

### **Submitted By:**

Hammad Asghar, 2021-CS-56

Waseem Yameen, 2021-CS-61

Abdullah Javed, 2021-CS-90

### **Submitted To:**

Sir. Samyan Qayum

**For Fulfillment of**

**DSA Lab**

**Department of Computer Science  
University of Engineering and Technology, Lahore**

## Abstract

Ecommerce Management System is a project which aims in developing an Application that provides a platform to purchase a distribution of products, or services through the internet or some other network. An E-commerce management system is a huge marketplace as most businesses are working on the internet. The data can be scrapped from AliExpress, EBay and DarazPk.

This project of ours is developed to facilitate people in buying products online and reduce human efforts.

## Acknowledgment

We would like to express our special Thanks to **Sir. Samyan** who gave us the golden opportunity to develop this project “Ecommerce Management System” and bring this idea to life. We came to learn so many new things while creating this project.

## Table of Contents

1. Introduction .....	5
1.1 Objective .....	5
2. Software Requirement Specification (SRS) .....	5
2.1 Product Functions .....	5
2.2 Safety Requirements .....	5
2.3 Security Requirements .....	5
3. Data Scraping .....	6
3.1 Attributes .....	6
3.1.1 Name .....	6
3.1.2 Category .....	6
3.1.3 Actual Price .....	6
3.1.4 Discounted Price .....	6
3.1.5 Sold Quantity .....	6
3.1.6 Reviews .....	6
3.1.7 Ratings .....	6
3.2 Scrapping Websites .....	7
3.2.1 Images of website .....	7
4. Sorting Algorithms .....	9
4.1 Insertion Sort .....	9
4.2 Merge Sort .....	10
4.3 Bubble Sort .....	11
4.4 Brick Sort .....	12
4.5 Bucket Sort .....	13
4.6 Counting Sort .....	14
4.7 Heap Sort .....	15
4.8 Pigeon Hole Sort .....	17
4.9 Selection Sort .....	18
4.10 Quick Sort .....	19
4.11 Radix Sort .....	20
4.12 Shell Sort .....	21
5. Searching Algorithms .....	22
5.1 Linear Search .....	22

5.2 Binary Search .....	23
5.3 Jump Search .....	24
5.4 Fibonacci Search .....	25
5.5 Interpolation Search .....	26
6. UI .....	28
6.1 Sign In Page Design .....	28
6.2 Sign Up Page Design .....	29
6.3 Main Screen Design .....	29
6.3.1 Add Items Menu .....	30
6.3.2 Delete Items Menu .....	30
6.4 Category Screen Design .....	31
6.5 Searching Screen Design .....	31
6.5.1 Multi Column Searching Menu .....	32
6.6 Items Screen Design .....	32
6.6.1 Price Range Menu .....	33
6.6.2 Multi Level Sorting Menu .....	33
6.6.3 Filter Search Menu .....	34
7. Limitations .....	34
8. Future Scope .....	34
9. Conclusion .....	35

## Table of Figures:

Figure 1 AliExpress Landing Page .....	7
Figure 2 EBay Landing Page .....	8
Figure 3 Daraz Landing Page .....	8
Figure 4 Sign In Page .....	28
Figure 5 Sign Up Page .....	29
Figure 6 Main Dashboard Page .....	29
Figure 7 Add New Items PopUp .....	30
Figure 8 Delete Items PopUp .....	30
Figure 9 Category Page .....	31
Figure 10 Searching Page .....	31
Figure 11 Multi Column Searching PopUp .....	32
Figure 12 View Items Page .....	32
Figure 13 Price Range PopUp .....	33
Figure 14 Multi Level Sorting PopUp .....	33
Figure 15 Searching Filter PopUp .....	34

# 1. Introduction

The best E-Commerce Platform that helps you to Purchase any product of your liking Online in minutes. This application is developed to reduce human efforts and provide user friendly experience. The thing which makes the E-commerce system powerful and reliable is the Rating and tenor of the Seller which is provided in Ecommerce management system. This system also helps seller to provide their products to right audience using our application.

## 1.1 Objective

- **Efficiency**  
This includes timeliness and accuracy of the data provided by given input.
- **Portability**  
This application should be portable for all devices at any environment
- **Security**  
The data provided by application should be secured using some authentication method. This can be done by signin/signup facility.

## 2. Software Requirement Specification (SRS)

A Software Requirement Specification is a document that includes complete description about how the application will perform on a system.

### 2.1 Product Functions

First it will authenticate the user so that unauthorized person can't get access to the application. The user will be able to customized the product list according to his needs whether it's related to specific category or price ranges.

### 2.2 Safety Requirements

All the data will be saved in local storage so there will be no data loss. This data can only be accessed by the authorized user. So data theft is not a possibility in this application.

### 2.3 Security Requirements

The application also have login feature to prevent any unauthorized access. You must have login credentials to access the application and it's features.

## 3. Data Scraping

### 3.1 Attributes

The items which were need to be scrapped contains some important attributes which were necessary for our data structure. The name and description of attributes are given below:

#### 3.1.1 Name

This include the title of the items page which was considered the name of the product.

#### 3.1.2 Category

This provides the category in which one items was placed.

#### 3.1.3 Actual Price

This tells the actual price of the product

#### 3.1.4 Discounted Price

This tells the discount or off sale value of the product.

#### 3.1.5 Sold Quantity

This provides the number of items sold.

#### 3.1.6 Reviews

This provides the number of reviews that a product page has.

#### 3.1.7 Ratings

This shows the star rating of a product.

## 3.2 Scrapping Websites

### 3.2.1 Images of website

AliExpress:

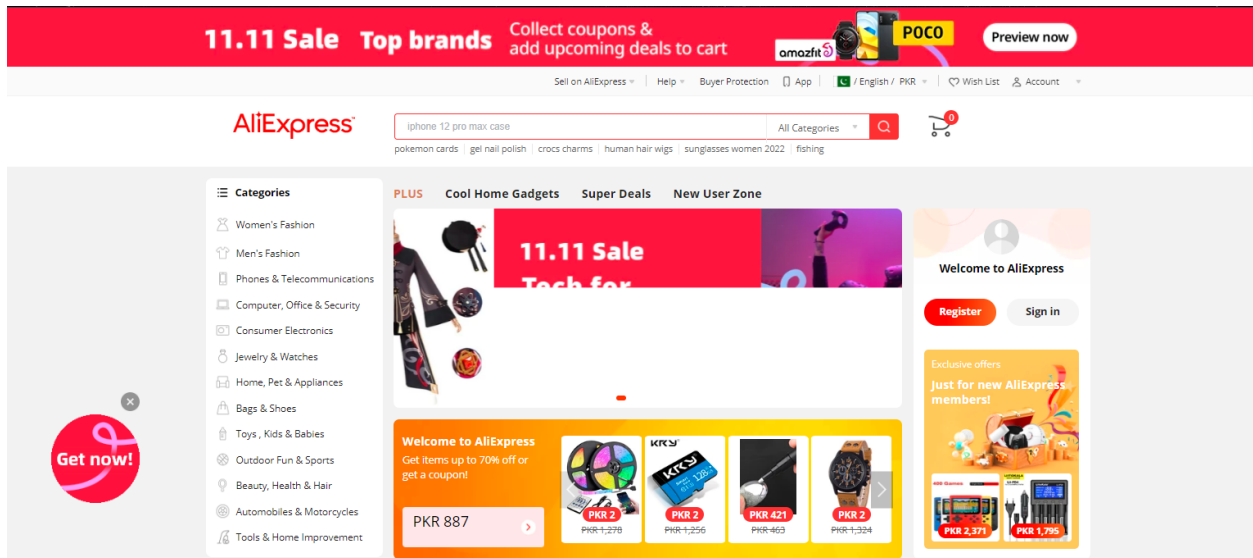


Figure 1 AliExpress Landing Page

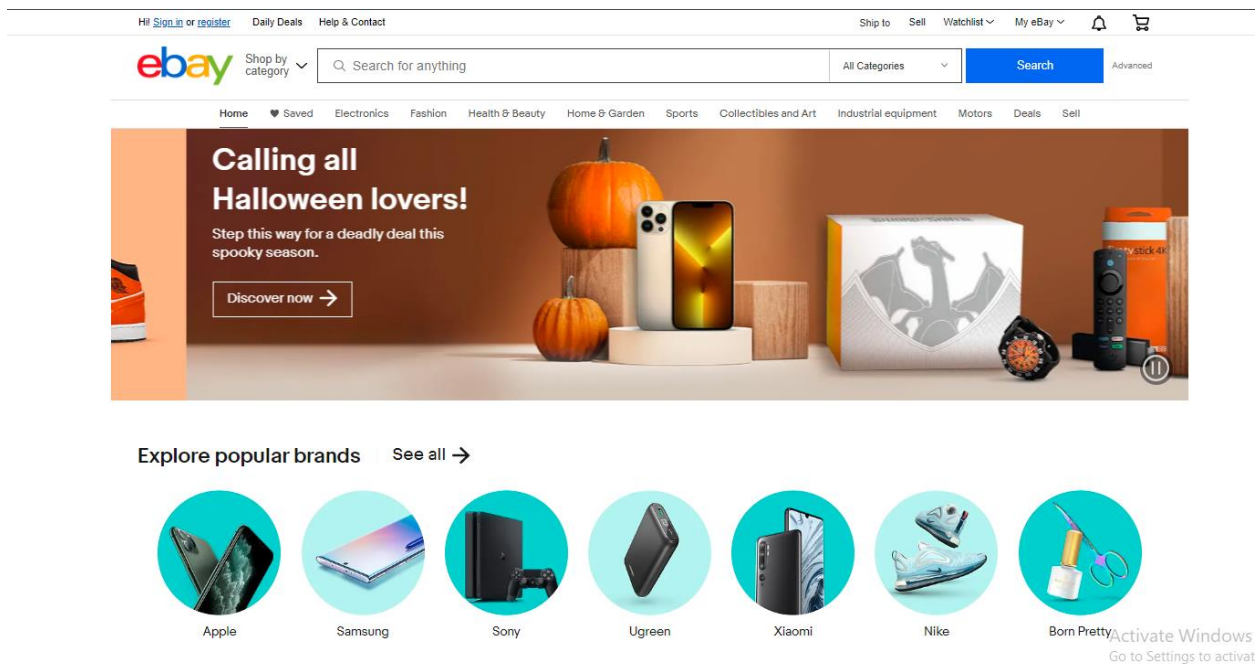
**EBay:**

Figure 2 EBay Landing Page

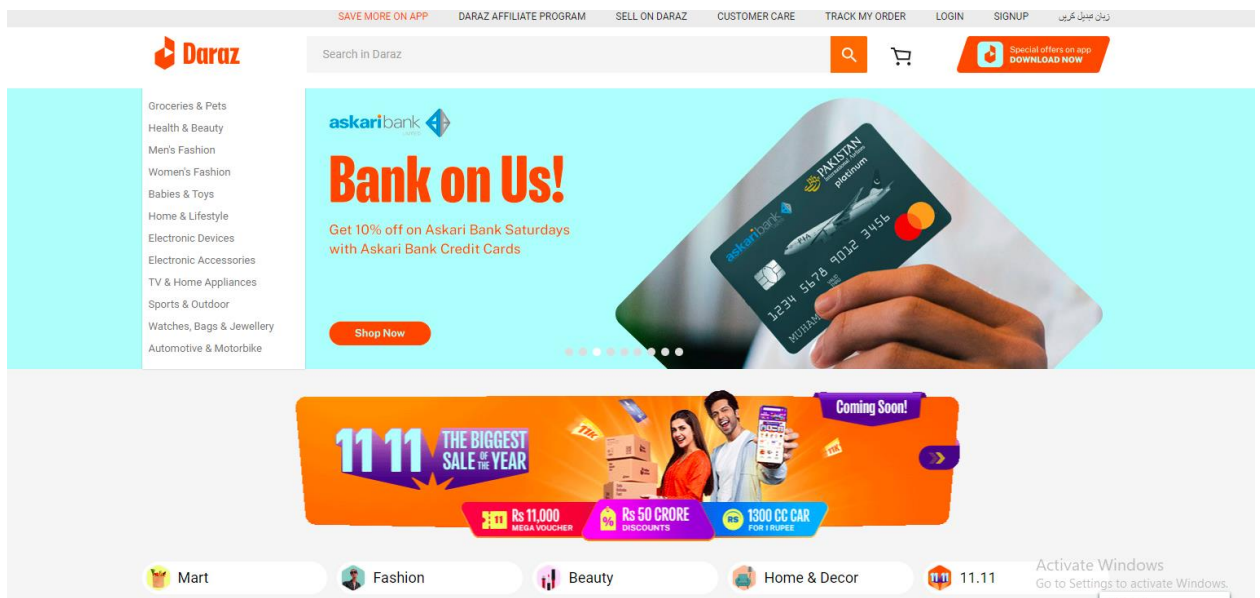
**Daraz:**

Figure 3 Daraz Landing Page



## 4. Sorting Algorithms

### 4.1 Insertion Sort

**Description:**

Insertion Sort algorithm picks an element and then compares it with the left elements if it is smaller than the left element it swaps and move an index behind and if it is greater the loop terminates.

**Time Complexity:**

$$O(n^2)$$

**Code:**

```
def InsertionSort_asc(array_2, rowNumber):  
    for i in range(1, len(array_2[rowNumber])):  
  
        key = getColumnFrom2D(array_2, i)  
        j = i - 1  
        while(j >= 0 and key[rowNumber] <= array_2[rowNumber][j]):  
  
            array_2 = exchangeColumns(array_2, j+1, j, rowNumber)  
            j = j - 1  
  
        exchangeObject(array_2, j+1, key)  
  
    return array_2
```

## 4.2 Merge Sort

### Description:

It works on Divide and Conquer Principle. Merge Sort divides the problem into sub-problems until the base case then compares the sub-problems and then combine the results.

### Time Complexity:

$O(n \cdot \lg n)$

### Code:

```
def Merge(array, p, q, r, rowNumber):
    leftArray = array[p:q+1]
    rightArray = array[q+1:r+1]
    if (isinstance(array[0][rowNumber], int) or isinstance(array[0][rowNumber], float)):
        leftArray.append([len(leftArray), sys.maxsize])
        rightArray.append([len(rightArray), sys.maxsize])
    else:
        maxlen=len(list(max([x[1] for x in leftArray]))) + 1
        leftArray.append([len(leftArray), ".".join([chr(255)]*maxlen)])
        rightArray.append([len(rightArray), ".".join([chr(255)]*maxlen)])
    leftArrayPointer = 0
    rightArrayPointer = 0
    i = p
    while (i <= r):
        if (leftArray[leftArrayPointer][rowNumber] <= rightArray[rightArrayPointer][rowNumber]):
            array[i] = leftArray[leftArrayPointer]
            leftArrayPointer += 1
        else:
            array[i] = rightArray[rightArrayPointer]
            rightArrayPointer += 1
        i += 1

def MergeSort(array, rowNumber, start, end):
    if (start < end):
        q = math.floor((start+end)/2)
        arr = MergeSort(array, 1, start, q)
        arr = MergeSort(array, 1, q+1, end)
        Merge(array, start, q, end, 1)
```

### 4.3 Bubble Sort

**Description:**

Bubble Sort algorithm compares two adjacent elements of the array if they are placed in the wrong order. In this way greater elements start compiling at the end of the array.

**Time Complexity:**

$O(n^2)$

**Code:**

```
def BubbleSort_asc(array,rowNumber):  
  
    for i in range(0,len(array[rowNumber])):  
  
        for j in range(len(array[rowNumber])-1,0,-1):  
  
            if(array[rowNumber][j] < array[rowNumber][j - 1]):  
  
                array = exchangeColumns(array, j, j-1, rowNumber)  
  
    return (array)
```

#### 4.4 Brick Sort

**Description:**

Brick sorts the element by odd transposition and then even transposition and so on until we get a sorted array.

**Time Complexity:**

$O(n^2)$

**Code:**

```
def BrickSort_asc(array,rowNumber):
    Flag = True
    while(Flag):
        Flag = False
        for i in range(0,len(array[rowNumber])-1,2):
            if(array[rowNumber][i] > array[rowNumber][i+1]):
                array = exchangeColumns(array, i, i+1, rowNumber)
                Flag = True

        for j in range(1,len(array[rowNumber])-1,2):
            if(array[rowNumber][j] > array[rowNumber][j+1]):
                array = exchangeColumns(array, j, j+1, rowNumber)
                Flag = True

    return array
```

## 4.5 Bucket Sort

### Description:

Bucket sort is a sorting algorithm that works by distributing the elements of an array into a number of buckets. Every element of the array is divided by the largest element and then sends it to respective bucket Each bucket is then sorted individually.

### Time Complexity:

$O(n)$

### Code:

```
def Bucketsort_asc(array,rowNumber):
    Buckets = []
    sortedarray = []
    n = 10
    for i in range(0,10):
        lis = []
        Buckets.append(lis)
        maximum = max(array[rowNumber]) + 1
        subarray = DivideElementsByMaximum(array[rowNumber],maximum)

        for j in range(0,len(array[rowNumber])):

            bucketnumber = int(n * subarray[j])
            array[rowNumber][j] = (array[rowNumber][j] * maximum)
            lis = getObject(array,j)
            Buckets[bucketnumber].append(lis)

    for k in range(0,len(Buckets)):
        L = Buckets[k]
        L = exchangeRowsandColumns(L)
        L = InsertionSort_asc(L,rowNumber)

        for m in range(0,len(L[0])):
            oneObject = exchangeRowsandColumns_2(L,m)
            sortedarray.append(oneObject)

    sortedarray = exchangeRowsandColumns(sortedarray)
    return sortedarray
```

## 4.6 Counting Sort

### Description:

Counting Sort algorithm does not compare elements. It picks the maximum element of the array and then declares an array of the same size. It then places the elements of first array in their respective position in second array.

### Time Complexity:

$O(n)$

### Code:

```
def countingSort_asc(arr,rowNumber):
    maximum = 100000
    indexes=[]
    for i in range(0,len(arr[0])):
        if(rowNumber == 6):
            indexes = exchange(arr, indexes, int(arr[rowNumber][i]*10),i)
        else:
            indexes = exchange(arr, indexes, int(arr[rowNumber][i]),i)

    indexes = [i for i in indexes if i != []]
    indexes = makeASingleList_asc(indexes)
    indexes = exchangeRowsandColumns(indexes)
    return indexes
```

## 4.7 Heap Sort

### Description:

Heap Sort is a comparison-based sorting algorithm it divides the array into a binary tree and then swaps the elements if they are placed in the wrong order.

### Time Complexity:

$$O(n \cdot \lg n)$$

### Code:

```
def Parent(i):
    return math.floor(i/2)

def Left(i):
    return 2*i

def Right(i):
    return 2*i + 1

def swapPositions(A, elem1, elem2):
    for i in range(0,7):
        (A[i][elem1], A[i][elem2]) = (A[i][elem2], A[i][elem1])

def MaxHeapify(A, i, n, rownumber):
    l = Left(i)
    r = Right(i)

    if (l <= n and (A[rownumber][l-1] > (A[rownumber][i-1])):
        largest = l
    else:
        largest = i
    if (r <= n and (A[rownumber][r-1] > (A[rownumber][largest-1])):
        largest = r
    if (largest != i):
        swapPositions(A, i-1, largest-1)
        MaxHeapify(A, largest, n, rownumber)
```

```
def BuildMaxHeap(A,Clue,rownumber):  
    n = len(A[rownumber])  
    for i in range(math.floor(len(A[rownumber])/2),0,-1):  
        if Clue:  
            MaxHeapify(A, i, n,rownumber)  
        else:  
            MinHeapify(A, i, n,rownumber)  
    return n  
  
def HeapSort_Asc(A,rownumber):  
    n = BuildMaxHeap(A,True,rownumber)  
    for i in range(len(A[rownumber]),1,-1):  
        swapPositions(A, 0, i-1)  
        n = n-1  
        MaxHeapify(A, 1, n,rownumber)  
    return A
```



## 4.8 Pigeon Hole Sort

### Description:

Pigeonhole sorts an element by creating nth empty array and inject one-by-one desired elements into the array.

### Time Complexity:

$O(n)$

### Code:

```
def PigeonHoleSort_asc(arr,rowNumber):
    if(rowNumber == 6):
        arr = MakeratingsInteger(arr)
        maximum = max(arr[rowNumber])
        minimum = min(arr[rowNumber])
        Range = maximum - minimum + 1

    sortedarraywithZeros = [[] for i in range(int(Range))]
    sortedarray = []

    for i in range(0,len(arr[rowNumber])):
        idx = arr[rowNumber][i] - minimum
        sortedarraywithZeros = storeCompleteObject(arr,rowNumber,i,idx,sortedarraywithZeros)

    sortedarray = [i for i in sortedarraywithZeros if len(i) > 0]
    sortedarray = makeASingleList_asc(sortedarray)
    sortedarray = exchangeRowsandColumns(sortedarray)
    if(rowNumber == 6):
        sortedarray = MakeratingsFloat(sortedarray)
    return sortedarray
```

## 4.9 Selection Sort

### Description:

Selection Sort picks the smallest element of the array and then place it at the beginning of the array.

### Time Complexity:

$$O(n^2)$$

### Code:

```
def SelectionSort_asc(array , rowNumber):  
  
    for i in range (0,len(array[rowNumber])):  
        minIndex = minimum(array, i,rowNumber)  
        array = exchangeColumns(array, minIndex, i, rowNumber)  
    return array  
  
def exchangeColumns(array, a, b, rowNumber):  
  
    for i in range(0,7):  
        c = array[i][a]  
        array[i][a] = array[i][b]  
        array[i][b] = c  
    return array  
  
def minimum(array , start, rowNumber):  
  
    minimumNumber = array[rowNumber][start]  
    minimumIndex = start  
  
    for i in range(start , len(array[rowNumber])):  
        if(minimumNumber > array[rowNumber][i]):  
            minimumNumber = array[rowNumber][i]  
            minimumIndex = i  
    return minimumIndex
```

## 4.10 Quick Sort

### Description:

Quick Sort algorithm also works on Divide and Conquer principle. It picks the last element as pivot and then compares it with the left elements. If the left element is greater, it places it towards right.

### Time Complexity:

$O(n \cdot \lg n)$

### Code:

```
def QuickSort_asc(array,rowNumber,left,right,mainList):
    if(left<right):
        pivot = Sort_asc(array,left,right,mainList)
        QuickSort_asc(array, rowNumber,left, pivot-1,mainList)
        QuickSort_asc(array, rowNumber,pivot+1, right,mainList)

def getsorted2DarraybyQuickSort_asc(mainList,rowNumber):
    QuickSort_asc(mainList[rowNumber], rowNumber, 0, len(mainList[0])-1,mainList)
    return mainList

def Sort_asc(subArray,left,right,mainList):
    pivot = subArray[right]
    i = left - 1

    for k in range(left,right):
        if(subArray[k] <= pivot):
            i = i + 1
            exchangeColumns(k, i, mainList)

    exchangeColumns(right, i+1, mainList)

    return i+1
```

### 4.11 Radix Sort

**Description:**

Radix Algorithm sorts element on the bases of the decimal place. First It will create buckets equals to the base of the number and then start send the number to the nth array is the LSB of every number.

**Time Complexity:**

$O(n)$

**Code:**

```
def RadixSort(arr):
    maximum = max(arr)
    length = len(str(maximum))
    for i in range(0, len(arr)):
        num = str(arr[i])
        num = num.zfill(length)
        arr[i] = num

    for j in range(length-1, -1, -1):
        arr = sortArrayUsingCountingSort(arr, j)
    return arr

def sortArrayUsingCountingSort(arr, radix):
    indexes1 = [0 for k in range(10)]
    indexes2 = [0 for k in range(10)]
    sortedarray = [0 for k in range(len(arr))]
    sum = 0

    for i in range(0, len(arr)):
        num = arr[i]
        num2 = int(num[radix])
        indexes1[num2] = indexes1[num2] + 1
    for i in range(0, 10):
        indexes2[i] = sum + indexes1[i]
        sum = sum + indexes1[i]
    for m in range(len(arr)-1, -1, -1):
        element = arr[m]
        elementsPart = int(element[radix])
        indexes2[elementsPart] = indexes2[elementsPart] - 1
        sortedarray[indexes2[elementsPart]] = arr[m]

    return sortedarray
```

## 4.12 Shell Sort

### Description:

Shell Sort algorithm sorts the elements in intervals and then sorts the array using insertion sort

### Time Complexity:

$$O(n \cdot \log n)$$

### Code:

```
def ShellSort_Asc(A,rownumber):
    n = math.floor(len(A[rownumber])/2)

    for i in range(n):
        while n>0:
            ShellInsertionSort(A,i,n,rownumber)

            n=math.floor(n/2)
    return A

def ShellInsertionSort(Arr,start,GapSize,rownumber):

    for i in range(start+GapSize,len(Arr[rownumber]),GapSize):
        current = []
        currentval = (Arr[rownumber])[i]
        ind = i
        for j in range(0,7):
            current.append((Arr[j])[i])

        while ind >= GapSize and (Arr[rownumber])[ind-GapSize] > currentval:
            (Arr[rownumber])[ind] = (Arr[rownumber])[ind-GapSize]
            exchange(Arr, ind, rownumber, GapSize)
            ind-=GapSize

        (Arr[rownumber])[ind] = currentval
        exchange2(Arr,ind,current,rownumber)
```

## 5. Searching Algorithms

### 5.1 Linear Search

**Description:**

It will search one by one from start to end of the array

**Time Complexity:**

$O(n)$

**Code:**

```
def Search(source,key,rowNumber):
    if(rowNumber == 0 or rowNumber == 1):
        if key in source:
            return True
    else:
        if(source == key):
            return True

def LinearSearch(data,rowNumber,key):
    dataColumnList = []
    searchedData = []
    for i in range(0,len(data[0])):
        if(Search(data[rowNumber][i],key,rowNumber)):
            dataColumnList.append(i)

    for k in range(0,len(dataColumnList)):
        lis = []
        for s in range(0,7):
            lis.append(data[s][dataColumnList[k]])

        searchedData.append(lis)
    Data = exchangeRowsandColumns(searchedData)
    return Data
```

## 5.2 Binary Search

**Description:**

It takes less time than linear search. Data should be sorted. It works on divide and conquer method. It divides the array in 2 equal sub arrays and continue doing this process recursively to array containing the element until the element is not found.

**Time Complexity:**

$O(\log n)$

**Code:**

```
def binarySearch(startingIndex, endingIndex, filterNo, findElement, arr):  
    if (startingIndex == endingIndex):  
        if (compareBySwitchingCondition(arr[startingIndex], findElement, filterNo)):  
            return startingIndex  
        else:  
            return -1  
    else:  
        q = math.floor((startingIndex + endingIndex)/2)  
        if (compareBySwitchingCondition(arr[q], findElement, filterNo)):  
            return q  
        else:  
            if (arr[q] > findElement):  
                return binarySearch(startingIndex, q-1, filterNo, findElement, arr)  
            elif (arr[q] < findElement):  
                return binarySearch(q+1, endingIndex, filterNo, findElement, arr)
```

### 5.3 Jump Search

**Description:**

Jump sort checks the elements by jumping in fixed steps if the element is greater, it jumps back and applies linear search

**Time Complexity:** $O(\sqrt{n})$ **Code:**

```
def Search(source,key,rowNumber):
    if(rowNumber == 0 or rowNumber == 1):
        if key in source:
            return True
    else:
        if(source == key):
            return True

def LinearSearch(data,rowNumber,key,startingidx,end):
    dataColumnList = []
    searchedData = []
    for i in range(startingidx,end):
        if(Search(data[rowNumber][i],key,rowNumber)):
            dataColumnList.append(i)

    for k in range(0,len(dataColumnList)):
        lis = []
        for s in range(0,7):
            lis.append(data[s][dataColumnList[k]])

        searchedData.append(lis)
    Data = exchangeRowsandColumns(searchedData)
    return Data
```



## 5.4 Fibonacci Search

### Description:

Data should be sorted. It works on divide and conquer method. It divides array into unequal parts. It uses Fibonacci numbers to search in a sorted array. It eliminates one side based on comparison.

### Time Complexity:

$O(\log n)$

### Code:

```
def GenerateNumbers(n):
    if n < 1:
        return 0
    elif n == 1:
        return 1
    else:
        return GenerateNumbers(n - 1) + GenerateNumbers(n - 2)

def FibonacciSearch_Forward(A, req, FilterNum):
    m = 0
    while GenerateNumbers(m) < len(A): #
        m = m + 1
    offset = -1
    while (GenerateNumbers(m) > 1):

        i = min( offset + GenerateNumbers(m - 2) , len(A) - 1)
        if compareBySwitchingCondition(str(A[i]), req, FilterNum):
            return i

        elif (req < A[i]):
            m = m - 2
        elif (req > A[i]):
            m = m - 1
        offset = i
    return None
```

## 5.5 Interpolation Search

### Description:

It is an improved version of binary search. It searches within the range of known points

### Time Complexity:

$O(\log n)$

### Code:

```
def interpolation_search_forNumbers(A, key):
    low = 0
    high = len(A) - 1
    key = math.floor(key)
    while math.floor(A[low]) <= key <= math.floor(A[high]) and math.floor(A[high]) != math.floor(A[low]):

        # Finds the closest or exact position of the element

        position = math.floor(low + ((key - math.floor(A[low])) * (high - low) / (math.floor(A[high]) -
math.floor(A[low]))))
        if math.floor(A[position]) == key:
            return position
        elif math.floor(A[position]) > key:
            high = position - 1
        else:
            low = position + 1
    if key == math.floor(A[low]):
        return low
    return None

#-----Returns All the index for Required Integers or Floats-----

def forNumbers(Arr, rownumber, Required):
    Arr = Arr[rownumber]
    ind = []
    offset = 0
    if(rownumber == 6):
        Required *= 10
    for i in range(len(Arr)):
        Arr[i] = Arr[i] * 10
```

```
for i in range(len(Arr)):
    try:
        a = interpolation_search_forNumbers(Arr[offset:], Required)
    except:
        a = None
    if a != None:
        if i == 0:
            actual = a
            offset = a
            ind.append(actual)
            offset += 1
            actual += 1
        else:
            break
    if rownumber == 6:
        for i in range(len(Arr)):
            Arr[i] = Arr[i] / 10
    return ind

def MainofInterpolation(Arr,rownumber,Required,FilterNum):

    if(rownumber == 0 or rownumber == 1):

        if(FilterNum == 0):

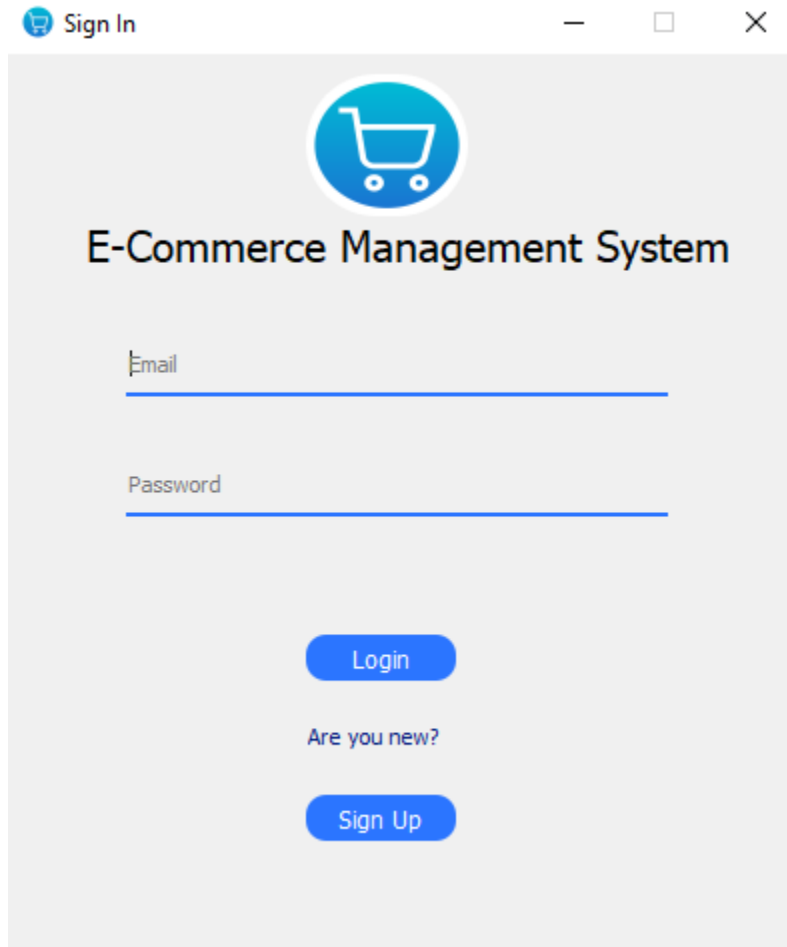
            return ChangelIdxToArray(forString(Arr, rownumber, Required), Arr)
        else:

            if(FilterNum == 2):

                return ChangelIdxToArray(forNumbers(Arr, rownumber, Required),Arr)
```

## 6. UI

### 6.1 Sign In Page Design

The image shows a desktop application window titled "Sign In" with a shopping cart icon. The window has standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top center is a blue circular icon with a white shopping cart. Below it, the text "E-Commerce Management System" is displayed in a bold, black, sans-serif font. Underneath the title, there are two input fields: the first is labeled "Email" and the second is labeled "Password". Both fields have a blue underline. Below the password field, there is a blue rounded rectangular button labeled "Login". Under the "Login" button, the text "Are you new?" is displayed in a smaller, gray font. At the bottom, there is another blue rounded rectangular button labeled "Sign Up".

Sign In

E-Commerce Management System

Email

Password

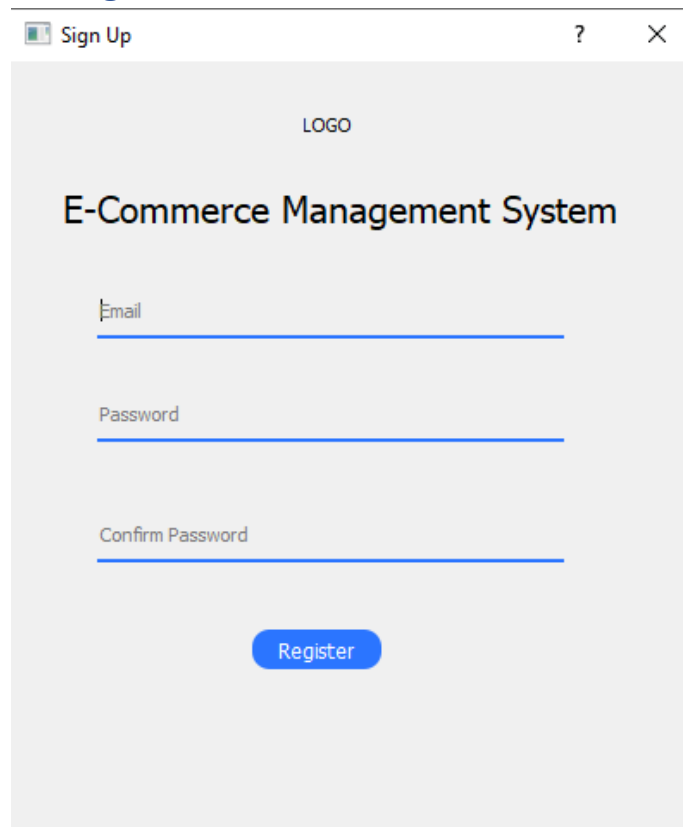
Login

Are you new?

Sign Up

Figure 4 Sign In Page

## 6.2 Sign Up Page Design



The image shows a desktop application window titled "Sign Up". Inside the window, there is a light gray background. At the top center, the word "LOGO" is displayed. Below it, the title "E-Commerce Management System" is centered. There are three input fields: "Email", "Password", and "Confirm Password", each with a blue underline. Below these fields is a blue rounded button labeled "Register".

Figure 5 Sign Up Page

## 6.3 Main Screen Design

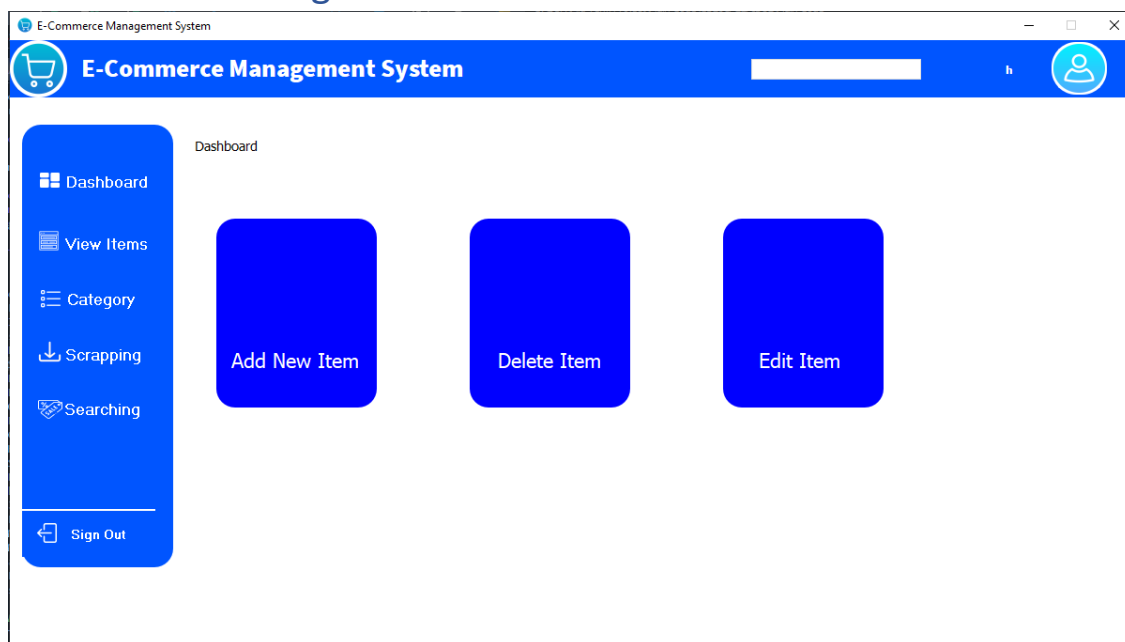


Figure 6 Main Dashboard Page

## 6.3.1 Add Items Menu

**ADD NEW ENTITY**

Name: Coldeez

Type: Medicine      Discounted Price: 1200

Price: 1100      Sold Qty: 1070

Review: 107      Ratings: 5.0

**OK**

Figure 7 Add New Items PopUp

## 6.3.2 Delete Items Menu

**DELETE**

eBay      Type      Start With      **Display**

2      **Delete**

	Name	Type	Price	Disc	Sold	Review...	Ratings
1							
2	4 New ...	eBay ...	109040.0	109040.0	0	0	0.0
3	4 Tires ...	eBay ...	39517.459	39517.459	0	25	5.0
4	4 Tires ...	eBay ...	39244.859	39244.859	0	51	4.9
5	4 NEW ...	eBay ...	52889.852	52889.852	0	0	0.0
6	4 New ...	eBay ...	73051.348	73051.348	0	9	5.0
7	4 New ...	eBay ...	113941.348	113941.348	0	38	4.7
8	4 New ...	eBay ...	61062.4	61062.4	0	0	0.0
9	4 NEW ...	eBay ...	135754.8	135754.8	0	0	0.0
10	2 New ...	eBay ...	42107.159	42107.159	0	13	4.9
11	4 New ...	eBay ...	127026.148	127026.148	0	18	4.7
12	4 New ...	eBay ...	123209.748	123209.748	0	14	4.7

Figure 8 Delete Items PopUp

## 6.4 Category Screen Design

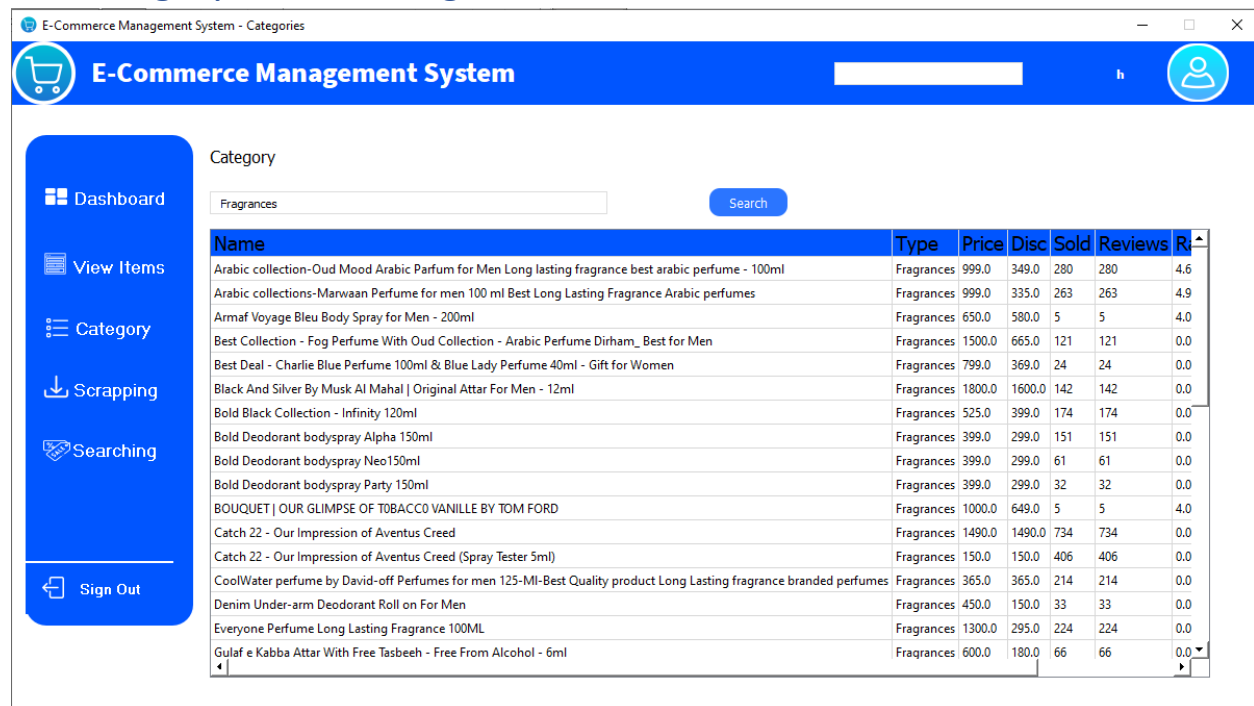


Figure 9 Category Page

## 6.5 Searching Screen Design

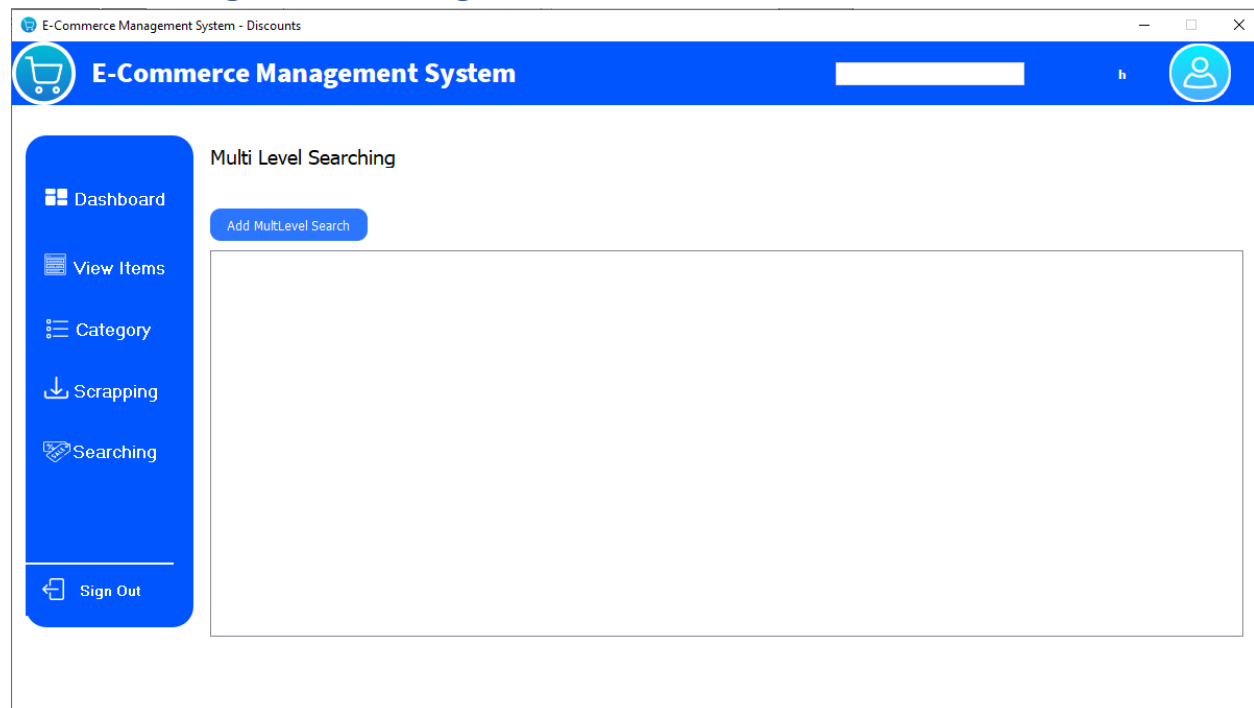


Figure 10 Searching Page

### 6.5.1 Multi Column Searching Menu

Multi Column Search

and Ratings  Add

operator	Fields	Search
	Name	Bread
or	Type	Beverages
not	Disc	1000.0
and	Ratings	4.7

Search

Figure 11 Multi Column Searching PopUp

## 6.6 Items Screen Design

E-Commerce Management System - View Items

**E-Commerce Management System**

View Items

Time (ms)

Insertion Sort Name

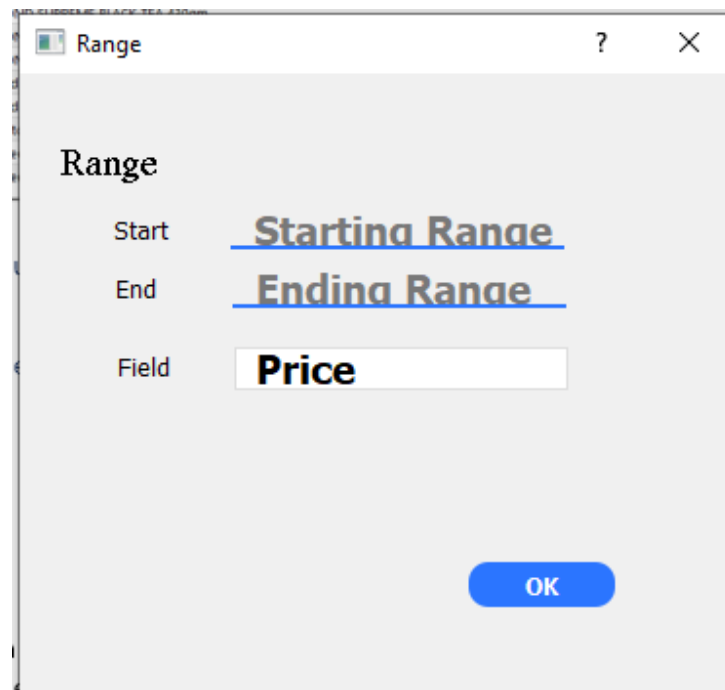
Sort Range Multi Level Sortnig Search

Name
ham
*Special Drink* (Pack of 2 Boxes) - Star Bulbulay Fizzy Drink Limka Flavor (24 Sachets)
3 Packs of Biah T. Tox Diet Tea Offer
3 Packs of Biah T. Tox Diet Tea Offer
7Up 500ml - Pack of 12
7Up 500ml - Pack of 12
Anchor Green Tea Pack (25 T-Bags) Combination of 10 organic Ingredients   100% Natural
Aquafina 1.5 L
Aquafina 1.5 L
BROOKE BOND SUPREME BLACK TEA 430gm
BROOKE BOND SUPREME BLACK TEA 85gm
BROOKE BOND SUPREME BLACK TEA 85gm
Brooke Bond Supreme Tea 190Gm
Brooke Bond Supreme Tea 190Gm
canada dry tonic water 300ml (imported & original)
Classic Coffee Sachets 2gm(1 Polybags of 60 Sachets)
Classic Coffee Sachets 2gm(1 Polybags of 60 Sachets)

Figure 12 View Items Page



## 6.6.1 Price Range Menu



A dialog box titled "Range" with a close button (X) and a help button (?). It contains three labels: "Start", "End", and "Field". The "Start" label is next to a text field containing "Starting Range". The "End" label is next to a text field containing "Ending Range". The "Field" label is next to a text field containing "Price". At the bottom right is a blue "OK" button.

Figure 13 Price Range PopUp

## 6.6.2 Multi Level Sorting Menu



A dialog box titled "Multi Level Sorting" with a close button (X) and a help button (?). It has a blue header bar with the text "MultiLevel Sorting". On the left, there are three dropdown menus: the first is empty, the second is set to "Insertion Sort", and the third is set to "Asc.". Below these are two blue buttons: "Add New Level" and "Delete Previous Level". On the right, there is a list box with a header "1" and a list of items: "Order", "Disc", "Sold", "Ratings", "Type", "Reviews", "Price", and "Name". The "Order" item is highlighted in blue. At the bottom right is a blue "Ok" button.

Figure 14 Multi Level Sorting PopUp

### 6.6.3 Filter Search Menu

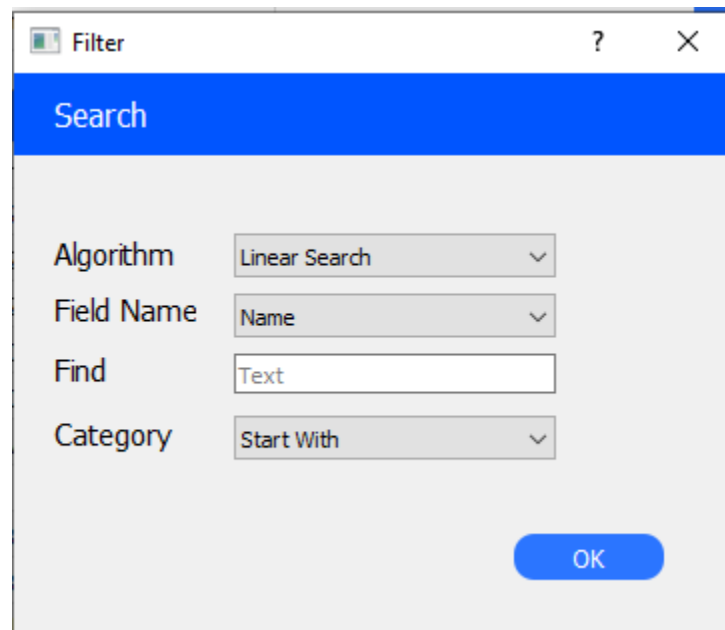


Figure 15 Searching Filter PopUp

## 7. Limitations

- **Interpolation search** only works with numbers and strings first characters.  
**Reason:** It's uses a mathematical formula to find the middle point of array which isn't compatible with list of strings. It will get very complex.
- **Counting sort** does not work for string attributes.
- **Radix sort** does not work for string attributes.
- **Bucket sort** does not work for string attributes.  
**Reason 1:** Linear time sorting algorithms doesn't work for string attributes.  
**Reason 2:** It consider two different strings having same ascii as same strings.

## 8. Future Scope

As by increasing some of the coding we can improve it's functionality. Online Payment system is yet not integrated to the system which can be featured in the near future. As the technology advances, it is possible to upgrade the system and can be adaptable to desired environment.

New security features can be added using new technologies.

## 9. Conclusion

The Ecommerce Management system has been successfully completed and also tested. It is user friendly and has most of the features that a basic user will need.

This application is developed using Python as a backend Logics, algorithms & PyQt5 as front end in windows environment.

The Goals that are achieved by the software:

- User Friendly.
- Portable.
- Less processing time in gathering and showing information to user.
- Secured from unauthorized access.