Project On

QuickSearch

Developed by

Name : *Zahra Arshadi*

Reg.No: *(R113021600046)*

Faculty: *Mr. Zareei*

NIIT
Tehran West Center
30216

QuickSearch

(Project Title)

Batch Code: *B110019*

Name of the Coordinator: *Mr. Zareei*

Name of Developers: *Zahra Arshadi*

NIIT

CERTIFICATE

This is the certify that this report, titled QuickSearch, embodies the original work done by Zahra Arshadi in partial fulfillment of his course requirement at NIIT.

Coordinator:

Mr. Zareei

ACKNOWLEDGMENT

I have benefited a lot from the feedback and suggestion given to us by Mr.zareei and other faculty members.

System Analysis

System summary:

RedSky manufactures a total of five products, each with a different product ID. The Sales department of RedSky maintains the sales data for each product on a separate text file. Each record in each file contains two fields,city and sales_figure.

The number of records in each file is very high. As a result, the employees of the sales department waste a lot of time in searching for sales information.

This utility needs to provide the following features :

Create index: this option will allow a user to create an index on a text file. For creating an index, the user will be required to specify the name of the file on which the index needs to be created and the name of the index file. If an index has already been created on a file, a user should not be required to create it again the next time an operation is to be performed on the file.

Load file: this option will allow a user to open a text file. All further operations will be performed on the file that is currently open. For loading a file, the user needs to specify the name of the file.

Insert records: this option will allow a user to insert records into the file that is currently open. For all newly-inserted records, the index should be automatically updated.

Search records: this option will allow a user to search for the record of a particular city in the currently loaded file. To search for a record, the user needs to specify the name of a city. The search operation should work with an efficiency of O(logn).

Implementation Details

We have got 5 products in this project so that their information is available in the 5 of text file which has two fields with the names as follow: (city,sales_figure)

1. For the performing search in the records and in order to decrease the search time, program will let the user to create an index for each text file. In the case of existence of the index file, program will pop up an alert to the user with this message: index file is exists!
2. With the help of combo boxes user can choose a product from combo box and preview the information file associated with the chosen product. Then user can insert a record with the help of available text boxes and appropriate index file will be updated synchronically.
3. Finally, search function has been programmed based on the name of cities and information about the sales will be prompted based on the selected city.

Individual Project Schedule

S. No.	Activity description	Planned Date of Completion	Actual Date	Responsibility	Remark
1	Writing code for implementing Index	17 th september	23 th	Zahra	
2	Writing code for read of text file and write it	19 th semtember	25 th	Zahra	
3	Writing code for search	20 th sepember	25 th	Zahra	

Program Coding

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace QuickSearch
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void CmbRead_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                string Product = CmbRead.Text;
                StreamReader str = new StreamReader("D:\\SaleInformation\\" +
Product + ".txt");
                TxtReadText.Text = str.ReadToEnd();
                str.Close();
            }
            catch (Exception ex)
            {

                throw;
            }

        }

        private void BtnWrite_Click(object sender, EventArgs e)
        {
            string Product = CmbRead.Text;
            StreamWriter str = new StreamWriter("D:\\SaleInformation\\" + Product
+ ".txt", true);
            str.WriteLine(TxtUserText.Text);
            str.Close();
            CmbRead_SelectedIndexChanged(null, null);
            DirectoryInfo Dio = new DirectoryInfo("D:\\SaleInformation\\");
            FileInfo F = Dio.GetFiles().ToArray().Where(f =>
f.Name.StartsWith(Product)).First();
            CreateIndex(F.FullName);

        }
```

Program Coding

```
private void BtnCreateIndex_Click(object sender, EventArgs e)
        {
            string IndexPath = "D:\\SaleInformation\\" +CmbRead.Text +
TxtIndexFile.Text + ".idx";
            if (File.Exists(IndexPath))
            {
                MessageBox.Show("The Index Exists.");
                return;

            }
            CreateIndex(IndexPath);

        }

        private void CreateIndex(string path)
        {
            StreamReader str = null;
            if (CmbRead.Text != "")
            {
                str = new StreamReader("D:\\SaleInformation\\" + CmbRead.Text +
".txt");
            }

            else
            {
                return;
            }
            HashSet<string> SoldedProduct = new HashSet<string>();

            int index = 0;
            while (str.Peek() != -1)
            {
                //Tehran,90
                SoldedProduct.Add(str.ReadLine().Split(',')[0].ToString() + "," +
index.ToString());
                index++;
            }
            str.Close();
            StreamWriter stw = new StreamWriter(path);
            var CityIndex = SoldedProduct.OrderBy(o => o.ToString());
            var Cities = from C in CityIndex select new { name =
C.ToString().Split(',')[0].ToString() };
            var NameList = Cities.Distinct().ToArray().Distinct();
            var CityIndexList = CityIndex.ToList();
            string Indexvalue = "";
            foreach (var item in NameList)
            {
                string cityrecords = item.name + ":";
                for (int i = 0; i < CityIndexList.Count; i++)
                {
```

Program Coding

```
if (item.name == CityIndexList[i].Split(',')[0])
                    {
                        cityrecords += CityIndexList[i].Split(',')[1] + ",";
                    }
}
                Indexvalue += cityrecords + "#";
            }
            stw.WriteLine(Indexvalue);
            stw.Close();
        }

        private void BtnSearch_Click(object sender, EventArgs e)
        {
            DirectoryInfo Dio = new DirectoryInfo("D:\\SaleInformation\\");

            var indexfile = Dio.GetFiles().Where(s =>
s.Name.StartsWith(CmbSearch.Text));

            if (indexfile.Count() > 0)
            {
                SearchwithIndex(TxtSearchValue.Text, indexfile.First().FullName);
            }
            else
            {
                Search(CmbSearch.Text, TxtSearchValue.Text);
            }
        }

        private void Search(string Product, string Search)
        {
            StreamReader str = new StreamReader("D:\\SaleInformation\\" + Product
+ ".txt");
            LstResault.Items.Clear();
            while (str.Peek() != -1)
            {
                string linetext = str.ReadLine();
                if (linetext.StartsWith(Search))
                {
                    LstResault.Items.Add(linetext);
                }
            }
        }
```

Program Coding

```
void SearchwithIndex(string searchvalue, string indexfile)
        {
            try
            {
                StreamReader str = new StreamReader(indexfile);
                string[] cities = str.ReadToEnd().Split('#');
                str.Close();
                string[] recordno = null;

                int count = cities.Where(s => s.StartsWith(searchvalue)).Count();
                if (count == 0)
                {
                    MessageBox.Show("City None Exists.");
                    return;
                }
                for (int i = 0; i < cities.Length; i++)
                {
                    if (cities[i].StartsWith(searchvalue))
                    {
                        string s = searchvalue + ":";
                        recordno = cities[i].Substring(s.Length).Split(',');
                        break;
                    }
                }
                str = new StreamReader("D:\\SaleInformation\\" + CmbSearch.Text +
".txt");
                LstResault.Items.Clear();
                int index = 0;
                while (str.Peek() != -1)
                {
                    string s = str.ReadLine();
                    for (int i = 0; i < recordno.Length - 1; i++)
                    {
                        if (int.Parse(recordno[i]) == index)
                        {
                            LstResault.Items.Add(s);
                        }
                    }
                    index++;
                }
                str.Close();
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }


        }
    }
}
```

Relationship Between Final Tables

Hardware: PC compatible with an Intel(R) Core(TM)2 Duo CPU2.53 GHz , 320 GB of hard disk , 4 GB of RAM, DVD/CD_ ROM Drivers

Operating System: Microsoft Seven(64 bit)

Software: visual studio