

# תכנות מונחה עצמים ב-C# - GUI Final

עומר כהן- תז. 208715813

טל סטולר תז. 313131880

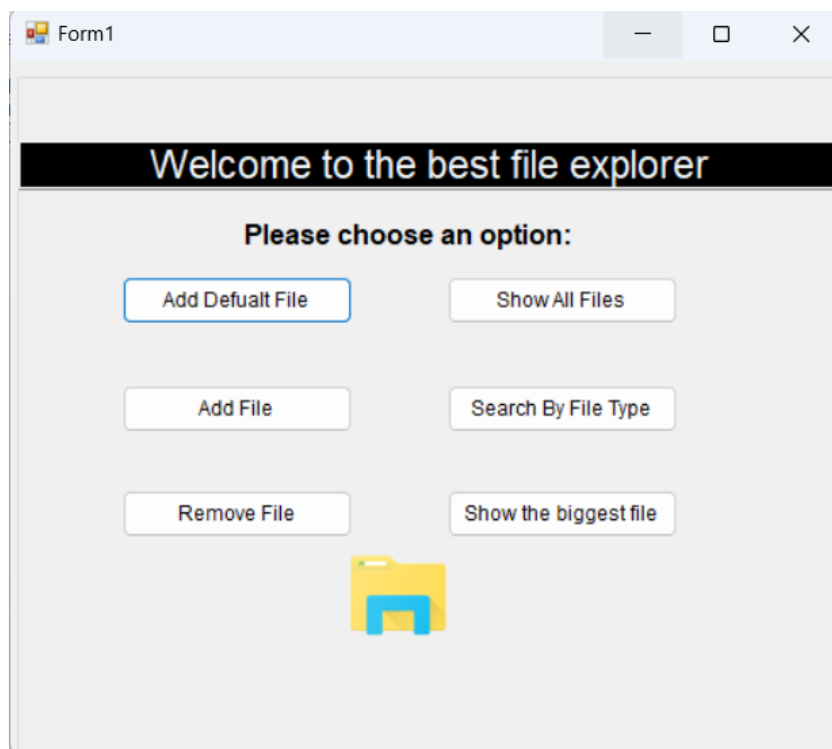
## שאלה 1:

\*יש לציין כי ההערות שניתנו במטלה 3 תוקנו.

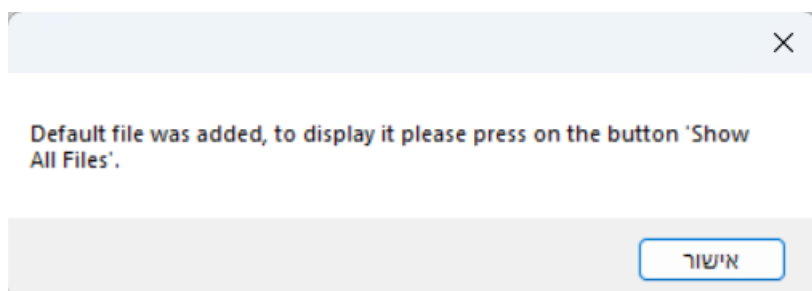
בשאלה זו מימשנו את הקוד ממטלה 3.

בחרנו לעצב את הממשק בצורה כזאת שלמשתמש יהיה נוח לגשת לכל אופציה בסייר הקבצים.

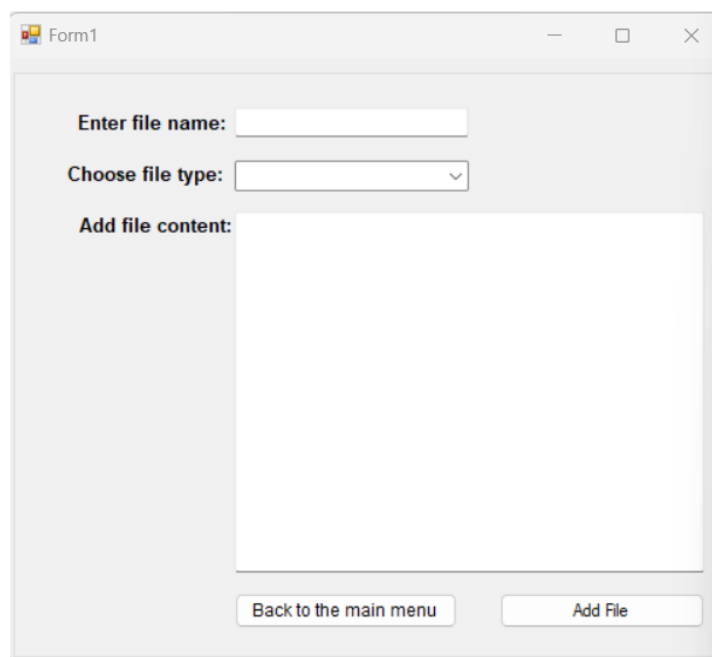
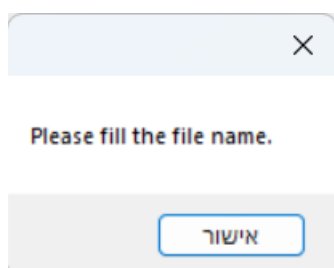
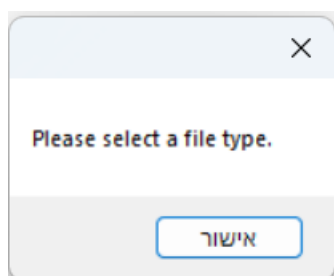
מסך הפתיחה:



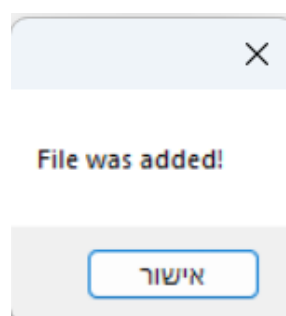
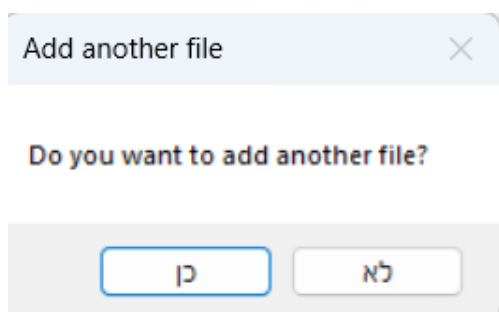
כאשר המשתמש לוחץ על המקש "Add Default File" יתווסף קובץ דיפולטיבי והמשתמש יקבל הודעה:



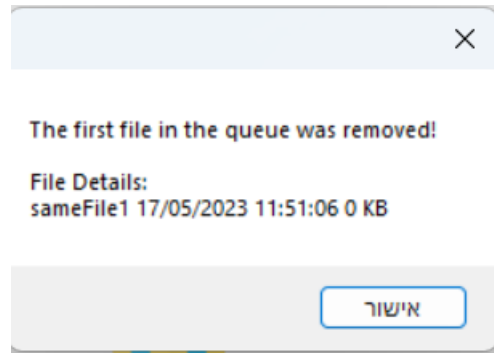
כאשר המשתמש לוחץ על המקש "Add File" יפתח מסך ובו על המשתמש להזין את כל פרטי הקובץ. יהיה עליו לבחור מתוך רשימה של הסיימות האפשריות (המימוש של ENUM), ושמו. במידה ולא יעשה את אחד מהאופציות יקבל הודעת שגיאה מתאימה לכל אי מילוי:



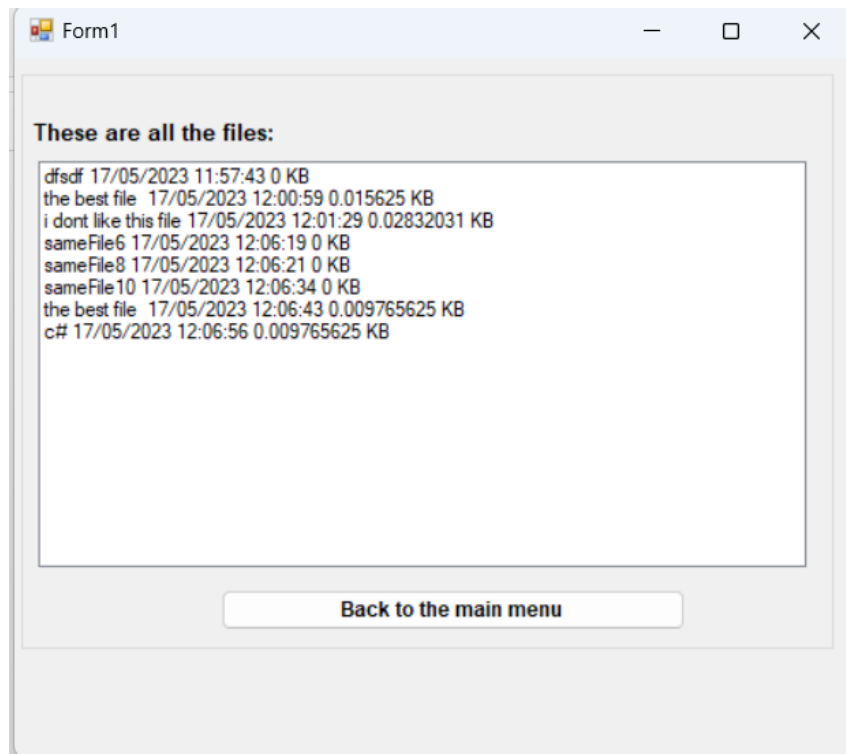
במידה וימלא הכל כשורה ולאחר מכן ילחץ על מקש "Add File" יקבל הודעה כי הקובץ הוסף והודעה במידה וירצה להוסיף קובץ נוסף. במידה ויקיש כן, ישאר במסך במידה ולא יוחזר למסך הפתיחה:



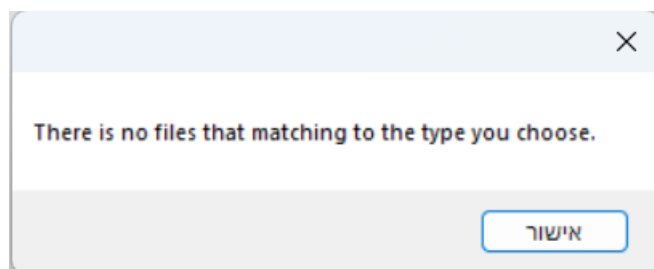
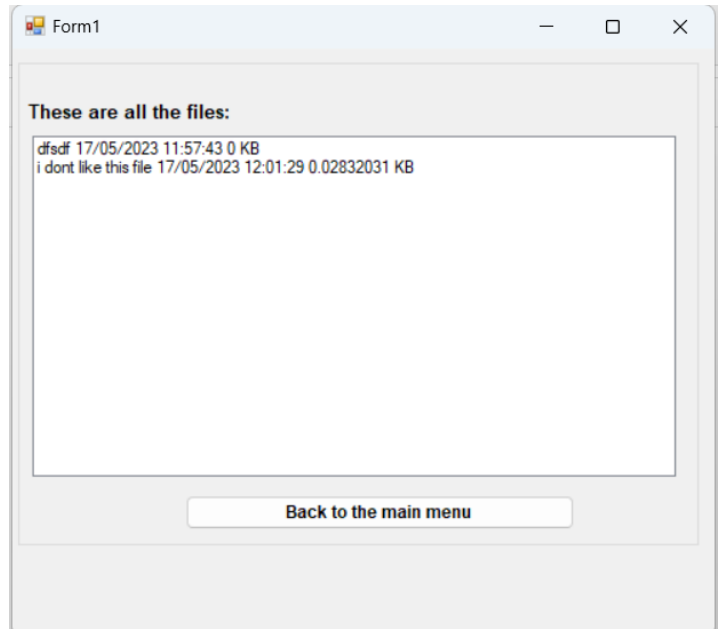
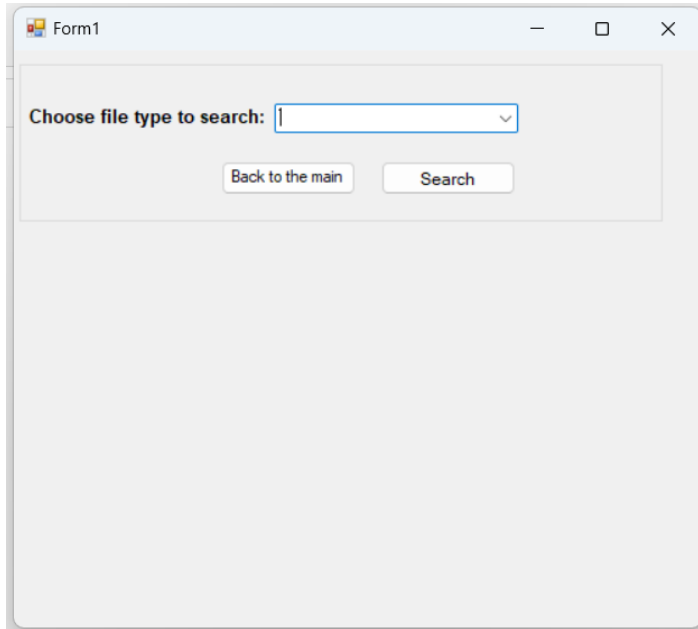
כאשר המשתמש לוחץ על המקש "Remove File" ימחק הקובץ שהוסף ראשון והמשתמש יקבל הודעה עם פרטי הקובץ:



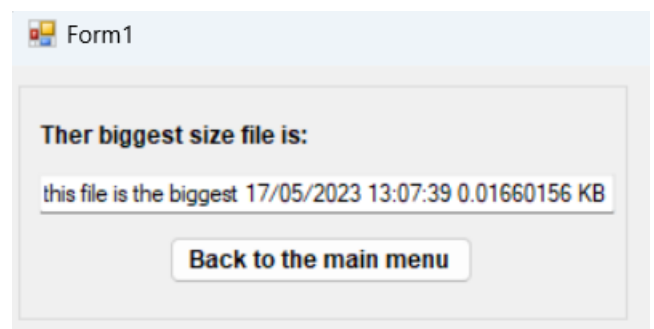
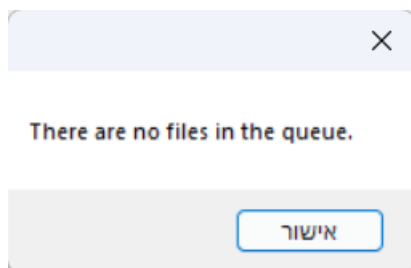
כאשר המשתמש לוחץ על המקש "Show All Files" יוצגו לו כל הקבצים הנמצאים במערך:



כאשר המשתמש לוחץ על המקש "**Search By File Type**" יפתח חלון ובו על המשתמש יהיה לבחור סיומת קובץ אותו הוא מעוניין לחפש. במידה ולא קיימים קבצים כאלו יקבל הודעה מתאימה, במידה וכן יוחזרו הקבצים הרלוונטיים:



כאשר המשתמש לוחץ על המקש "**Show The Biggest File**" יוצג למשתמש הקובץ בעל המשקל הגבוה ביותר ופרטיו במידה ולא קיימים קבצים במערכת יקבל הודעה מתאימה:



## קוד התכנית:

### Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace GUIfinalQ1
{
    public partial class Form1 : Form
    {
        private QueueFiles files;
        private QueueFiles queueFiles = new QueueFiles();
        private FileTypeExtension selectedFileType;

        public Form1()
        {
            InitializeComponent();
        }

        private void btnAddDefault_Click(object sender, EventArgs e)
        {
            DataFile dataFile = new DataFile();
            queueFiles.Enqueue(dataFile);
            UpdateFileList();
            MessageBox.Show("Default file was added, to display it please  
press on the button 'Show All Files'.");
            HideAllGroupBoxes();
            groupBox1.Visible = true;
        }

        private void btnAddFile_Click(object sender, EventArgs e)
        {
            HideAllGroupBoxes();
            groupBoxAddFile.Visible = true;
            comboBoxTypeAdd.SelectedIndex = -1;
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            if (queueFiles.IsEmpty())
            {
                MessageBox.Show("No files to remove.");
            }
            else
            {
                DataFile removedFile = queueFiles.Dequeue();
                UpdateFileList();
            }
        }
    }
}
```

```

        MessageBox.Show("The first file in the queue was
removed!\n\nFile Details:\n" + removedFile.Dir());
    }
}

private void btnSearch_Click(object sender, EventArgs e)
{
    if (queueFiles.IsEmpty())
    {
        MessageBox.Show("There is no files to search.");
        HideAllGroupBoxes();
        groupBox1.Visible = true;
        return;
    }
    HideAllGroupBoxes();
    groupBoxSearch.Visible = true;
}

private void btnBiggest_Click(object sender, EventArgs e)
{
    if (queueFiles.IsEmpty())
    {
        MessageBox.Show("There are no files in the queue.");
    }
    else
    {
        DataFile biggestFile = queueFiles.BigFile();

        if (biggestFile != null)
        {
            HideAllGroupBoxes();
            groupBoxBiggest.Visible = true;

            textBoxBiggest.Text = biggestFile.Dir();
        }
    }
}

private void comboBoxTypeSearch_SelectedIndexChanged(object
sender, EventArgs e)
{
    if (comboBoxTypeSearch.SelectedItem != null)
    {
        selectedFileType =
(FileTypeExtension)Enum.Parse(typeof(FileTypeExtension),
comboBoxTypeSearch.SelectedItem.ToString());
    }
}

private void btnAddContent_Click(object sender, EventArgs e)
{
    try
    {
        if (string.IsNullOrEmpty(textBoxFileName.Text))
        {
            MessageBox.Show("Please fill the file name.");
            return;
        }
        string fileName = textBoxFileName.Text;
        string data = txtBoxContent.Text;
        if (comboBoxTypeAdd.SelectedItem != null)
        {

```

```

        FileTypeExtension fileType =
(FileTypeExtension)Enum.Parse(typeof(FileTypeExtension),
comboBoxTypeAdd.SelectedItem.ToString());
        DataFile dataFile = new DataFile(fileName, data,
fileType);
        if (dataFile.GetFileName() == fileName) // Check if
the file name is valid
        {
            queueFiles.Enqueue(dataFile);
            UpdateFileList();
            MessageBox.Show("File was added!");
            DialogResult result = MessageBox.Show("Do you want
to add another file?", "Add another file", MessageBoxButtons.YesNo);
            if (result == DialogResult.No)
            {
                groupBox1.Visible = true;
                groupBoxAddFile.Visible = false;
            }
        }
        else
        {
            MessageBox.Show("The file name you entered is not
valid. Please try again with a different file name.");
            return;
        }
    }
    else
    {
        MessageBox.Show("Please select a file type.");
    }
    textBoxFileName.Clear();
    txtBoxContent.Clear();
    comboBoxTypeAdd.SelectedIndex = -1;
}
catch (Exception ex)
{
    MessageBox.Show("An error occurred: " + ex.Message);
}
}

private void btnSearchByType_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    groupBoxDisplay.Visible = true;
    DataFile[] files =
queueFiles.SearchFileByType(selectedFileType);
    listBoxDisplayFiles.Items.Clear();

    if (files != null && files.Length > 0) // Add found files to
the list
    {
        foreach (DataFile file in files)
        {
            listBoxDisplayFiles.Items.Add(file.Dir());
        }
    }
    else
    {
        HideAllGroupBoxes();
        groupBoxSearch.Visible = true;
    }
}

```

```

        MessageBox.Show("There is no files that matching to the
type you choose.");
    }
}

private void UpdateFileList()
{
    listBoxDisplayFiles.Items.Clear();

    foreach (DataFile file in queueFiles.GetNonEmptyFiles()) //
Add files from the queue to the list
    {
        listBoxDisplayFiles.Items.Add(file.Dir());
    }
}

private void groupBoxSearch_Enter(object sender, EventArgs e)
{
    comboBoxTypeSearch.DataSource =
Enum.GetValues(typeof(FileTypeExtension));
}

private void btnDisplay_Click(object sender, EventArgs e)
{
    if (queueFiles.IsEmpty())
    {
        MessageBox.Show("There is no files to display!");
    }
    else
    {
        HideAllGroupBoxes();
        groupBoxDisplay.Visible = true;
    }
}

private void Form1_Load_1(object sender, EventArgs e)
{
    if (files == null)
    {
        files = new QueueFiles();
    }
    comboBoxTypeAdd.DataSource =
Enum.GetValues(typeof(FileTypeExtension));
    HideAllGroupBoxes();
    groupBox1.Visible = true;
}

private void btnBackBiggest_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    groupBox1.Visible=true;
}

private void button1_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    groupBox1.Visible = true;
}

private void btnBack_Click(object sender, EventArgs e)
{

```



```

        HideAllGroupBoxes();
        groupBox1.Visible = true;
    }

    private void btnSearchTypeBack_Click(object sender, EventArgs e)
    {
        HideAllGroupBoxes();
        groupBox1.Visible = true;
    }

    private void HideAllGroupBoxes()
    {
        groupBox1.Visible = false;
        groupBoxAddFile.Visible = false;
        groupBoxSearch.Visible = false;
        groupBoxDisplay.Visible = false;
        groupBoxBiggest.Visible = false;
    }
}

```

## DataFiles.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUIfinalQ1
{
    public enum FileTypeExtension // Q1
    {
        txt = 1,
        doc = 2,
        docx = 3,
        pdf = 4,
        pptx = 5
    }

    class DataFile
    {
        private static int counter = 1; // Q3 B
        private string fileName;
        private DateTime lastUpdateTime;
        private string data;
        private FileTypeExtension type; // Q3 A

        public DataFile(string fileName, string data, FileTypeExtension
type) // Q3 C i
        {
            SetFileName(fileName);
            SetData(data);
            SetTime();
            this.type = type;
        }

        public DataFile() : this("sameFile" + counter++, "",
FileTypeExtension.txt) // Q3 C ii
        {

```

```

    }

    public FileTypeExtension GetFileType()
    {
        return this.type;
    }
    public DataFile(string name, DataFile other) : this()
    {
        fileName = "Copy of " + other.fileName;
        data = other.data;
        SetTime();
    }

    public void SetFileName(string fileName)
    {
        if (IsValidFileName(fileName))
        {
            this.fileName = fileName;
            counter++;
        }
    }

    public string GetFileName()
    {
        return fileName;
    }

    public void SetData(string data)
    {
        this.data = data;
    }

    public string GetData()
    {
        return data;
    }

    public void SetTime()
    {
        lastUpdateTime = DateTime.Now;
    }

    public DateTime GetTime()
    {
        return lastUpdateTime;
    }

    private bool IsValidFileName(string fileName)
    {
        foreach (char invalidChar in new char[] { '}', '>', '?', '*',
':', '/', '\\', '|', '<' })
        {
            if (fileName.Contains(invalidChar))
            {
                return false;
            }
        }
        return true;
    }

```

```

        public int GetSize()
        {
            return data.Length;
        }

        public string Dir()
        {
            return fileName + " " + lastUpdateTime.ToString() + " " +
((float)this.GetSize() / 1024) + " KB ";
        }

    }
}

```

## QueueFiles.cs:

```

using GUIPT2;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUIfinalQ1
{
    class QueueFiles
    {
        private DataFile[] files;
        private int freeIndex;

        public QueueFiles()
        {
            files = new DataFile[0];
            freeIndex = -1;
        }

        public bool IsEmpty() // check if the queue is empty
        {
            if (freeIndex == -1)
            {
                return true;
            }
            return false;
        }

        public void Enqueue(DataFile file) // add file to the end of the
queue
        {
            try
            {
                bool fileExists = false;
                foreach (var f in files)
                {
                    if (CompareFiles.EqualFiles(f, file))
                    {

```

```

        fileExists = false;
        break;
    }
}
if (fileExists)
{
    return;
}

freeIndex++;
if (freeIndex == files.Length)
{
    Array.Resize(ref files, files.Length + 1);
}
files[freeIndex] = file;
}
catch (Exception ex)
{
    MessageBox.Show("An error occurred while enqueueing the
file: " + ex.Message);
}
}

```

```

public DataFile Dequeue() // take out file from the queue
{
    if (IsEmpty())
    {
        return null;
    }

    DataFile removedFile = files[0];
    for (int i = 1; i <= freeIndex; i++)
    {
        files[i - 1] = files[i];
    }
    freeIndex--;
    return removedFile;
}

```

```

public DataFile BigFile() // gives ref to the biggest size file
{
    if (IsEmpty())
    {
        return null;
    }

    if (freeIndex == 0)
    {
        return files[0];
    }

    QueueFiles tempQueue = new QueueFiles();
    DataFile maxSizeFile = files[0];

    foreach (DataFile file in files)
    {
        if (CompareFiles.CompareSizeFiles(file, maxSizeFile) > 0)
        {
            maxSizeFile = file;
        }
    }
}

```

```

        tempQueue.Enqueue(file);
    }
    if (!tempQueue.IsEmpty())
    {
        Enqueue(tempQueue.Dequeue());
    }
    return maxSizeFile;
}

public void PrintQueue() // print the whole details of files in
the queue
{
    if (IsEmpty())
    {
        return;
    }

    for (int i = 0; i <= freeIndex; i++)
    {
        files[i].Dir();
    }
}

public DataFile[] GetNonEmptyFiles()
{
    DataFile[] nonEmptyFiles = new DataFile[freeIndex + 1];
    Array.Copy(files, nonEmptyFiles, freeIndex + 1);
    return nonEmptyFiles;
}

public DataFile[] SearchFileByType(FileTypeExtension type) // gets
file type and gives the file in the queue with the same extantion
{
    if (IsEmpty())
    {
        return null;
    }
    List<DataFile> fileList = new List<DataFile>();
    foreach (DataFile file in files)
    {
        if (file.GetFileType() == type)
        {
            fileList.Add(file);
        }
    }
    return fileList.ToArray();
}
}
}

```

## CompareFiles.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace GUIfinalQ1
{
    static class CompareFiles
    {
        public static bool EqualFiles(DataFile file1, DataFile file2) //
        Q2 check if the files equals
        {
            if (file1.GetFileName() == file2.GetFileName() &&
            file1.GetData() == file2.GetData())
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public static int CompareSizeFiles(DataFile file1, DataFile file2)
        // Q2 B compare between the files
        {
            int fileSize1 = file1.GetSize();
            int fileSize2 = file2.GetSize();

            if (fileSize1 > fileSize2)
            {
                return 1;
            }
            else if (fileSize2 > fileSize1)
            {
                return -1;
            }
            else
            {
                return 0;
            }
        }
    }
}

```

## שאלה 2:

בשאלה זו החלטנו ליצור חנות "אינטרנטית" של ספרי קריאה. החנות מכילה מגוון אפשרויות למשתמש וניסינו לתכנן אותה כך שתהיה נוחה לשימוש והבנה.

הקוד עצמו מכיל 4 מחלקות:

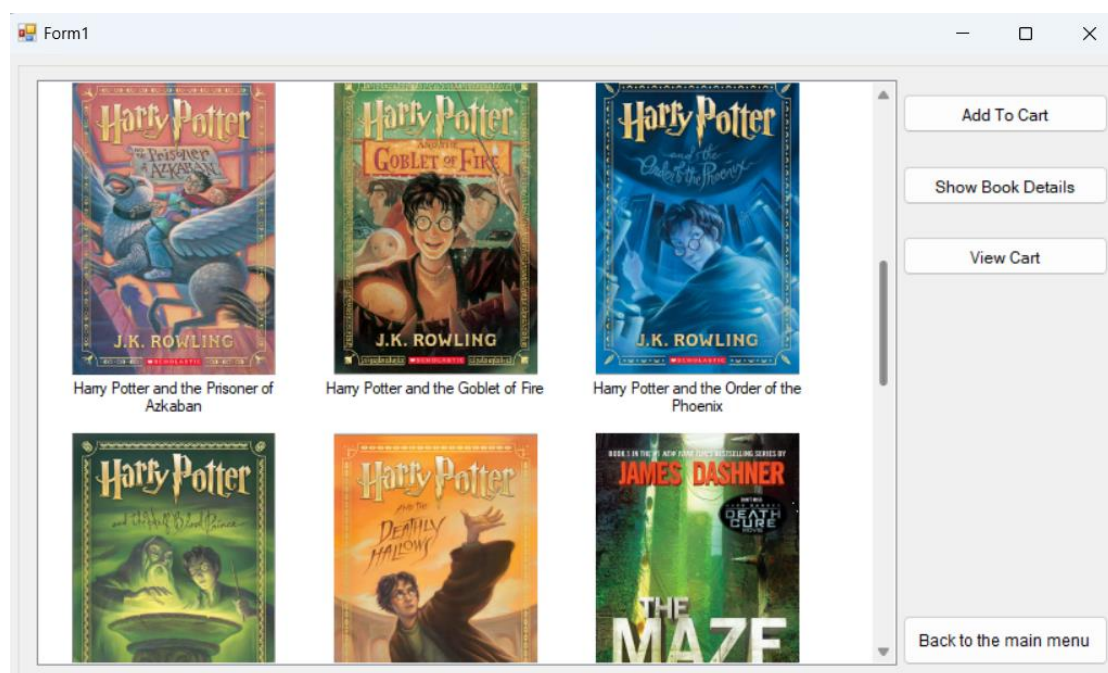
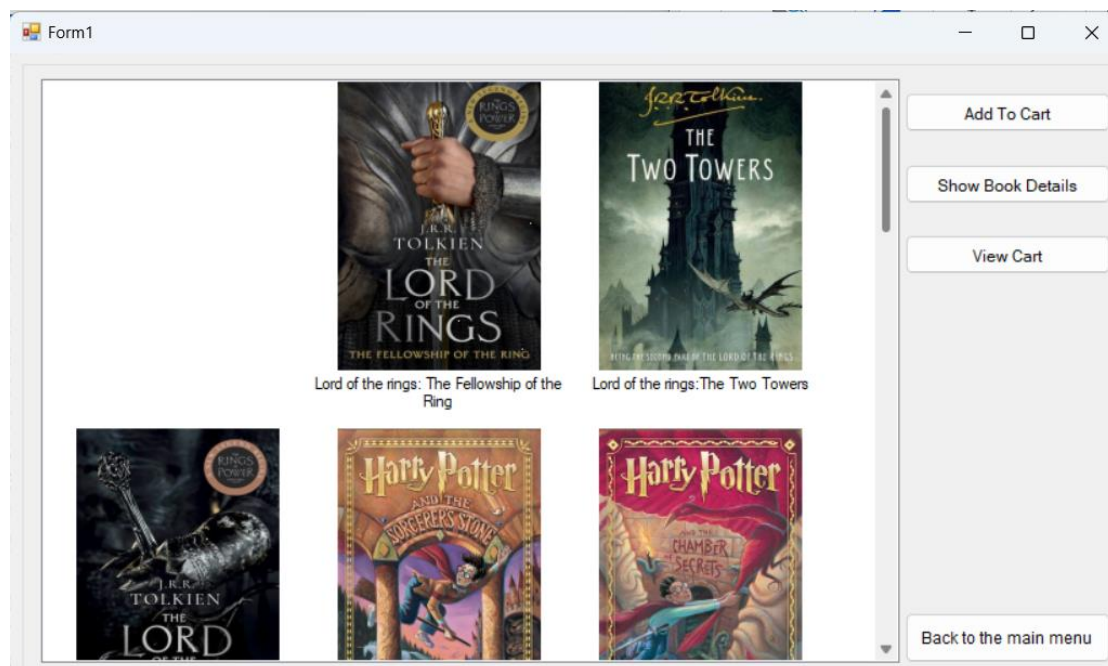
1. מחלקה מרכזית "Form1" המנהלת את הממשק החזותי למשתמש
2. מחלקה בשם "BookLibrary" המחזיקה בתוכה את שמות הספרים והתוכן שלהם ופעולות נוספות הקשורות
3. מחלקה בשם "Payment" המחזיקה ומנהלת את התשלום, התנאים על פרטי התשלום וכדומה
4. מחלקה בשם "Cart" המנהלת את סל הרכישה ובה ניתן למצוא את ניהול סל הקניות.

לצד הקוד נמצא פירוט המביע את פעולות הקוד!

תחילה בפני המשתמש מוצג מסך הפתיחה ומשם יש לו את האפשרויות לבחירה:



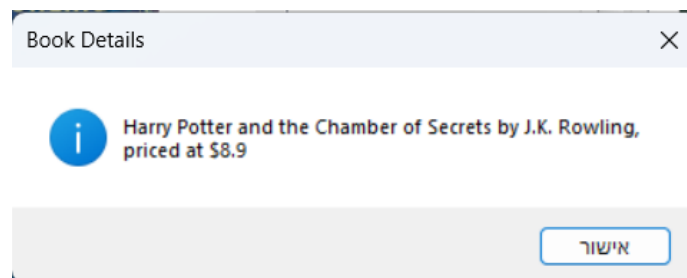
כאשר המשתמש לוחץ על כפתור **"View all the available books online"** מוצג בפני המשתמש כל הספרים האפשריים לרכישה אינטרנטית (ישנם ספרים נוספים בקטלוג מעבר למוצגים):



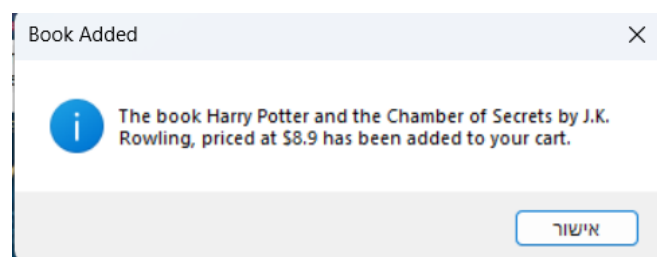


במסך זה יש למשתמש את האפשרות לצפות בפרטי הספר ולהוסיפו לסל. בכל פעולה יקבל המשתמש הודעה מתאימה.

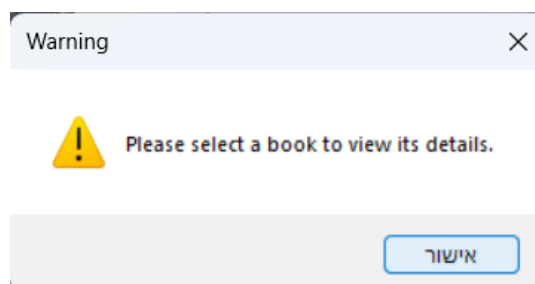
במידה ולחץ על **"Show book details"**:



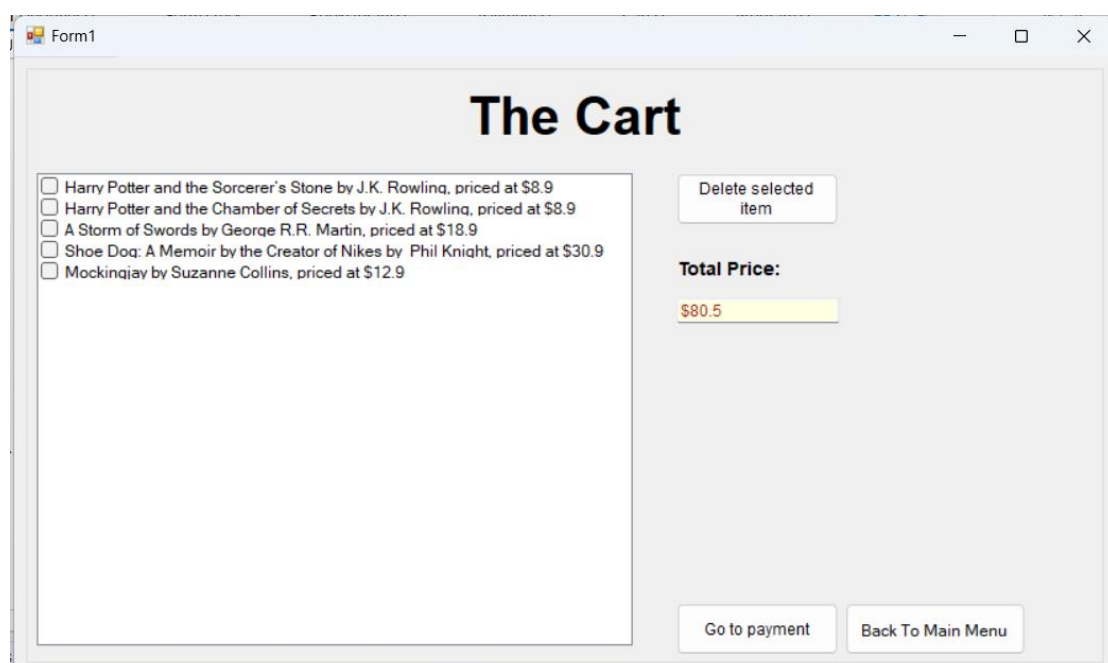
במידה ולחץ על **"Add to cart"**:



במידה ולא סימן ספר וביצע פעולה יקבל הודעה מתאימה:



כאשר המשתמש לוחץ על כפתור **"Go to cart"** במסך הראשי או **"View cart"** במסך הספרים מוצג בפני המשתמש סל הרכישה:



בפני המשתמש ישנה האפשרות למחוק מוצר מהרשימה ולהתקדם לתשלום. בעת מחיקה יתעדכן מיידית המחיר וימחק המוצר.

במידה ולחץ על מחיקה ולא סימן מוצר יקבל הודעה מתאימה.

בעת לחיצה על "Go To Payment" יועבר המשתמש למסך התשלום, במידה והסל ריק יקבל המשתמש הודעה מתאימה:

The screenshot shows a web form titled "Final Step!" with a total price of \$123.4. It contains two main sections: "Shipping Details" and "Payment Details". The "Shipping Details" section includes fields for Full Name, Tel Number, Email Address, and Shipping Address (Street Address, city, Zip Code). The "Payment Details" section includes fields for Name On Card, Card Number, Expiring MM/YY (with dropdown menus), and CVV code. A green checkmark is visible next to the "Confirm Payment" button. A "Back To Main Menu" button is also present.

Form1

Total Price: \$123.4  
\*All fields are required

### Final Step!

**Shipping Details**

Full Name:

Tel Number:

Email Address:

**Shipping Address**

Street Address:

city:

Zip Code:

**Payment Details**

Name On Card:

Card Number:

Expiring MM/YY:

CVV code:

**Confirm Payment**

Back To Main Menu

The screenshot shows a web page titled "The Cart". It features a large empty box for the cart items, a "Delete selected item" button, and a "Total Price:" field showing "\$0". A warning dialog box is displayed in the center, stating "The cart is empty. Please add items to the cart before proceeding." with a yellow warning icon and a button labeled "אישור" (Confirm). At the bottom, there are buttons for "Go to payment" and "Back To Main Menu".

Form1

## The Cart

Delete selected item

Total Price: \$0

**Warning**

The cart is empty. Please add items to the cart before proceeding.

אישור

Go to payment

Back To Main Menu

במסך התשלום על המשתמש למלא את פרטיו. לחלק משדות הפרטים יש מגבלה, כמו CVV אשר בתיבה זו על המשתמש למלא 3 ספרות בלבד, תיבת האימייל למלא מבנה של כתובת אימייל בלבד ועוד (ההגבלות רשומות). על המשתמש למלא את כל השדות כדי להשלים את הרכישה.

במידה ולא ימלא את אחד הפרטים נכונה או לא ימלא באופן כללי את אחד מהם יקבל הודעה מתאימה. במידה ומילא את פרטיו בהצלחה יקבל המשתמש הודעה עם קבלה לרכישה:

The screenshot shows a web form titled "Form1" with a "Total Price: \$99.4" and a note "\*All fields are required". The form is divided into two main sections: "Shipping Details" and "Payment Details".

**Shipping Details:**

- Full Name: omer cohen
- Tel Number: 0509403660
- Email Address: omer430@gmail.com (Example@examp.example)

**Payment Details:**

- Name On Card: omer cohen
- Card Number: 4580978654982579 (digit only)
- Expiring MM/YY: May Y2028
- CVV code: 598 (3 digit)

**Shipping Address:**

- Street Address: hahagana 93
- city: herzeliya
- Zip Code: 454059 (digit only)

A "Receipt" modal is displayed in the center, showing a summary of the order:

- Full Name: omer cohen
- Telephone Number: 0509403660
- Email Address: omer430@gmail.com
- Street Address: hahagana 93
- City: herzeliya
- Zip Code: 454059
- Name on Card: omer cohen
- Card Number: 4580978654982579
- Expiry Date: May/Y2028
- CVV: 598
- Total Price: 99.4

At the bottom right of the form, there is a green checkmark icon, a "Confirm Payment" button, and a "Back To Main Menu" button. At the bottom of the receipt modal, there is an "אישור" (Confirm) button.

A "Warning" dialog box with a yellow warning icon and the text "Please fill all required fields correctly." At the bottom, there is an "אישור" (Confirm) button.

בנוסף, כאשר המשתמש מנסה להזין תאריך תפוגה שלפני התאריך הנוכחי של היום יקבל הודעת שגיאה:

A "Warning" dialog box with a yellow warning icon and the text "Please select a valid expiry date." At the bottom, there is an "אישור" (Confirm) button.

כאשר המשתמש לוחץ על כפתור "View The Store Details" במסך הראשי יופיע למשתמש החנויות הפיזיות של החברה:

The screenshot shows a Windows application window titled "Form1". Inside the window, the title "Our Locations" is centered at the top. Below the title, there are three columns of information, each representing a store location:

- Hadera:** Jabotinsky 56 Hadera, Tel no. +972-509-865-8923, Sunday - Thursday 9:30-19:30, Friday 8:30-14:30, Saturday Closed.
- Herzeliya:** Hhagana 56 Herzeliya, Tel no. +972-509-766-8223, Sunday - Thursday 9:30-19:30, Friday 8:30-14:30, Saturday Closed.
- Tel Aviv:** Hasalom 21 Tel Aviv, Tel no. +972-509-865-8946, Sunday - Thursday 9:30-19:30, Friday 8:30-14:30, Saturday Closed.

At the bottom right of the form, there is a button labeled "Back To Main Menu".

## קוד התכנית:

### Form1.cs:

```
using GUIPT2Q3;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUIfinalQ2
{
    public partial class Form1 : Form
    {
        private Cart cart;

        public Form1() // handling the user interface
        {
            InitializeComponent();
            BookLibrary.InitializeAvailableBooks();
            timer1.Interval = 1000;
            timer1.Enabled = true;
            cart = new Cart();

            foreach (Payment.Month month in
Enum.GetValues(typeof(Payment.Month)))
            {
                cmbBoxMM.Items.Add(month);
            }
            foreach (Payment.Year year in
Enum.GetValues(typeof(Payment.Year)))
            {
```

```

        cmbBoxYY.Items.Add(year);
    }

    foreach (BookLibrary book in BookLibrary.AvailableBooks)
    {
        ListViewItem item = new ListViewItem();
        item.Text = book.Title;
        item.ImageIndex =
BookLibrary.AvailableBooks.IndexOf(book); // Assuming that images are
added to ImageList in same order as books
        listView1.Items.Add(item);
    }
    listView1.LargeImageList = imageList1;
}

private void timer1_Tick(object sender, EventArgs e) // Keeps the
total price in the payment group box updated
{
    grpBoxPayment.Text = "Total Price: $" + cart.TotalPrice;
}

private void btnShowDetails_Click(object sender, EventArgs e)
//Shows the details of a selected book.
{
    if (listView1.SelectedItems.Count > 0)
    {
        var selectedBook =
BookLibrary.AvailableBooks[listView1.SelectedItems[0].ImageIndex];
        MessageBox.Show(selectedBook.GetDetails(), "Book Details",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show("Please select a book to view its
details.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

private void btnAddToCart_Click(object sender, EventArgs e) //Adds
the selected book to the cart.
{
    if (BookLibrary.IsBookSelected(listView1.SelectedItems.Count))
    {
        var selectedBook =
BookLibrary.GetSelectedBook(listView1.SelectedItems[0].ImageIndex);
        cart.AddBookToCart(selectedBook, chklstCart, textBox1);
        MessageBox.Show("The book " + selectedBook.GetDetails() +
" has been added to your cart.", "Book Added", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
    else
    {
        ShowWarning("Please select a book to add to your cart.");
    }
}

private void btnDeleteItem_Click(object sender, EventArgs e)
//Deletes the selected book from the cart.
{
    if (cart.IsBookSelected(chklstCart.SelectedIndex))
    {
        cart.RemoveBookFromCart(chklstCart.SelectedIndex,
chklstCart, textBox1);
    }
}

```

```

    }
    else
    {
        ShowWarning("Please select a book to remove from your
cart.");
    }
}

private void ShowWarning(string message) // Shows a warning
message
{
    MessageBox.Show(message, "Warning", MessageBoxButtons.OK,
    MessageBoxIcon.Warning);
}

private void button1_Click_1(object sender, EventArgs e) // checks
if the feils the user insert are correct
{
    // Check if all required fields are filled and the expiry date
is valid
    if (!Payment.IsFormValid(txtBoxFullName.Text,
txtBoxTelNo.Text, lblEmailAddress.Text, txtBoxStreetAddress.Text,
txtBoxCity.Text, txtBoxZipCode.Text, txtBoxNameOnCard.Text,
txtBoxCardNo.Text, cmbBoxMM.Text, cmbBoxYY.Text, txtBoxCVV.Text))
    {
        MessageBox.Show("Please fill all required fields
correctly.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else if (!Payment.IsExpiryDateValid(cmbBoxMM.Text,
cmbBoxYY.Text))
    {
        MessageBox.Show("Please select a valid expiry date.",
"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        string receipt = "Receipt:\n\nFull Name: " +
txtBoxFullName.Text + "\nTelephone Number: " + txtBoxTelNo.Text + "\nEmail
Address: " + lblEmailAddress.Text + "\nStreet Address: " +
txtBoxStreetAddress.Text + "\nCity: " + txtBoxCity.Text + "\nZip Code: " +
txtBoxZipCode.Text + "\nName on Card: " + txtBoxNameOnCard.Text + "\nCard
Number: " + txtBoxCardNo.Text + "\nExpiry Date: " + cmbBoxMM.Text + "/" +
cmbBoxYY.Text + "\nCVV: " + txtBoxCVV.Text + "\n\nTotal Price: " +
cart.TotalPrice;
        MessageBox.Show(receipt, "Receipt", MessageBoxButtons.OK,
        MessageBoxIcon.Information); //recipt
    }
}

private void btnGoToPayment_Click(object sender, EventArgs e)
//Checks if the cart is empty, if not sending the user to the payment
{
    if (cart.Books.Count == 0)
    {
        MessageBox.Show("The cart is empty. Please add items to
the cart before proceeding.", "Warning", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
    }
    else
    {
        HideAllGroupBoxes();
    }
}

```

```

        grpBoxPayment.Visible = true;
    }
}

private void btnLoactionBack_Click_1(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void btnLoactionBack_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void btnBackToMenu_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void btnCartBack_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void Form1_Load(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void btnViewAvailableBooks_Click_1(object sender,
EventArgs e)
{
    HideAllGroupBoxes();
    grpboxBooks.Visible = true;
}

private void btnMainMenuCart_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpBoxCart.Visible = true;
}

private void btnStoreDetails_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpBoxLocation.Visible = true;
}

private void btnCatalogBack_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void btnViewCart_Click(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpBoxCart.Visible = true;
}

```

```

private void btnCatalogBack_Click_1(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpboxMainMenu.Visible = true;
}

private void btnViewCart_Click_1(object sender, EventArgs e)
{
    HideAllGroupBoxes();
    grpBoxCart.Visible = true;
}

private void HideAllGroupBoxes() // set as default all the group
box visible to false for easier handle
{
    grpboxMainMenu.Visible = false;
    grpboxBooks.Visible = false;
    grpBoxCart.Visible = false;
    grpBoxLocation.Visible = false;
    grpBoxPayment.Visible = false;
}
}
}

```

## Cart.cs:

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GUIfinalQ2
{
    public class Cart
    {
        public List<BookLibrary>
            Books { get; set; }
        public double TotalPrice { get; set; }

        public Cart()
        {
            Books = new List<BookLibrary>();
            TotalPrice = 0;
        }

        //Add book from and to the cart and change the total price
        public void AddBook(BookLibrary book)
        {
            Books.Add(book);
            TotalPrice += book.Price;
        }
    }
}

```



```

    }

    //Remove book from and to the cart and change the total price
    public void RemoveBook(BookLibrary book)
    {
        Books.Remove(book);
        TotalPrice -= book.Price;
    }

    //Add book from and to the cart update the list box
    public void AddBookToCart(BookLibrary selectedBook, CheckedListBox
chklstCart, TextBox textBox1)
    {
        AddBook(selectedBook);
        chklstCart.Items.Add(selectedBook.GetDetails());
        textBox1.Text = "$" + TotalPrice;
    }

    //Remove book from and to the cart update the list box
    public void RemoveBookFromCart(int selectedIndex, CheckedListBox
chklstCart, TextBox textBox1)
    {
        var selectedBook = Books[selectedIndex];
        RemoveBook(selectedBook);
        chklstCart.Items.RemoveAt(selectedIndex);

        if (Books.Count == 0)
        {
            TotalPrice = 0;
        }

        textBox1.Text = "$" + TotalPrice;
    }

    //Checks if a book is selected in the cart
    public bool IsBookSelected(int selectedIndex)
    {
        return selectedIndex != -1;
    }
}

```

## BookLibrary.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Windows.Forms;

```

```

namespace GUIfinalQ2
{
    public class BookLibrary
    {
        public string Title { get; set; }
        public string Author { get; set; }
        public double Price { get; set; }
        public Image Cover { get; set; }

        public static List<BookLibrary> AvailableBooks { get; private set; }
    }

    public BookLibrary(string title, string author, double price)
    {
        Title = title;
        Author = author;
        Price = price;
    }

    public string GetDetails() // Return the book details
    {
        return Title + " by " + Author + ", priced at $" + Price;
    }

    //Checks if a book is selected in the available books
    public static bool IsBookSelected(int selectedItemsCount)
    {
        return selectedItemsCount > 0;
    }

    public static BookLibrary GetSelectedBook(int selectedIndex)
    //Returns the selected book
    {
        return AvailableBooks[selectedIndex];
    }

    // Initializes the list of available books, we prefer to use it
    for better dynamism
    public static void InitializeAvailableBooks()
    {
        AvailableBooks = new List<BookLibrary>()
        {
            new BookLibrary("Lord of the rings: The Fellowship of the
Ring", "J.R.R. Tolkien", 9.90),
            new BookLibrary("Lord of the rings:The Two Towers", "J.R.R.
Tolkien", 9.90),
            new BookLibrary("Lord of the rings:The Return of the King",
"J.R.R. Tolkien", 9.90),
            new BookLibrary("Harry Potter and the Sorcerer's Stone",
"J.K. Rowling", 8.90),
            new BookLibrary("Harry Potter and the Chamber of Secrets",
"J.K. Rowling", 8.90),
            new BookLibrary("Harry Potter and the Prisoner of Azkaban",
"J.K. Rowling", 9.00),
            new BookLibrary("Harry Potter and the Goblet of Fire", "J.K.
Rowling", 9.70),
            new BookLibrary("Harry Potter and the Order of the Phoenix",
"J.K. Rowling", 7.90),
            new BookLibrary("Harry Potter and the Half-Blood Prince",
"J.K. Rowling", 9.50),
            new BookLibrary("Harry Potter and the Deathly Hallows",
"J.K. Rowling", 9.30),
            new BookLibrary("The Maze Runner: book 1", "James Dashner",
11.90),

```

```

        new BookLibrary("The Book Thief", "Markus Zusak", 13.90),
        new BookLibrary("The Hunger Games", "Suzanne Collins",
14.50),
        new BookLibrary("Catching Fire", "Suzanne Collins", 15.90),
        new BookLibrary("Mockingjay", "Suzanne Collins", 12.90),
        new BookLibrary("Fire & Blood", "George R.R. Martin",
11.00),
        new BookLibrary("A Game of Thrones", "George R.R. Martin",
20.00),
        new BookLibrary("A Clash of Kings", "George R.R. Martin",
21.90),
        new BookLibrary("A Storm of Swords", "George R.R. Martin",
18.90),
        new BookLibrary("Shoe Dog: A Memoir by the Creator of
Nikes", " Phil Knight", 30.90),
    };
}
}
}

```

## Payment.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace GUIfinalQ2
{
    public class Payment
    {
        // the values for the expiring date
        public enum Month
        {
            January = 1,
            February,
            March,
            April,
            May,
            June,
            July,
            August,
            September,
            October,
            November,
            December
        }

        // the values for the expiring date
        public enum Year

```

```

    {
        Y2023 = 2023, Y2024, Y2025, Y2026, Y2027, Y2028, Y2029, Y2030
    }
    // I had to add the character Y

    //Checks if all form fields are correctly
    public static bool IsFormValid(string fullName, string telNo,
    string email, string streetAddress, string city, string zipCode, string
    nameOnCard, string cardNo, string expiryMonth, string expiryYear, string
    cvv)
    {
        if (string.IsNullOrEmpty(fullName) ||
            string.IsNullOrEmpty(telNo) ||
            string.IsNullOrEmpty(email) ||
            string.IsNullOrEmpty(streetAddress) ||
            string.IsNullOrEmpty(city) ||
            string.IsNullOrEmpty(zipCode) ||
            string.IsNullOrEmpty(nameOnCard) ||
            string.IsNullOrEmpty(cardNo) ||
            string.IsNullOrEmpty(expiryMonth) ||
            string.IsNullOrEmpty(expiryYear) ||
            string.IsNullOrEmpty(cvv))
        {
            return false;
        }
        else if (!Regex.IsMatch(email, @"^\w-+(\.\w-+)*@([\w-
        ]+\.)+[a-zA-Z]{2,7}$"))
        {
            return false;
        }

        else if (!Regex.IsMatch(telNo, @"^\d+$")) // Check if telNo
contains only numbers
        {
            return false;
        }

        else if (!Regex.IsMatch(cvv, @"^\d{3}$")) // Check if cvv
contains exactly 3 digits
        {
            return false;
        }

        else if (!Regex.IsMatch(zipCode, @"^\d+$")) // Check if
zipCode contains only numbers
        {
            return false;
        }

        else if (!Regex.IsMatch(cardNo, @"^\d+$")) // Check if cardNo
contains only numbers
        {
            return false;
        }
        return true;
    }

    public static bool IsExpiryDateValid(string expiryMonth, string
    expiryYear) // check if the the expiring date is not before the date today
    {
        int selectedMonth = (int)(Month)Enum.Parse(typeof(Month),
    expiryMonth);
    }

```

```
        int selectedYear = (int)(Year)Enum.Parse(typeof(Year),
expiryYear);

        DateTime currentDate = DateTime.Now;
        DateTime selectedDate = new DateTime(selectedYear,
selectedMonth, 1);

        if (selectedDate < currentDate)
        {
            return false;
        }
        return true;
    }
}
```