# **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА**

# **ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ**

**Катедра: Компютърни науки и технологии**

**Специалност: Компютърни науки и технологии**

### **КУРСОВ ПРОЕКТ**

**по дисциплина „Софтуерни технологии“**

|  |  |
| --- | --- |
| Студент: Калоян Маринов Енев |  |
| Курс: III Група:1б |  |
| Факултетен номер: 21621330  **Срок на предаване:** 31.05.2024 г. |  |

**Ръководител:** . . . . . . . . . . . . . . . . .

(доц. В. Божикова)

1. **Задание**
   1. Цел на курсовия проект:Изисква се да се разработи „Софтуер за магазин за настолни игри“.
   2. Функционални изисквания:  
       Системата трябва да позволява съхраняването на данни в база от данни за:  
      - идентификационен номер на продукт (Id),  
      - име на игра (Name),  
      **-** вид на играта (Type),  
      - средна продължителност (Average playtime),  
      - препоръчителна възраст на играчите(Recommended age),  
      - цена (Price) и  
      - налично количество (Quantity).  
       Софтуерът трябва да позволи на потребителя да достъпи до базата данни и да може да редактира данните за всеки продукт, да добавя нови продукти или да изтрива съществуващи. Също така има функциите да търси продукт по избрана категория и да добавя и изтрива продукти в „количка“, като пригледно показва избраните продукти, заедно с единичната и общата им цена, след което има възможност да редактира наличностите.
   3. Нефункционални изисквания:  
       Във всяко от полето за въвеждане се използват функции за предотвратяване на грешно въведени данни. При употреба на функционалностите се генерират необходимите съобщения за потвърждение или насока на потребителя.
2. **Реализация на проекта**

Програмната реализация на курсовия проект е разделена на четири форми. Първата форма е за навигация и играе роля на главно меню. Включва четири опции, измежду които потребителя може да избере. Достъпът до тях се реализира чрез бутони. При натискане на бутон се зарежда съответната форма :

private void button1\_Click(object sender, EventArgs e);

При избор на опция “Exit”, програмата изкарва съобщение за потвърждение от потребителя:

private void button3\_Click(object sender, EventArgs e);

След отваряне на форма Datasetview се използват функции за визуализиране на базата данни във елемент DataGridView и за опресняване на данните при нужда:

private void PopulateDataGridView();

private void RefreshDataGridView();

В тази форма има 4 бутона: за добавяне на продукт, за изтриване на продукт по въведено име, за редактиране на продукт по въведено име и за въвеждане в текстовите полета на данните на определен продукт по въведен идентификатор:

private void button1\_Click(object sender, EventArgs e) - Insert

private void button2\_Click(object sender, EventArgs e) - Edit

private void button3\_Click(object sender, EventArgs e) - Delete

private void button4\_Click(object sender, EventArgs e) – Search by Id.

За всяко от текстовите полета има функция, която забранява на потребителя да въвежда грешни данни, според това какви данни се изискват (ако се изисква Id се забранява да се въвеждат всякакви символи с изключение на цифри):

private void textBox7\_KeyPress(object sender, KeyPressEventArgs e);.

След отваряне на форма Searching се дава възможност на потребителя да се използват четири различни критерии за търсене на продукти в базата данни, всеки от които е обслужван от бутон и текстово поле. Техните функции са:

private void button1\_Click(object sender, EventArgs e) – за търсене по Id

private void button2\_Click(object sender, EventArgs e) – за търсене по име

private void button3\_Click(object sender, EventArgs e) – за търсене по тип

private void button4\_Click(object sender, EventArgs e) – за търсене по възраст

Намереното се визуализира в елемент ListViewResults, който се опреснява след всяко ново търсене. Отново са направени функции за предотвратяване на нежелани символи в текстовите полета за данни:

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

В последната форма се дава възможност за попълване на елемент ListViewCart с продукти, взети от базата данни. ListView играе ролята на „количка“. Опциите са за добавяне на продукт по въведен идентификатор, премахване на продукт по въведен идентификатор и изчистване на количката:

private void button1\_Click(object sender, EventArgs e) – Add item

private void button2\_Click(object sender, EventArgs e) – Delete item

private void button3\_Click(object sender, EventArgs e) – Clear cart

При попълване на кошницата се пресмята общата стойност на всички добавени в „кошницата“ продукти. При попълнена кошница има възможността за „Checkout“, което изчиства всички въведени в „количката“ продукти:

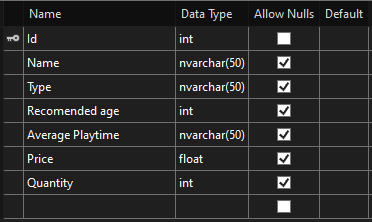
private void button4\_Click(object sender, EventArgs e);

При натискане на този бутон се отчитат промените в наличностите на всеки избран продукт. Това се извършва чрез функция:

private void UpdateQuantityInDatabase(int itemId, int quantityToSubtract)

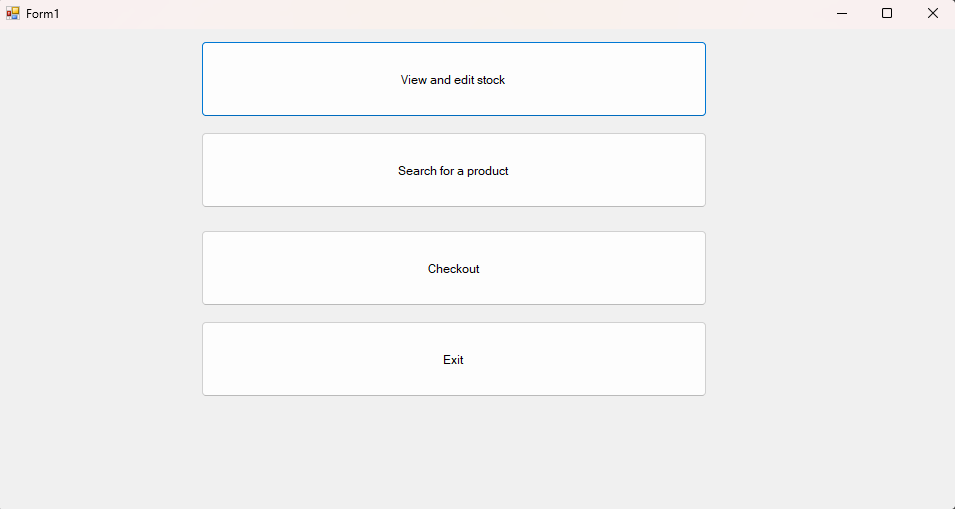
Отново се проверява дали се въвеждат правилни символи в текстовото поле за идентификатор:

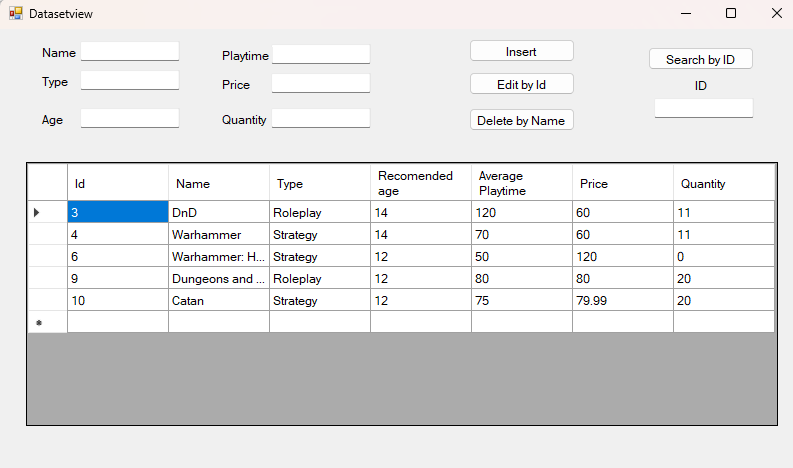
private void textBox2\_KeyPress(object sender, KeyPressEventArgs e).



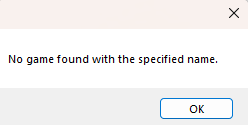
Елементите на базата данни Board\_games

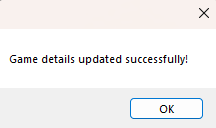
1. **Ръководство за потребителя**

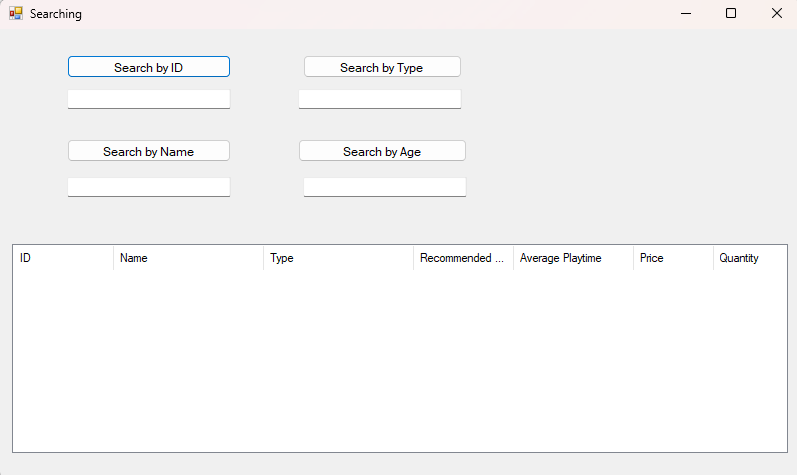
 След стартиране на приложението се визуализира главното меню. То има четири опции:  
1. Вход към формата за визуализация на базата данни и достъп до функциите за редактиране.  
2. Вход към формата за търсене на продукти по избран критерии.  
3. Вход към формата за управление на „количка“.  
4. Изход от програмата.

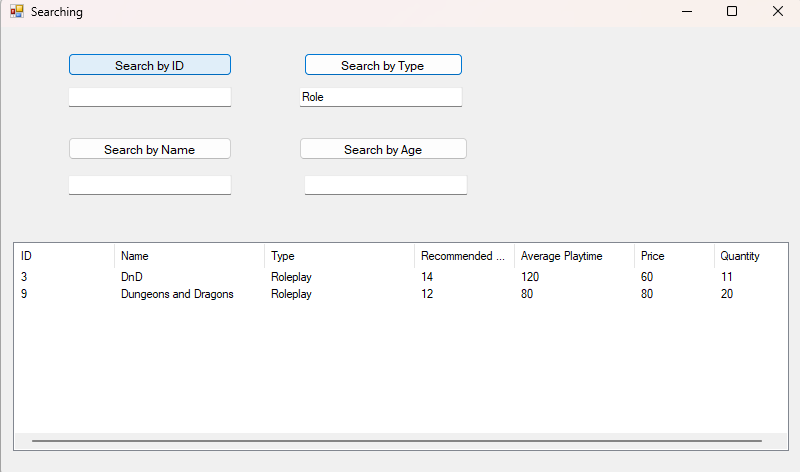
 Първата опция визуализира базата данни и дава възможност тя да бъде редактирана чрез бутони. Примерен изглед на този прозорец:

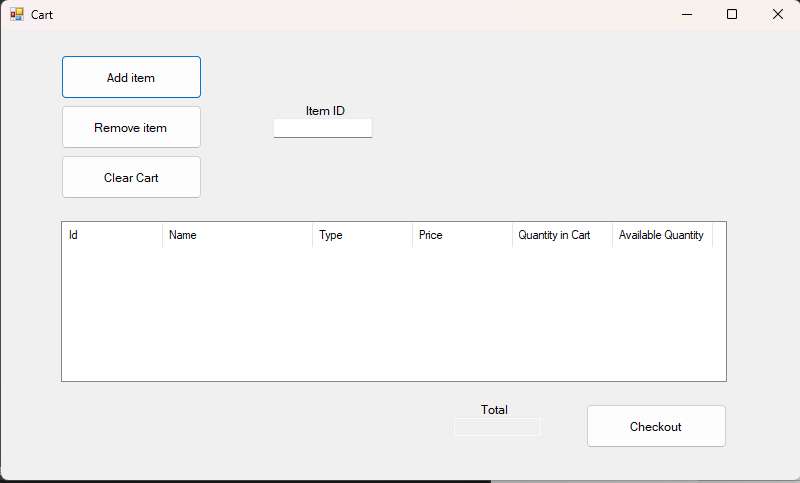
Бутон Insert създава нов продукт в базата данни и за целта използва информацията попълнена в шестте текстови полета от лявата му страна.  
 Бутон Edit by Id използва текстовото поле отдясно за идентификатор и променя данните на съответния елемент от базата данни според информацията попълнена отляво в шестте текстови полета. За улеснение на потребителя за попълването на шестте текстови полета се използва бутонът Search by ID, който според въведения идентификатор, попълва данните на съответния елемент в шестте текстови полета отляво.  
 Бутон Delete by Name изтрива елемент от базата данни според въведеното в съответното текстово поле име на продукт.

При несъществуващ търсен елемент излиза подходящо съобщение:

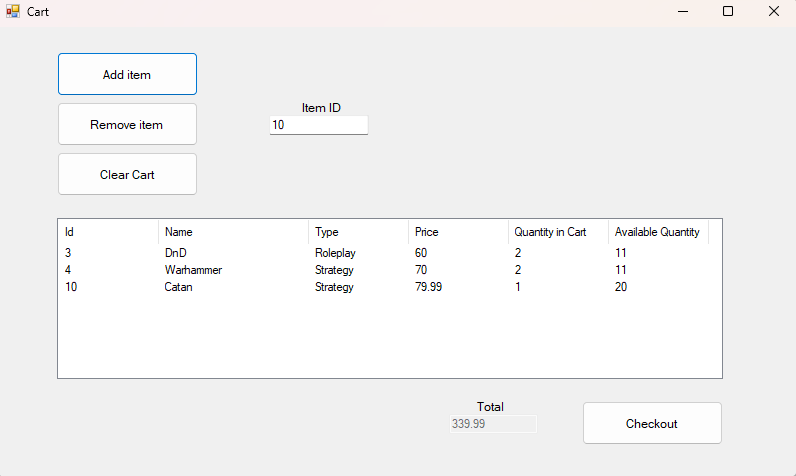
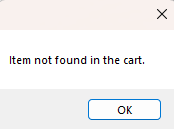
 При успешно въвеждане, изтриване или редактиране на продукт излиза съобщение:

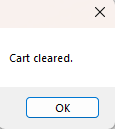
Втората опция от главното меню отваря прозорец за търсене на елементи в базата данни. Прозорецът изглежда така:

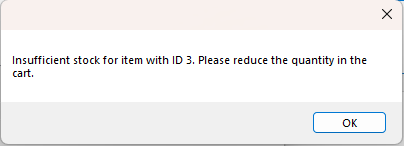
Този прозорец има четири бутона, всеки от които има различен критерии за извеждане на търсени продукти в списък:  
 1. Search by ID извежда продуктът с попълнен съответния индикатор.  
 2. Search by Name извежда продуктите, чиито имена имат общо с попълненото в съответното текстовото поле.  
 3. Search by Type извежда продуктите, чиито типове имат общо с попълненото в съответното текстово поле.  
 4. Search by Age извежда продуктите, чиито препоръчителни възрасти са по-големи от попълнената в съответното текстово поле.  
 Примерно търсене чрез тип:  


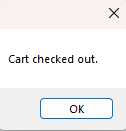
При избор на опция Checkout от главното меню се отваря прозорец, който изглежда така:

В този прозорец има четири опции, обслужвани от четири бутона:  
 1. Бутон Add item добавя продуктът съответстващ на идентификатора, въведен в текстовото поле Item ID. При повторно въвеждане на същия продукт се увеличава количеството му в количката с 1. Цената на всеки въведен продукт се пресмята и се добавя към общата цена, визуализирана под списъка на кошницата.

  
 2. Бутон Remove item премахва една бройка от вече въведен в кошницата продукт. Извежда подходящ съобщение ако продукт с такъв идентификатор не е в кошницата.

 3. Бутон Clear cart изчиства всички въведени продукти от списъка и се извежда съобщение при успех.

4. Бутон Checkout изчиства всички въведени продукти от списъка и отчита променените наличности на изчистените продукти. Ако се продаде последнат бройка от някой продукт се извежда подходящо съобщение. Ако добавеното в кошницата количество е повече от наличното се прекратява изчистването и се извежда съобщение за потребителя да коригира количествата. 

При успешно завършване извежда съобщението:  


1. **Изводи и възможности за развитие**

Курсовия проект за софтуер за магазин за настолни игри беше реализиран успешно. Включва функции за обслужване на база от данни, редактиране на продуктите в нея, добавяне, изтриване, търсене по критерии и „количка“, които да са лесно разбираеми и достъпни за потребителя. Функционалността му ще позволи и улесни отчитането на един такъв магазин на наличностите и други данни за продуктите.

За бъдещи подобрения на приложението може да се добавят повече опции за редакция на базата данни и повече начини за търсене на продукти и филтриране.

1. **Използвана литература**

**1. Технология на софтуерното производство – Ръководство за лабораторни упражнения**

**2.** <https://stackoverflow.com/> **3.** https://github.com/

1. Приложение
   1. Form1.cs

using System;

using System.ComponentModel.Design;

using System.Windows.Forms;

namespace WindowsFormsApp3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

this.tableTableAdapter.Fill(this.database1DataSet.Board\_games);

}

private void button1\_Click(object sender, EventArgs e)

{

Datasetview form2 = new Datasetview();

form2.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Searching form3 = new Searching();

form3.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Cart form4 = new Cart();

form4.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Are you sure you want to exit?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

Application.Exit();

}

}

}

}

* 1. Datasetview.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp3

{

public partial class Datasetview : Form

{

string connectionString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Kokito\\source\\repos\\WindowsFormsApp3\\WindowsFormsApp3\\Database1.mdf;Integrated Security=True;Connect Timeout=30";

public Datasetview()

{

InitializeComponent();

PopulateDataGridView();

}

private void RefreshDataGridView()

{

DataTable dt = new DataTable();

string query = "SELECT \* FROM Board\_games";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))

{

try

{

connection.Open();

adapter.Fill(dt);

tableDataGridView.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show("Error refreshing DataGridView: " + ex.Message);

}

}

}

private void PopulateDataGridView()

{

DataTable dataTable = new DataTable();

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT \* FROM Board\_games";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

adapter.Fill(dataTable);

}

tableDataGridView.DataSource = dataTable;

}

private void button1\_Click(object sender, EventArgs e)

{

string name = textBox1.Text;

string type = textBox2.Text;

int age = Convert.ToInt32(textBox3.Text);

string playtime = textBox4.Text;

double price = Convert.ToDouble(textBox5.Text);

int quantity = Convert.ToInt32(textBox6.Text);

string query = "INSERT INTO Board\_games (Name, Type, [Recomended age], [Average playtime], Price, Quantity) VALUES (@Name, @Type, @Age, @Playtime, @Price, @Quantity)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@Type", type);

command.Parameters.AddWithValue("@Age", age);

command.Parameters.AddWithValue("@Playtime", playtime);

command.Parameters.AddWithValue("@Price", price);

command.Parameters.AddWithValue("@Quantity", quantity);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

MessageBox.Show(rowsAffected + " row(s) inserted successfully!");

}

catch (Exception ex)

{

MessageBox.Show("Error inserting data: " + ex.Message);

}

}

PopulateDataGridView();

}

private void button2\_Click(object sender, EventArgs e)

{

string nameToDelete = textBox1.Text;

string query = "DELETE FROM Board\_games WHERE Name = @Name";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", nameToDelete);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

MessageBox.Show("Game deleted successfully!");

else

MessageBox.Show("No game found with the specified name.");

}

catch (Exception ex)

{

MessageBox.Show("Error deleting game: " + ex.Message);

}

}

RefreshDataGridView();

}

private void button3\_Click(object sender, EventArgs e)

{

string IDToEdit = textBox7.Text;

string query = "SELECT \* FROM Board\_games WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", IDToEdit);

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

textBox1.Text = reader["Name"].ToString();

textBox2.Text = reader["Type"].ToString();

textBox3.Text = reader["Recomended age"].ToString();

textBox4.Text = reader["Average playtime"].ToString();

textBox5.Text = reader["Price"].ToString();

textBox6.Text = reader["Quantity"].ToString();

}

else

{

MessageBox.Show("No game found with the specified name.");

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error retrieving game details: " + ex.Message);

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

int itemId = Convert.ToInt32(textBox7.Text);

string nameToUpdate = textBox1.Text;

string type = textBox2.Text;

int recommendedAge = Convert.ToInt32(textBox3.Text);

int averagePlaytime = Convert.ToInt32(textBox4.Text);

double price = Convert.ToDouble(textBox5.Text);

int quantity = Convert.ToInt32(textBox6.Text);

string query = "UPDATE Board\_games SET Name = @Name, Type = @Type, [Recomended age] = @RecommendedAge, [Average playtime] = @AveragePlaytime, Price = @Price, Quantity = @Quantity WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", itemId);

command.Parameters.AddWithValue("@Name", nameToUpdate);

command.Parameters.AddWithValue("@Type", type);

command.Parameters.AddWithValue("@RecommendedAge", recommendedAge);

command.Parameters.AddWithValue("@AveragePlaytime", averagePlaytime);

command.Parameters.AddWithValue("@Price", price);

command.Parameters.AddWithValue("@Quantity", quantity);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Game details updated successfully!");

RefreshDataGridView();

}

else

MessageBox.Show("No game found with the specified ID.");

}

catch (Exception ex)

{

MessageBox.Show("Error updating game details: " + ex.Message);

}

}

}

private void textBox7\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox7.Focus();

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox1.Focus();

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox2.Focus();

}

}

private void textBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar) && e.KeyChar != '.' && e.KeyChar != '-')

{

e.Handled = true;

textBox5.Focus();

}

if (e.KeyChar == '.' && (sender as TextBox).Text.IndexOf('.') > 0)

{

e.Handled = true;

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox3.Focus();

}

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox4.Focus();

}

}

private void textBox6\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox6.Focus();

}

}

}

}

* 1. Searching.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp3

{

public partial class Searching : Form

{

private string connectionString= "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Kokito\\source\\repos\\WindowsFormsApp3\\WindowsFormsApp3\\Database1.mdf;Integrated Security=True;Connect Timeout=30";

public Searching()

{

InitializeComponent();

listViewResults.View = View.Details;

listViewResults.Columns.Add("ID", 100);

listViewResults.Columns.Add("Name", 150);

listViewResults.Columns.Add("Type", 150);

listViewResults.Columns.Add("Recommended Age", 100);

listViewResults.Columns.Add("Average Playtime", 120);

listViewResults.Columns.Add("Price", 80);

listViewResults.Columns.Add("Quantity", 80);

}

private void button1\_Click(object sender, EventArgs e)

{

int searchId;

if (!int.TryParse(textBox1.Text, out searchId))

{

MessageBox.Show("Please enter a valid item ID.");

return;

}

string query = "SELECT \* FROM Board\_games WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", searchId);

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

listViewResults.Items.Clear();

if (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Id"].ToString());

item.SubItems.Add(reader["Name"].ToString());

item.SubItems.Add(reader["Type"].ToString());

item.SubItems.Add(reader["Recomended age"].ToString());

item.SubItems.Add(reader["Average playtime"].ToString());

item.SubItems.Add(reader["Price"].ToString());

item.SubItems.Add(reader["Quantity"].ToString());

listViewResults.Items.Add(item);

}

else

{

MessageBox.Show("No item found with the specified ID.");

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error searching for item: " + ex.Message);

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

string searchName = textBox2.Text.Trim();

string query = "SELECT \* FROM Board\_games WHERE Name LIKE @Name";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", "%" + searchName + "%");

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

listViewResults.Items.Clear();

while (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Id"].ToString());

item.SubItems.Add(reader["Name"].ToString());

item.SubItems.Add(reader["Type"].ToString());

item.SubItems.Add(reader["Recomended age"].ToString());

item.SubItems.Add(reader["Average playtime"].ToString());

item.SubItems.Add(reader["Price"].ToString());

item.SubItems.Add(reader["Quantity"].ToString());

listViewResults.Items.Add(item);

}

reader.Close();

if (listViewResults.Items.Count == 0)

{

MessageBox.Show("No items found with the specified name.");

}

}

catch (Exception ex)

{

MessageBox.Show("Error searching for items: " + ex.Message);

}

}

}

private void button3\_Click(object sender, EventArgs e)

{

string searchType = textBox3.Text.Trim();

string query = "SELECT \* FROM Board\_games WHERE Type LIKE @Type";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Type", "%" + searchType + "%");

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

listViewResults.Items.Clear();

while (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Id"].ToString());

item.SubItems.Add(reader["Name"].ToString());

item.SubItems.Add(reader["Type"].ToString());

item.SubItems.Add(reader["Recomended age"].ToString());

item.SubItems.Add(reader["Average playtime"].ToString());

item.SubItems.Add(reader["Price"].ToString());

item.SubItems.Add(reader["Quantity"].ToString());

listViewResults.Items.Add(item);

}

reader.Close();

if (listViewResults.Items.Count == 0)

{

MessageBox.Show("No items found with the specified type.");

}

}

catch (Exception ex)

{

MessageBox.Show("Error searching for items: " + ex.Message);

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

int searchAge;

if (!int.TryParse(textBox4.Text.Trim(), out searchAge))

{

MessageBox.Show("Please enter a valid age.");

return;

}

string query = "SELECT \* FROM Board\_games WHERE [Recomended age] >= @Age";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Age", searchAge);

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

listViewResults.Items.Clear();

while (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Id"].ToString());

item.SubItems.Add(reader["Name"].ToString());

item.SubItems.Add(reader["Type"].ToString());

item.SubItems.Add(reader["Recomended age"].ToString());

item.SubItems.Add(reader["Average playtime"].ToString());

item.SubItems.Add(reader["Price"].ToString());

item.SubItems.Add(reader["Quantity"].ToString());

listViewResults.Items.Add(item);

}

reader.Close();

if (listViewResults.Items.Count == 0)

{

MessageBox.Show("No items found with the specified age.");

}

}

catch (Exception ex)

{

MessageBox.Show("Error searching for items: " + ex.Message);

}

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox1.Focus();

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox3.Focus();

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox2.Focus();

}

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox4.Focus();

}

}

}

}

* 1. Cart.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

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 WindowsFormsApp3

{

public partial class Cart : Form

{

private string connectionString = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\Kokito\\source\\repos\\WindowsFormsApp3\\WindowsFormsApp3\\Database1.mdf;Integrated Security=True;Connect Timeout=30";

private double totalPrice = 0;

public Cart()

{

InitializeComponent();

listViewCart.View = View.Details;

listViewCart.Columns.Add("Id", 100);

listViewCart.Columns.Add("Name", 150);

listViewCart.Columns.Add("Type", 100);

listViewCart.Columns.Add("Price", 100);

listViewCart.Columns.Add("Quantity in Cart", 100);

listViewCart.Columns.Add("Available Quantity", 100);

}

private double CalculateTotalPrice()

{

double total = 0;

foreach (ListViewItem item in listViewCart.Items)

{

double price = Convert.ToDouble(item.SubItems[3].Text);

int quantity = int.Parse(item.SubItems[4].Text);

total += price \* quantity;

}

return total;

}

private void button1\_Click(object sender, EventArgs e)

{

int itemId;

if (!int.TryParse(textBox2.Text, out itemId))

{

MessageBox.Show("Please enter a valid item ID.");

return;

}

bool itemInCart = false;

foreach (ListViewItem item in listViewCart.Items)

{

if (item.SubItems[0].Text == itemId.ToString())

{

int quantityInCart = int.Parse(item.SubItems[4].Text);

quantityInCart++;

item.SubItems[4].Text = quantityInCart.ToString();

itemInCart = true;

break;

}

}

if (!itemInCart)

{

string query = "SELECT \* FROM Board\_games WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", itemId);

try

{

connection.Open();

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

ListViewItem item = new ListViewItem(reader["Id"].ToString());

item.SubItems.Add(reader["Name"].ToString());

item.SubItems.Add(reader["Type"].ToString());

item.SubItems.Add(reader["Price"].ToString());

item.SubItems.Add("1");

item.SubItems.Add(reader["Quantity"].ToString());

listViewCart.Items.Add(item);

totalPrice = CalculateTotalPrice();

textBox1.Text = totalPrice.ToString();

}

else

{

MessageBox.Show("Item not found.");

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error adding item: " + ex.Message);

}

}

}

else

{

totalPrice = CalculateTotalPrice();

textBox1.Text = totalPrice.ToString();

}

}

private void button2\_Click(object sender, EventArgs e)

{

int itemId;

if (!int.TryParse(textBox2.Text, out itemId))

{

MessageBox.Show("Please enter a valid item ID.");

return;

}

ListViewItem itemToRemove = null;

foreach (ListViewItem item in listViewCart.Items)

{

if (item.SubItems[0].Text == itemId.ToString())

{

itemToRemove = item;

break;

}

}

if (itemToRemove != null)

{

// Check if the item has multiple quantities in the cart

int quantityInCart = int.Parse(itemToRemove.SubItems[4].Text);

if (quantityInCart > 1)

{

quantityInCart--;

itemToRemove.SubItems[4].Text = quantityInCart.ToString();

}

else

{

listViewCart.Items.Remove(itemToRemove);

}

totalPrice = CalculateTotalPrice();

textBox1.Text = totalPrice.ToString();

MessageBox.Show("Item removed from the cart.");

}

else

{

MessageBox.Show("Item not found in the cart.");

}

}

private void button3\_Click(object sender, EventArgs e)

{

listViewCart.Items.Clear();

totalPrice = 0;

textBox1.Text = totalPrice.ToString();

MessageBox.Show("Cart cleared.");

}

private void button4\_Click(object sender, EventArgs e)

{

foreach (ListViewItem item in listViewCart.Items)

{

int itemId = int.Parse(item.SubItems[0].Text);

int quantityInCart = int.Parse(item.SubItems[4].Text);

int availableQuantity = int.Parse(item.SubItems[5].Text);

if (quantityInCart > availableQuantity)

{

MessageBox.Show($"Insufficient stock for item with ID {itemId}. Please reduce the quantity in the cart.");

return;

}

else if (quantityInCart == availableQuantity)

{

MessageBox.Show($"The last item with ID {itemId} has been sold.");

}

UpdateQuantityInDatabase(itemId, quantityInCart);

}

listViewCart.Items.Clear();

totalPrice = 0;

textBox1.Text = totalPrice.ToString();

MessageBox.Show("Cart checked out.");

}

private void UpdateQuantityInDatabase(int itemId, int quantityToSubtract)

{

string query = "UPDATE Board\_games SET Quantity = Quantity - @QuantityToSubtract WHERE Id = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", itemId);

command.Parameters.AddWithValue("@QuantityToSubtract", quantityToSubtract);

try

{

connection.Open();

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Error updating quantity in database: " + ex.Message);

}

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

textBox1.Focus();

}

}

}

}