**MINISTRY OF EDUCATION OF THE REPUBLIC OF BELARUS**

**EDUCATIONAL INSTITUTION  
 «SUKHOI STATE TECHNICAL UNIVERSITY OF GOMEL»**

Automation and Information Systems Faculty

Department of Information Technology

information technology «object oriented design and programming »

Course Work

windows form application for accounting for the rental of equipement for winter skiing

Compiled by: student gr. IT-23 Ilyass Amrane

(signature, date)

Practice leader: Курочка Константин Сергеевич

(job title)

(signature, date)

Date of protection

Grade

Signatures of the commission members:

Gomel 2024

Contents

[IntroductionError: Reference source not found](#_Toc162214394)

[Chapter.01 Technologies for Solutions:Error: Reference source not found](#_Toc162214395)

[1.1 the Use of C# as the main Programming language](#_Toc162214396) 4

[1.2 the use of MSSQL DBMS as a DatabaseError: Reference source not found](#_Toc162214398)

[1.3 MS SQL ServerError: Reference source not found](#_Toc162214399)

[1.4 the use of ADO.NET TechnologyError: Reference source not found](#_Toc162214403)

Chapter.02 project architectre………………………………………………………………………14

2[.1 application logic Error: Reference source not found](#_Toc162214406)

2[.2 application classes Error: Reference source not found](#_Toc162214406)

2[.3 database structure Error: Reference source not found](#_Toc162214406)

2[.4 database relations Error: Reference source not found](#_Toc162214406)

Chapter.03 application testing and solutions…………………………………………………………..

3.0 [Conclusion](#_Toc162214413) …..30

Error: Reference source not foundError: Reference source not found

# **Introduction**

Creating a skiing company Windows Form application is essential to streamline operations, enhance customer experience, and improve overall efficiency. The relevance of this work topic lies in catering to the specific needs of both the company and its clients within the skiing industry, offering a tailored and user-friendly platform for managing reservations, tracking rental equipment and facilitating seamless communication. This application will not only boost productivity but also elevate the skiing experience for all stakeholders involved ,Developing a skiing company Windows Form application is crucial for modernizing business processes, optimizing resource management, and increasing customer satisfaction. By incorporating features such as online booking, real-time updates, personalized recommendations, and interactive UI, the application can significantly enhance the overall skiing experience for customers while helping the company operate more efficiently. This topic is relevant as it addresses the growing demand for tech-driven solutions in the outdoor recreation industry and underscores the importance of leveraging technology to stay competitive, attract customers, and enhance operational performance.

Error: Reference source not foundError: Reference source not found

[1.1 the Use of C# as the main Programming Error: Reference source not found](#_Toc162214396)

C# (pronounced as C-sharp) was introduced by Microsoft in 2000 as part of its .NET initiative. Since then, it has gained widespread adoption among developers due to its simple syntax, robust performance, rich set of libraries, and seamless integration with other Microsoft technologies. C# is widely used for building desktop applications, web applications, mobile apps, games, enterprise software, and more.

\*Advantages of Using C#:\*  
1. \*Versatility\*: C# is a multi-paradigm language that supports object-oriented, functional, generic, and component-oriented programming paradigms. This versatility allows developers to tackle a wide range of programming challenges.  
   
2. \*Productivity\*: With its rich set of libraries and tools, C# enables developers to write code more efficiently and with fewer lines, leading to increased productivity.  
   
3. \*Cross-platform Development\*: While traditionally associated with Windows development, C# has expanded its reach to other platforms through initiatives like .NET Core and Xamarin, enabling developers to build applications for Windows, macOS, Linux, iOS, and Android.  
   
4. \*Strongly Typed Language\*: C# is a strongly-typed language, which helps catch errors at compile time rather than runtime, leading to more robust and reliable code.  
   
5. \*Integrated Development Environment (IDE)\*: Developers can leverage Microsoft's Visual Studio IDE for C# development, which provides powerful features like code completion, debugging, and profiling tools.  
  
\*Applications of C#:\*  
  
1. \*Desktop Applications\*: C# is commonly used for developing Windows desktop applications using Windows Forms, Windows Presentation Foundation (WPF), or Universal Windows Platform (UWP).  
  
2. \*Web Applications\*: C# is a popular choice for building dynamic web applications using frameworks like ASP.NET MVC, ASP.NET Core, and Blazor.  
  
3. \*Mobile Apps\*: With Xamarin, developers can write cross-platform mobile apps using C#, sharing a significant amount of code across iOS and Android platforms.  
  
4. \*Game Development\*: Unity, a popular game development engine, supports C# as a scripting language, making it a preferred choice for building games.  
  
5. \*Enterprise Software\*: Many businesses utilize C# for creating enterprise applications, such as CRM systems, ERP solutions, and business intelligence tools.  
  
\*Best Practices for C# Development:\*  
  
1. \*Follow Coding Standards\*: Adhere to coding conventions like naming conventions, code formatting, and documentation to maintain readability and consistency across the codebase.  
   
2. \*Utilize Object-Oriented Design Concepts\*: Take advantage of object-oriented programming principles like encapsulation, inheritance, and polymorphism to write maintainable and scalable code.  
   
3. \*Error Handling\*: Implement robust error handling mechanisms using features like exceptions to gracefully handle runtime errors and prevent application crashes.  
   
4. \*Performance Optimization\*: Optimize code performance by leveraging asynchronous programming, efficient data structures, and profiling tools to identify and address bottlenecks.  
   
5. \*Security\*: Ensure data security by following best practices like input validation, output encoding, data encryption, and secure API communication protocols.  
  
\*Conclusion:\*  
In conclusion, C# stands out as a versatile and powerful programming language that caters to a wide range of application development needs. Its rich features, strong ecosystem, and broad community support make it an ideal choice for developers looking to build scalable, performant, and secure software solutions. By following best practices and leveraging the full potential of C#, developers can create high-quality applications that meet the demands of modern software development.

C# is a versatile and powerful programming language developed by Microsoft that offers numerous advantages and pros for developers. One of the key benefits of C# is its strong type system, which provides robust compile-time type checking to help catch errors early in the development process. This can lead to more reliable and stable code.  
  
Another advantage of C# is its integration with the .NET framework, which offers a wide range of libraries and tools for building diverse applications across different platforms. The extensive standard library of C# simplifies common tasks and accelerates development processes.  
  
Additionally, C# supports Object-Oriented Programming (OOP) principles, making it easier to organize and manage complex codebases. This allows developers to create scalable and maintainable software solutions with ease. Furthermore, C# provides features like inheritance, polymorphism, and encapsulation, enhancing code reusability and promoting modular design.  
  
C# also includes support for modern programming paradigms such as asynchronous programming, allowing developers to write efficient and responsive applications that can handle multiple tasks concurrently. Its LINQ (Language Integrated Query) feature simplifies data manipulation and querying, streamlining database operations.  
  
Furthermore, C# is a statically typed language, which helps improve code readability and maintainability by enforcing strict type rules. This can reduce the likelihood of runtime errors and enhance the overall quality of the codebase.

In addition to the benefits mentioned earlier, C# also boasts a vibrant and supportive developer community. This large community provides access to a wealth of resources, tutorials, forums, and libraries that can help developers learn and grow their skills in C#. Moreover, the availability of open-source projects and contributions further enrich the ecosystem, fostering collaboration and knowledge sharing among developers.  
  
C# is known for its high performance and efficiency, making it suitable for developing performance-critical applications such as games, simulations, and high-traffic web applications. Its Just-In-Time compilation and advanced memory management capabilities contribute to the overall speed and responsiveness of C# applications.  
  
Another advantage of C# is its cross-platform compatibility through technologies like .NET Core and Xamarin. Developers can leverage C# to build applications that run seamlessly on various operating systems, including Windows, macOS, and Linux. This flexibility allows developers to target a broader audience and reach users on different platforms with minimum effort.  
  
The tooling support for C# is top-notch, with robust Integrated Development Environments (IDEs) like Visual Studio and Visual Studio Code that offer rich features like code completion, debugging tools, and project management capabilities. These tools enhance developer productivity and streamline the development workflow, enabling developers to focus on writing code and building innovative solutions.  
  
C# also prioritizes security by offering features like strong data typing, exception handling, and built-in memory safety mechanisms that help prevent common security vulnerabilities. By following best practices and leveraging security features in C#, developers can build secure and resilient applications that protect sensitive data and ensure user privacy.  
  
Overall, C# stands out as a versatile, feature-rich, and developer-friendly language that offers a wide range of advantages and benefits for building modern software applications. Its strong community support, performance capabilities, cross-platform compatibility, excellent tooling, and emphasis on security make it an excellent choice for developers looking to create innovative and reliable solutions across various domains.

Error: Reference source not foundError: Reference source not found

[21:31, 16/05/2024] Lyes: Microsoft SQL Server (MSSQL) and Database Management Systems (DBMS) are crucial components in today's digital landscape. MSSQL is a powerful relational database management system developed by Microsoft that offers robust features for storing, retrieving, and managing data. It is widely used by organizations of all sizes due to its scalability, security, and performance capabilities.  
  
DBMS, on the other hand, is a software system that facilitates the creation and management of databases. It acts as an intermediary between the user and the database, ensuring efficient data storage, retrieval, and manipulation. DBMS provides a structured approach to data management, allowing users to define data relationships, enforce data integrity, and ensure data security.  
  
The combination of MSSQL and DBMS offers a comprehensive solution for managing data effectively. MSSQL provides a reliable platform for storing and organizing large volumes of data, while DBMS offers the tools and functionalities to interact with the data in a secure and efficient manner. Together, they form a strong foundation for building robust applications and supporting business operations.  
  
Organizations leverage MSSQL and DBMS to streamline their data management processes, improve decision-making capabilities, and enhance overall operational efficiency. By utilizing these technologies, businesses can ensure data integrity, optimize data access and retrieval, and mitigate potential risks associated with data management.  
  
In conclusion, MSSQL and DBMS play a vital role in modern data management practices, offering organizations the tools and capabilities needed to effectively store, retrieve, and manage data. Their combined strengths make them indispensable components in today's data-driven world, empowering businesses to make informed decisions and drive innovation through effective data management practices.  
[21:31, 16/05/2024] Lyes: Microsoft SQL Server (MSSQL) is known for its robust performance in handling complex database operations, ensuring data accuracy and integrity. It offers a wide range of features such as data encryption, high availability, and advanced security measures to protect sensitive data. Meanwhile, Database Management Systems (DBMS) provide a framework for efficiently organizing and accessing data, ensuring seamless interactions between applications and databases. Together, MSSQL and DBMS empower businesses to make informed decisions, optimize data-driven processes, and drive innovation through effective data management practices. Their combined capabilities form a solid foundation for scalable, secure, and reliable data management solutions across various industries.

Microsoft SQL Server (MSSQL) is a powerful relational database management system (RDBMS) developed by Microsoft that is widely used by organizations across various industries to store, manage, and retrieve data efficiently.

MSSQL databases offer robust features such as support for transactions, triggers, stored procedures, and indexing to ensure data integrity and optimize performance. With its scalability and high availability options,

MSSQL is suitable for both small businesses and large enterprises with high data volumes and complex querying needs. The comprehensive security features of MSSQL, including role-based security and encryption, help safeguard sensitive information from unauthorized access and ensure compliance with data protection regulations. Additionally, MSSQL provides tools for backup and recovery, replication, and data warehousing, making it a versatile solution for managing structured and unstructured data effectively. The user-friendly interface of MSSQL Management Studio facilitates database administration tasks, such as designing schemas, monitoring performance metrics, and fine-tuning queries for optimal results. Overall, MSSQL is a reliable and feature-rich DBMS that empowers organizations to streamline their data management processes and derive valuable insights to drive business growth and innovation.  
Microsoft SQL Server (MSSQL) stands out as a leading relational database management system (RDBMS) in the realm of enterprise data management due to its robust capabilities and sophisticated features. MSSQL databases are designed to handle vast amounts of structured and unstructured data efficiently, making them ideal for businesses of all sizes seeking a scalable and high-performance solution for storing and processing information. With its support for ACID properties,

transactions in MSSQL are reliable and maintain data integrity, ensuring that database operations are executed accurately and consistently. The use of triggers and stored procedures in MSSQL allows for automation of routine tasks and the enforcement of data integrity rules,

enhancing the overall reliability and efficiency of database operations. Indexing plays a crucial role in optimizing query performance in MSSQL by enabling rapid data retrieval based on defined keys, resulting in faster response times and improved user experience. Moreover,

MSSQL's advanced security features, such as encryption and fine-grained access controls, bolster data protection efforts and help organizations comply with stringent regulatory requirements in industries like finance and healthcare. The high availability options offered by MSSQL, including failover clustering and database mirroring, ensure continuous access to critical data and minimize downtime in the event of system failures. Administrators benefit from a suite of management tools provided by MSSQL, such as SQL Server Management Studio, which simplifies tasks like database design, performance tuning, and monitoring, enabling them to optimize database performance and troubleshoot issues effectively. With support for data warehousing, replication, and business intelligence tools like SQL Server Reporting Services and Analysis Services, MSSQL empowers businesses to leverage their data for strategic decision-making and gain valuable insights into their operations. In conclusion, MSSQL stands as a versatile and powerful DBMS that aids organizations in managing their data efficiently, securely, and with the flexibility needed to adapt to evolving business requirements and technological advancements.

Error: Reference source not foundError: Reference source not found

Microsoft SQL Server is a powerful relational database management system (RDBMS) that has been a staple in the industry for decades. With its robust features, user-friendly interface, and scalability, SQL Server has established itself as a top choice for organizations of all sizes. In this comprehensive report, we will delve into the usability, options, and key features that make Microsoft SQL Server a preferred database solution for many businesses.  
  
\*Usability:\*  
SQL Server is renowned for its ease of use, particularly for users familiar with the Microsoft ecosystem. The management tools provided with SQL Server, such as SQL Server Management Studio (SSMS) and Azure Data Studio, offer intuitive interfaces that simplify database administration tasks. These tools provide a wide range of functionalities, from designing databases and writing queries to monitoring server performance and configuring security settings.  
  
Additionally, SQL Server's query optimization capabilities help users write efficient SQL queries that retrieve data quickly and accurately. The rich set of built-in functions and stored procedures enable developers to perform complex operations without having to reinvent the wheel, thereby boosting productivity and reducing development time.  
  
\*Options:\*  
Microsoft SQL Server offers various deployment options to meet diverse business needs. Organizations can choose to deploy SQL Server on-premises, in the cloud, or in a hybrid environment. With the introduction of Azure SQL Database, a fully managed cloud database service, businesses can leverage the benefits of SQL Server without the overhead of managing infrastructure.  
  
SQL Server also supports high availability and disaster recovery solutions, such as failover clustering and Always On Availability Groups, ensuring data availability and minimizing downtime in the event of hardware or software failures. Moreover, SQL Server's support for backup and restore operations simplifies data protection and recovery processes, providing peace of mind to businesses handling critical data.  
  
\*Key Features:\*  
One of the standout features of SQL Server is its robust security capabilities. SQL Server offers encryption at rest and in transit, row-level security, dynamic data masking, and auditing features to help organizations comply with industry regulations and protect sensitive data from unauthorized access.  
  
Another key strength of SQL Server is its scalability. With support for parallel processing and in-memory capabilities, SQL Server can handle large volumes of data and deliver lightning-fast query performance for demanding workloads. The integration of machine learning and AI services through SQL Server Machine Learning Services empowers organizations to derive valuable insights from their data and improve decision-making.  
  
Furthermore, SQL Server's support for various programming languages, including T-SQL, R, Python, and .NET, enables developers to build sophisticated applications and analytics solutions within the SQL Server environment. The extensibility of SQL Server through custom extensions and third-party tools further enhances its functionality and adaptability to specific business requirements.  
  
In conclusion, Microsoft SQL Server stands out as a comprehensive and versatile RDBMS that offers a wide range of features, deployment options, and scalability for businesses across industries. Its user-friendly interface, robust security, and advanced capabilities make it a preferred choice for organizations looking to manage their data effectively and drive innovation through data analytics and insights.

Microsoft SQL Server is a versatile relational database management system known for its user-friendly interface, robust features, and scalability.

From its intuitive management tools to its deployment options and advanced capabilities, SQL Server is a top choice for businesses looking to efficiently manage their data and derive valuable insights. With comprehensive security features, high availability solutions, and support for various programming languages,

SQL Server continues to be a dependable solution for organizations of all sizes seeking a powerful database platform.

SQL Server, developed by Microsoft, is a relational database management system (RDBMS) that is widely used by organizations around the world for storing and retrieving data. It has been a popular choice for businesses due to its robust features, scalability, and integration with other Microsoft products. SQL Server supports various editions tailored to different needs, including Express, Standard, and Enterprise editions.  
  
One of the key features of SQL Server is its Transact-SQL language, which is an enhanced version of SQL featuring additional programming constructs for developing complex queries and stored procedures. This powerful language allows developers and database administrators to perform a wide range of operations on the database, such as retrieving and updating data, creating triggers, and defining constraints.  
  
SQL Server offers a range of tools for managing and monitoring databases, including SQL Server Management Studio (SSMS), a graphical user interface for interacting with the database, and SQL Server Profiler, a tool for capturing and analyzing events occurring in the database. These tools make it easier for administrators to optimize database performance, troubleshoot issues, and automate routine tasks.  
  
In terms of security, SQL Server provides robust authentication and authorization mechanisms to control access to the database. Database administrators can configure user permissions at various levels, such as server, database, and object levels, to ensure that only authorized users can view or modify data. SQL Server also offers features like Transparent Data Encryption (TDE) and Always Encrypted for securing sensitive data at rest and in transit.  
  
SQL Server supports high availability and disaster recovery solutions to minimize downtime and ensure data integrity. Features like Always On Availability Groups and Failover Clustering provide options for configuring redundant database environments that can fail over seamlessly in case of hardware or software failures. These features are essential for businesses that require continuous access to their data and minimal data loss in case of failures.  
  
For performance optimization, SQL Server offers tools like the Query Optimizer, which analyzes queries and suggests efficient execution plans, and the Database Tuning Advisor, which recommends indexes and other optimizations to improve query performance. Additionally, SQL Server provides features like in-memory OLTP and columnstore indexes for accelerating query processing and analytics on large datasets.  
  
SQL Server integrates seamlessly with other Microsoft products, such as Azure cloud services, Power BI for data visualization, and Visual Studio for application development. This integration enables organizations to build end-to-end solutions that leverage the capabilities of SQL Server alongside other Microsoft technologies, thereby enhancing productivity and efficiency.  
  
In conclusion, SQL Server is a powerful and versatile database management system that offers a wide range of features for storing, retrieving, and managing data. With its robust security, high availability, and performance optimization capabilities, SQL Server is a popular choice for organizations looking to build scalable and reliable database solutions.

SQL Server has additional features like SQL Server Reporting Services (SSRS) for creating and sharing reports, SQL Server Integration Services (SSIS) for data integration and ETL processes, and SQL Server Analysis Services (SSAS) for online analytical processing (OLAP) and data mining. These components extend the functionality of SQL Server and provide comprehensive solutions for business intelligence and data warehousing needs. With its rich set of tools and capabilities, SQL Server continues to be a leading choice for developpers.

Error: Reference source not foundError: Reference source not found

Error: Reference source not foundError: Reference source not found

It supports various data providers for popular databases like SQL Server, Oracle, MySQL, and more. Overall, ADO.NET is a versatile and widely-used technology for data access in .NET applications, offering a wide range of functionality to developers.

ADO.NET technology simplifies data access for developers by providing a flexible and scalable framework to interact with databases. Its disconnected data architecture allows for efficient data retrieval, manipulation,

and storage without the need for a continuous connection to the database server. ADO.NET supports features like data binding, data caching, and transaction management, making it ideal for building enterprise-level applications. Its integration with other .NET technologies like ASP.NET and WinForms allows for seamless development of data-driven applications. ADO.NET's ability to work with various data providers and handle large volumes of data efficiently makes it a preferred choice for building robust and high-performance applications.

**project architecture**

login name surname UI code:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Data.SqlClient;

using ClassLibrary1;

using System.Configuration;

namespace myfirstuiproject

{

public partial class Form1 : Form

{

[DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]

private static extern IntPtr CreateRoundRectRgn

(

int nLeftRect, // x-coordinate of upper-left corner

int nTopRect, // y-coordinate of upper-left corner

int nRightRect, // x-coordinate of lower-right corner

int nBottomRect, // y-coordinate of lower-right corner

int nWidthEllipse, // width of ellipse

int nHeightEllipse // height of ellipse

);

public Form1()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.None;

Region = System.Drawing.Region.FromHrgn(CreateRoundRectRgn(0, 0, Width, Height, 10, 10));

}

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

{

deleteAll();

Application.Exit();

}

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

{

SqlConnection con = new SqlConnection("Data Source=DESKTOP-DGKGMTV;" +

"Initial Catalog=login;Integrated Security=True;TrustServerCertificate=true");

con.Open();

SqlCommand cmd = new SqlCommand("insert into login(name) values('"+ textBox1.Text+ "')", con);

int i=cmd.ExecuteNonQuery();

if (i == 0)

{

MessageBox.Show("your data has been saved");

}

else

{

MessageBox.Show("your data has been saved");

}

if (string.IsNullOrWhiteSpace(textBox1.Text))

{

MessageBox.Show("You need to input your surname.");

}

if (string.IsNullOrWhiteSpace(textBox2.Text))

{

MessageBox.Show("You need to input your name.");

}

else

{

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

}

static string cnnString = ConfigurationManager.ConnectionStrings["myfirstuiproject.Properties.Settings.loginConnectionString"].ToString();

public void deleteAll()

{

SqlConnection con = new SqlConnection(cnnString);

con.Open();

string request = "delete FROM [manager].[dbo].[basket] where id <> 0";

SqlCommand c = new SqlCommand(request, con);

c.ExecuteNonQuery();

con.Close();

}

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

{

Form9 f9 = new Form9();

f9.Show();

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

}

}

}

basket system code:

using ClassLibrary1;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Configuration;

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 myfirstuiproject

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

foreach (Produit p in getlistmanager())

{

richTextBox1.Text += $" \t{p.Getname()} {p.Getprice()}$\n";

}

}

static string cnnString = ConfigurationManager.ConnectionStrings["myfirstuiproject.Properties.Settings.loginConnectionString"].ToString();

public List<Produit> getlistmanager()

{

List<Produit> produits = new List<Produit>();

Clothes clothes = new Clothes();

SqlConnection con = new SqlConnection(cnnString);

con.Open();

SqlCommand c = new SqlCommand("select \* from [manager].[dbo].[basket]", con);

SqlDataReader reader = c.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

int id = reader.GetInt32(0);

Produit produit = new Clothes(reader.GetString(1), reader.GetDouble(2)) ;

produits.Add(produit);

}

}

con.Close();

return produits;

}

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

{

deleteAll();

this.Close();

}

public void deleteAll()

{

SqlConnection con = new SqlConnection(cnnString);

con.Open();

string request = "delete FROM [manager].[dbo].[basket] where id <> 0";

SqlCommand c = new SqlCommand(request, con);

c.ExecuteNonQuery();

con.Close();

}

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

}

}

}

manager mode UI code:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Data.SqlClient;

using System.Configuration;

namespace myfirstuiproject

{

public partial class Form9 : Form

{

public Form9()

{

InitializeComponent();

}

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

{

this.Close();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

this.Close();

}

private void label4\_Click(object sender, EventArgs e)

{

}

static string cnnString = ConfigurationManager.ConnectionStrings["myfirstuiproject.Properties.Settings.loginConnectionString"].ToString();

SqlConnection con = new SqlConnection(cnnString);

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

{

string surname = textBox1.Text, name = textBox5.Text, renteditems = comboBox1.Text;

DateTime leaseduration = DateTime.Parse(dateTimePicker1.Text);

con.Open();

string request = "insert into [manager].[dbo].[managertab] (surname,name,renteditems,leaseduration) values(@surname, @name, @renteditems, @leaseduration)";

SqlCommand c = new SqlCommand(request, con);

c.Parameters.AddWithValue("surname", surname);

c.Parameters.AddWithValue("name", name);

c.Parameters.AddWithValue("renteditems", renteditems);

c.Parameters.AddWithValue("leaseduration", leaseduration);

c.ExecuteNonQuery();

MessageBox.Show("Insert complete");

getlistmanager();

}

void getlistmanager()

{

SqlCommand c = new SqlCommand("select \* from [manager].[dbo].[managertab]", con);

SqlDataAdapter sd = new SqlDataAdapter(c);

DataTable dt = new DataTable();

sd.Fill(dt);

dataGridView1.DataSource = dt;

}

private void Form9\_Load(object sender, EventArgs e)

{

getlistmanager();

}

}

}

one of the UI containing products:

using ClassLibrary1;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Configuration;

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 myfirstuiproject

{

public partial class Form5 : Form

{

public Form5()

{

InitializeComponent();

}

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

{

this.Hide();

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

Clothes clothes = new Clothes();

clothes.Name = "skiing pants";

clothes.Price = 15;

insert(clothes);

MessageBox.Show("item added");

}

private void pictureBox7\_Click(object sender, EventArgs e)

{

Clothes clothes = new Clothes("the north face bundle",10);

MessageBox.Show("item added");

insert(clothes);

}

private void pictureBox6\_Click(object sender, EventArgs e)

{

Clothes clothes = new Clothes();

clothes.Name = "skiing boots";

clothes.Price = 20;

insert(clothes);

MessageBox.Show("item added");

}

static string cnnString = ConfigurationManager.ConnectionStrings["myfirstuiproject.Properties.Settings.loginConnectionString"].ToString();

public void insert(Produit produit)

{

SqlConnection con = new SqlConnection(cnnString);

con.Open();

string request = "insert into [manager].[dbo].[basket] values( @name, @Price)";

SqlCommand c = new SqlCommand(request, con);

c.Parameters.AddWithValue("name", produit.Getname());

c.Parameters.AddWithValue("Price", produit.Getprice());

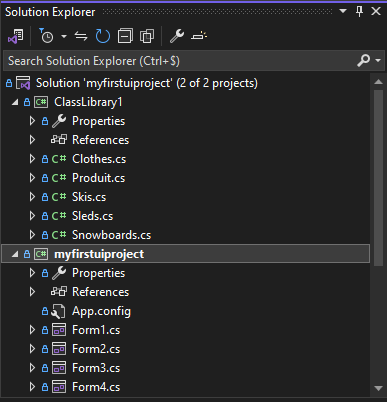
c.ExecuteNonQuery();

MessageBox.Show("Insert complete");

}

}

}



a photo of all classes used in the project

some of the classes code:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibrary1

{

public class Clothes : Produit

{

public Clothes()

{

}

public Clothes(string v1, double v2)

{

Name = v1;

Price = v2;

}

public string Name { get; set; }

public double Price { get; set; }

public int V1 { get; }

public string V2 { get; }

public string Getname()

{

return Name;

}

public double Getprice()

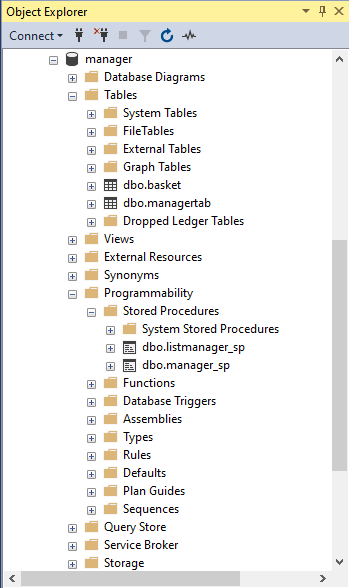
{

return Price;

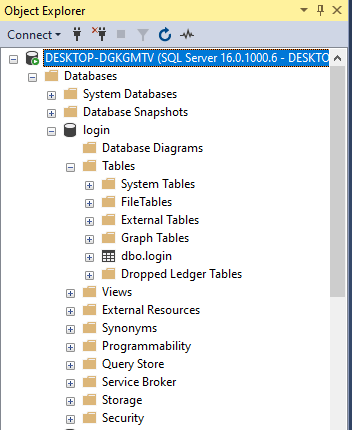
}

}

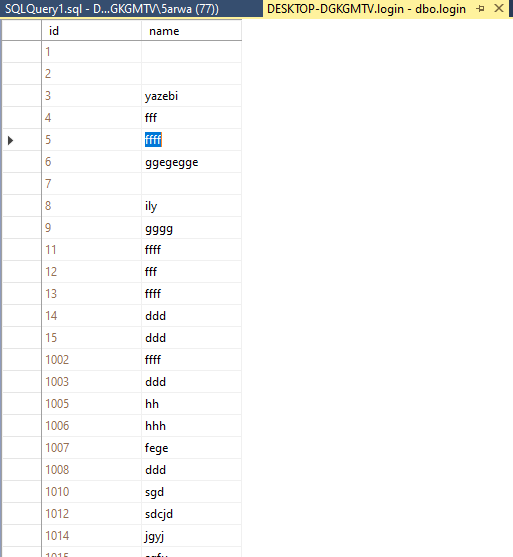
}



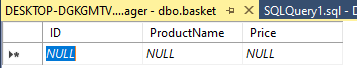
sql server object explorer view



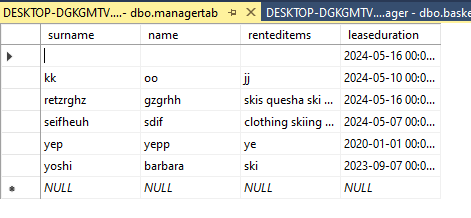
sql server object explorer view



sql login database



product database



manager tab database table

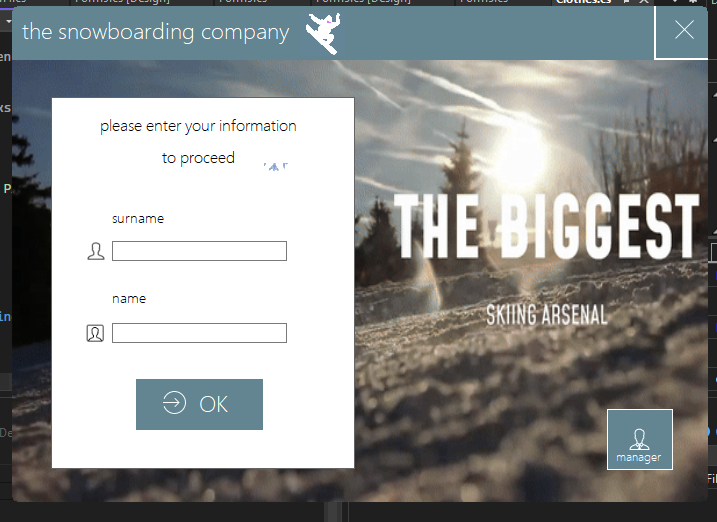
2Error: Reference source not foundError: Reference source not found

In the realm of database management, database relations play a vital role in organizing and structuring data efficiently. SQL Server, developed by Microsoft, is a popular relational database management system that leverages the power of SQL (Structured Query Language) to manage and manipulate data stored in relational databases. Understanding database relations in SQL Server is crucial for database designers, developers, and administrators to ensure data integrity and optimize performance.  
  
SQL Server follows the principles of relational database management, where data is organized into tables, and relationships between these tables are established using keys. The primary keys uniquely identify each record in a table, while foreign keys establish relationships between tables by referencing the primary key of another table. These relationships help maintain data consistency and integrity by enforcing referential integrity constraints.  
  
There are various types of relationships that can exist between tables in a SQL Server database, including one-to-one, one-to-many, and many-to-many relationships. In a one-to-one relationship, each record in one table is related to exactly one record in another table. In a one-to-many relationship, each record in one table can be related to multiple records in another table. In a many-to-many relationship, multiple records in one table can be related to multiple records in another table, typically using a junction table.  
  
Normalization is a key concept in database design that aims to reduce data redundancy and improve data integrity. By organizing data into separate tables and establishing relationships between them, normalization helps prevent anomalies such as insertion, update, and deletion anomalies. SQL Server provides tools and features to help database designers normalize their database schema effectively, such as normalization forms and design best practices.  
  
In SQL Server, relationships between tables are defined using constraints, such as foreign key constraints, which enforce rules for maintaining data consistency. When data modifications are made in a table, these constraints ensure that related data in other tables remains valid and consistent. SQL Server also supports cascading actions on foreign key constraints, allowing for automatic updates or deletions in related tables when changes are made to primary key values.  
  
Optimizing database relations in SQL Server is essential for improving query performance and overall system efficiency. By properly indexing columns involved in joins and relationships, database administrators can enhance query execution speed and reduce resource consumption. SQL Server provides various indexing options, such as clustered indexes and non-clustered indexes, to help optimize query performance based on the specific access patterns and workload requirements.  
  
Additionally, SQL Server offers tools like Query Execution Plans and Database Engine Tuning Advisor to analyze query performance and suggest optimizations for database relations. By monitoring and fine-tuning the database schema, indexing strategies, and query execution plans, SQL Server users can achieve better performance and scalability for their applications.  
  
In conclusion, database relations in SQL Server play a critical role in structuring and organizing data to ensure data integrity, consistency, and performance. Understanding the various types of relationships, normalization principles, constraints, and optimization techniques in SQL Server is essential for designing efficient and well-performing database systems. By mastering database relations in SQL Server, database professionals can effectively manage and leverage the power of relational database management systems for their data management needs.

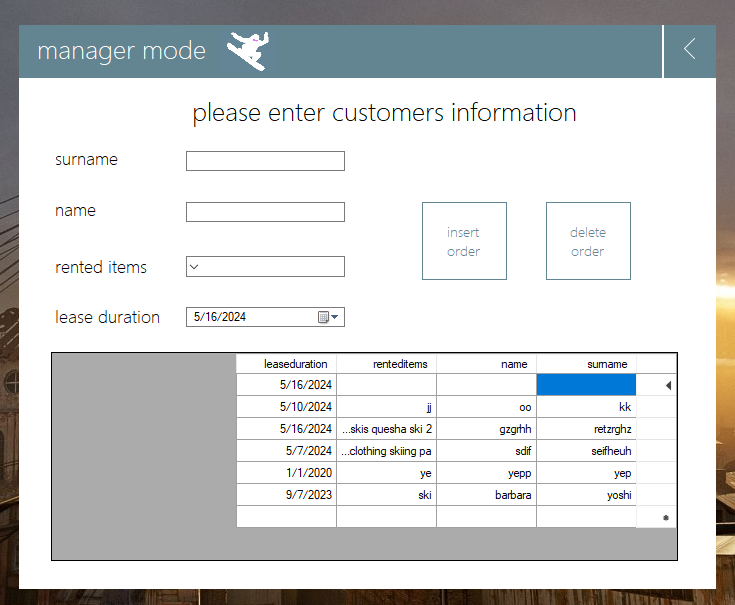


An example about database relations

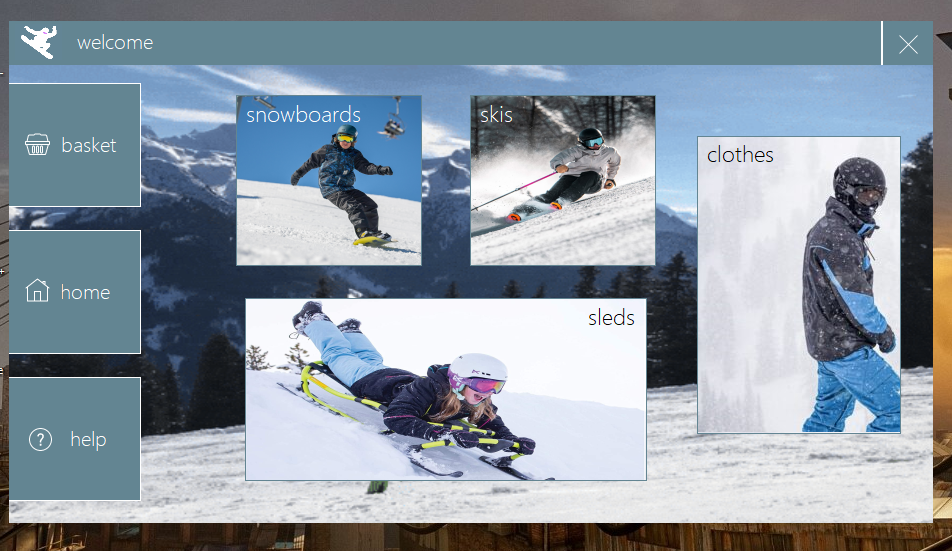
**application testing and solutions**



user UI format



manager mode UI



user friendly shopping UI

**conclusion**

In conclusion, diving into the realm of software development with C#, SQL databases, DBMS, ASP.NET, JavaScript, and Windows Forms applications offers a multifaceted learning experience for programming students. By crafting a user-friendly skiing rental application, students can grasp the intricacies of back-end server interactions, data storage, manipulation, and retrieval, as well as creating intuitive front-end interfaces for seamless user experiences.   
  
Through working with C#, students gain a strong foundation in object-oriented programming, enabling them to build scalable and maintainable applications. SQL databases empower them to understand structured data management, querying, and optimization, crucial for developing robust data-driven applications. Learning about Database Management Systems (DBMS) equips students with the knowledge to design schemas, implement relationships between data, and ensure data integrity.  
  
Exploring ASP.NET allows students to delve into web development and server-side programming, enabling them to create dynamic, interactive web applications with powerful functionality. JavaScript, often paired with ASP.NET, enhances students' skills in client-side scripting, enabling them to create responsive and engaging user interfaces.

working with Windows Forms applications provides students with an insight into desktop application development, where they can harness the power of the .NET framework to build feature-rich applications for Windows environments.  
  
In the process of developing a skiing rental application that combines these technologies, programming students can gain valuable insights into the importance of planning, design, implementation, testing, and iteration. They can learn the significance of user experience design, error handling, performance optimization, and the importance of data security in application development.  
  
Overall, this comprehensive journey through C#, SQL databases, DBMS, ASP.NET, JavaScript, and Windows Forms applications offers programming students a rich learning experience that encompasses a wide array of skills and knowledge essential for succeeding in the complex and ever-evolving field of software development.

Delving into the intricacies of C#, SQL databases, DBMS, ASP.NET, JavaScript, and Windows Forms applications while developing a user-friendly skiing rental application presents programming students with a holistic learning experience.

By mastering C#, students understand object-oriented programming principles essential for building scalable applications. Working with SQL databases hones their skills in data management and querying, while learning about DBMS equips them with database design expertise.  
  
Exploring ASP.NET empowers students to create dynamic web applications, while JavaScript enhances their abilities in client-side scripting. Constructing Windows Forms applications provides insight into desktop development, leveraging the .NET framework for feature-rich applications. Creating a skiing rental application with these tools teaches students about planning, design, testing, user experience, error handling, optimization, and data security.

The amalgamation of these technologies not only equips programming students with a diverse skill set but also instills in them the essential practices and insights needed to excel in the constantly evolving domain of software development.

C# and SQL are two powerful technologies that serve distinct yet complementary roles in modern software development. C# is a versatile, high-level programming language developed by Microsoft, widely used for building various types of applications ranging from desktop to web and mobile. Its object-oriented approach, strong type system, and extensive standard library make it a preferred choice for many developers.

SQL (Structured Query Language) is the standard language for interacting with relational databases. It provides a robust and efficient way to manage and query data, making it a fundamental tool for storing, retrieving, and manipulating information in a structured manner. SQL's declarative nature allows developers to focus on what data they want rather than how to retrieve it, promoting efficiency and readability in database operations.  
  
When C# and SQL are used together, they form a powerful combination for creating sophisticated applications that leverage the capabilities of both technologies. C# can be used to build the application logic, user interfaces, and business processes, while SQL handles the data storage, retrieval, and manipulation aspects. This division of responsibilities between the frontend (C#) and backend (SQL) components helps in achieving a clean and scalable architecture for applications.

Moreover, the integration of C# and SQL allows developers to create data-driven applications that are dynamic, responsive, and efficient. By utilizing C# to communicate with SQL databases through ADO.NET, Entity Framework, or other data access frameworks, developers can seamlessly interact with data stored in databases, perform complex queries, and ensure data integrity throughout the application lifecycle.

Furthermore, the performance of applications built using C# and SQL can be optimized by implementing best practices such as query optimization, indexing, caching, and asynchronous programming techniques. These optimizations not only enhance the speed and responsiveness of the applications but also ensure scalability and reliability, especially when dealing with large volumes of data.

In conclusion, the combination of C# and SQL offers developers a robust and flexible toolkit for building sophisticated applications that effectively manage, process, and present data to users.

By leveraging the strengths of C# as a versatile programming language and SQL as a reliable database query language, developers can create high-performance, data-centric applications that meet the demands of modern software development. Their synergy opens up a world of opportunities for creating innovative and efficient solutions across various domains, solidifying their roles as indispensable technologies in the software development landscape.

C# and SQL stand out as pillars of modern software development, offering developers a robust set of tools to create powerful and efficient applications.

C# provides a versatile environment for building various types of software solutions, thanks to its rich feature set, strong typing system, and extensive standard library. By leveraging its object-oriented programming paradigm and versatility, developers can create complex applications that are scalable, maintainable, and high-performing.

SQL serves as the backbone for managing data in relational databases, offering a standardized language for querying, updating, and interacting with data.

Its declarative nature allows developers to express complex data operations in a clear and concise manner, promoting efficiency and readability in database-driven applications. SQL's ability to handle transactions, constraints, and indexing makes it an essential tool for ensuring data integrity and security in modern software systems.

When combined, C# and SQL form a powerful duo that enables developers to build sophisticated applications that seamlessly integrate application logic with data management capabilities. With C# handling the frontend and business logic and SQL managing the data storage and retrieval aspects, developers can create well-structured, data-driven applications that cater to a wide range of user needs.

The integration of C# and SQL opens up a world of possibilities for developers, allowing them to leverage the strengths of both technologies to create innovative solutions across various domains. Whether developing desktop applications, web applications, mobile apps, or enterprise systems, the combination of C# and SQL provides a solid foundation for building reliable, scalable, and efficient software solutions.

In essence, C# and SQL play pivotal roles in modern software development, offering a powerful combination of programming language and database query language to create robust, data-centric applications. By mastering these technologies and understanding their synergies, developers can unlock new opportunities for building high-performance software solutions that meet the demands of today's digital landscape.

When considering which languages students should learn, it is important to take into account various factors such as career prospects, industry demands, versatility, and personal interests. While there is no one-size-fits-all answer to this question, it is widely acknowledged that learning a diverse set of programming languages can greatly enhance one's skill set and employability in today's digital landscape.

Among the plethora of programming languages available, C# stands out as a particularly important language for students to learn for several compelling reasons. First and foremost, C# is a versatile and powerful language that is widely used in the development of software applications, especially for the Microsoft ecosystem. By mastering C#, students can build a wide range of desktop, web, and mobile applications, making them valuable assets in the tech industry.

C# is also known for its robust features and strong typing system, which promote good coding practices and help developers write clean, efficient, and scalable code. Moreover, C# is highly compatible with other languages and platforms, allowing developers to easily integrate their code with various technologies and frameworks, such as Unity for game development and Xamarin for cross-platform mobile app development.

Additionally, proficiency in C# can open doors to lucrative career opportunities in fields like software development, game development, web development, and enterprise application development. Many companies, particularly those in the finance, healthcare, and technology sectors, actively seek professionals with C# skills, making it a valuable asset for students looking to kickstart their careers in these industries.  
  
In conclusion, while the choice of programming languages to learn ultimately depends on individual goals and interests, mastering C# can provide students with a solid foundation in software development, enhance their job prospects, and equip them with the skills needed to thrive in the ever-evolving tech industry. By investing time and effort in learning C#, students can position themselves for success and make significant contributions to the world of technology and innovation.

Learning C# offers students a gateway to explore various programming paradigms, such as object-oriented programming and functional programming, enhancing their problem-solving abilities and fostering a deeper understanding of software development principles. The language's extensive support for tools and libraries, like the .NET framework, provides students with a rich ecosystem to build robust and scalable applications. Furthermore, as C# continues to evolve with new features and improvements, students who invest in mastering this language can stay ahead of the curve and adapt to emerging trends in the tech industry. Ultimately, learning C# can empower students to become versatile and proficient developers with the potential to make a significant impact in the field of software engineering.

Creating a Windows Form application using C# taught me the fundamentals of building user interfaces and implementing functionality using an object-oriented programming language. I learned how to design forms, handle user inputs, and manipulate controls to create a responsive application. Additionally, understanding data binding, event handling, and UI design principles were crucial aspects of the development process. Overall, this experience provided me with a solid foundation in developing desktop applications with C# and empowered me to continue exploring and expanding my skills in software development.

Creating a Windows Form application using C# was a practical learning experience that deepened my understanding of GUI programming and event-driven programming. I gained insights into working with controls, managing user interactions, and structuring code to enhance the application's functionality and usability. Moreover, I honed my skills in debugging, testing, and optimizing the application for better performance. This hands-on project not only sharpened my technical abilities in C# but also improved my problem-solving skills and attention to detail when developing software solutions.

**References**

\* - Microsoft Docs. (n.d.). C# Guide. Retrieved from https://docs.microsoft.com/en-us/dotnet/csharp/   
\* Albahari, J., & Albahari, B. (2017). C# 7.0 in a Nutshell: The Definitive Reference. O'Reilly Media.