Laboratory Practical Report

of

**Visual Programming with C#**

**(ICT ED 465)**

Submitted To

**TRIBHUVAN UNIVERSITY**

In Partial Fulfillment of the Requirements of the course

**B.Ed. ICTE 6th Semester**

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-214-54-2019

Under the guidance of

**Er. Santosh Dahal**

Lecturer

Sukuna Multiple Campus

**SUKUNA MULTIPLE CAMPUS**

Sundarharaincha-12, Morang, Nepal

2080

**CERTIFICATE**

This is to certify that the Laboratory Practical Report

of

**Visual Programming with C#**

**(ICT ED 465)**

In Partial Fulfillment of the Requirements of the course

**B.Ed. ICTE 6th Semester**

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-214-54-2019

is a bonafide record of experiments carried out by him/her under by guidance.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Er. Santosh Dahal

Lecturer

Sukuna Multiple Campus

Sundarharaincha-12, Morang

(Internal Examiner)

Submitted for the Final Examination on: 2080/08/20

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lecturer

(External Examiner

**Table of Contents**

[Introduction to Database Connections](#_Toc2026092691) 1

[Example 1:Connection and Retriving Data](#_Toc1823464094) 2

[Example 2:Inserting Example](#_Toc1823464094) 3

[Example 3: Using a LINQ to SQL data context and a lambda expression](#_Toc2086445230) 4

[Example 4: Updating Data](#_Toc194041786) 5

# **Introduction to Database Connections**

A database connection is a communication link between a client application and a database server. It allows the client application to send queries to the database and receive results. Database connections are essential for any application that needs to access or store data in a database.

Steps Involved in Establishing a Database Connection

The following are the general steps involved in establishing a database connection:

Import the necessary database libraries. The specific libraries will depend on the database type being used. For example, to connect to a MySQL database, you would import the MySQL Connector/NET library.

Create a connection string. A connection string is a string of text that contains the information needed to connect to the database, such as the database server address, the database name, and the user credentials.

connection string = "Server=localhost;Database=mydatabase;Uid=myusername;Pwd=mypassword";

Create a connection object. A connection object represents the connection to the database. You can create a connection object using the connection string.

SqlConnection connection = new SqlConnection(connectionString);

Open the connection. To start communicating with the database, you need to open the connection.

connection.Open();

**Example 1:** **Connecting and Retrieving Data**

using System;

using System.Data.SqlClient;

namespace DatabaseConnectionExample

{

class Program

{

static void Main(string[] args)

{

string connectionString = "Server=localhost;Database=mydatabase;User Id=myusername;Password=mypassword";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Customers";

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Console.WriteLine("{0} {1}", reader["CustomerID"], reader["CustomerName"]);

}

}

}

}

}

# **Example 2:Inserting Example**

using System;

using System.Data.SqlClient;

namespace DatabaseConnectionExample

{

class Program

{

static void Main(string[] args)

{

string connectionString = "Server=localhost;Database=mydatabase;User Id=myusername;Password=mypassword";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Customers (CustomerName, ContactName) VALUES (@CustomerName, @ContactName)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@CustomerName", "New Customer");

command.Parameters.AddWithValue("@ContactName", "John Doe");

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

Console.WriteLine("Record inserted successfully.");

}

}

}

}

}

# **Example 3: Using a LINQ to SQL data context and a lambda expression**

using System;

using System.Data.Linq;

using System.Linq;

namespace Example3

{

class Program

{

static void Main(string[] args)

{ string connectionString = "server=(local)\\SQLExpress;database=Northwind;integrated Security=SSPI;";

using (DataContext context = new DataContext(connectionString))

{

Table<Customer> customers = context.GetTable<Customer>();

var query = customers.OrderBy(c => c.CustomerID).Take(5);

foreach (var customer in query)

{ Console.WriteLine("{0}\t{1}\t{2}", customer.CustomerID, customer.CompanyName, customer.ContactName);

}

}

}

}

// Define a class to represent a customer entity

[Table(Name = "dbo.Customers")]

public class Customer

{

[Column(IsPrimaryKey = true)]

public string CustomerID { get; set; }

[Column]

public string CompanyName { get; set; }

[Column]

public string ContactName { get; set; }

}

}

# **Example 4: Updating Data**

using System;

using System.Data.SqlClient;

namespace DatabaseConnectionExample

{

class Program

{

static void Main(string[] args)

{

string connectionString = "Server=localhost;Database=mydatabase;User Id=myusername;Password=mypassword";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "UPDATE Customers SET ContactName = @ContactName WHERE CustomerID = @CustomerID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@CustomerID", 1);

command.Parameters.AddWithValue("@ContactName", "Jane Doe");

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

Console.WriteLine("Record updated successfully.");

}

}

}

}

}