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](#_Toc543469219) 1

[Steps of Database Connection in C#](#_Toc1520914349) 1

[Example 1: Connecting to SQL server Database](#_Toc514148271) 2

[Example 2: Connecting to a MySQL database using the MySQL Connector/NET library](#_Toc1052654782) 3

[Example 3: Connecting to a PostgreSQL database using the Npgsql library](#_Toc293360904) 4

[Example 4: Connecting to a SQLite database using the SQLite.Net library](#_Toc848647049) 5

[R](#_Toc794310559)

# **Introduction**

A database connection is a communication channel 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 of Database Connection in C#**

1. Import the Database Driver: Begin by importing the necessary database driver libraries into your application. The specific driver depends on the type of database you're using, such as MySQL Connector/NET for MySQL databases or Oracle.DataAccess.Client for Oracle databases.
2. Construct the Connection String: Create a connection string that encapsulates the essential information required to connect to the database. This string typically includes the database server address, database name, user credentials, and optional connection parameters.
3. Establish the Connection: Utilizing the connection string, establish a connection to the database using the appropriate connection object provided by the database driver. This object represents the active communication channel between the application and the database.
4. Execute SQL Queries: Once the connection is established, you can execute SQL queries to interact with the database. Use the connection object to create command objects, which represent the SQL statements you want to execute.
5. Process Results: After executing a query, process the retrieved results. For SELECT queries, use a data reader object to iterate through the returned rows and access the data. For UPDATE, INSERT, or DELETE queries, check the command object's return value to verify the operation's success.
6. Close the Connection: When you're finished interacting with the database, properly close the connection object to release resources and terminate the communication link. This step ensures efficient resource management and prevents unnecessary connections to the database.

# **Example 1: Connecting to SQL server Database**

using System.Data.SqlClient;

public class DatabaseConnectionExample {

public static void Main(string[] args) {

// Create a connection string

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

// Create a connection object

SqlConnection connection = new SqlConnection(connectionString);

// Open the connection

connection.Open();

// Execute a SQL query

SqlCommand command = new SqlCommand("SELECT \* FROM Customers", connection);

SqlDataReader reader = command.ExecuteReader();

// Read the data from the reader

while (reader.Read()) {

Console.WriteLine(reader["CustomerName"] + " " + reader["Email"]);

}

// Close the reader and connection

reader.Close();

connection.Close();

}

}

# **Example 2: Connecting to a MySQL database using the MySQL Connector/NET library**

using MySql.Data.MySqlClient;

public class DatabaseConnectionExample {

public static void Main(string[] args) {

// Create a connection string

string connectionString = "server=localhost;database=mydatabase;uid=myusername;password=mypassword";

// Create a connection object

MySqlConnection connection = new MySqlConnection(connectionString);

// Open the connection

connection.Open();

// Execute a SQL query

MySqlCommand command = new MySqlCommand("SELECT \* FROM Customers", connection);

MySqlDataReader reader = command.ExecuteReader();

// Read the data from the reader

while (reader.Read()) {

Console.WriteLine(reader["CustomerName"] + " " + reader["Email"]);

}

// Close the reader and connection

reader.Close();

connection.Close();

}

}

# **Example 3: Connecting to a PostgreSQL database using the Npgsql library**

using Npgsql;

public class DatabaseConnectionExample {

public static void Main(string[] args) {

// Create a connection string

string connectionString = "Host=localhost;Port=5432;Database=mydatabase;Username=myusername;Password=mypassword";

// Create a connection object

NpgsqlConnection connection = new NpgsqlConnection(connectionString);

// Open the connection

connection.Open();

// Execute a SQL query

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM Customers", connection);

NpgsqlDataReader reader = command.ExecuteReader();

// Read the data from the reader

while (reader.Read()) {

Console.WriteLine(reader["CustomerName"] + " " + reader["Email"]);

}

// Close the reader and connection

reader.Close();

connection.Close();

}

}

# **Example 4: Connecting to a SQLite database using the SQLite.Net library**

using SQLite;

public class DatabaseConnectionExample {

public static void Main(string[] args) {

// Create a connection to the SQLite database file

using (var connection = new SQLiteConnection("mydatabase.db")) {

// Create a table if it doesn't exist

connection.CreateTable<Customer>();

// Insert a new customer record

connection.Insert(new Customer {

Name = "John Doe",

Email = "johndoe@example.com"

});

// Query the database for all customers

var customers = connection.Table<Customer>().ToList();

// Print the customer data

foreach (var customer in customers) {

Console.WriteLine(customer.Name + " " + customer.Email);

}

}

}

}