Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**10**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **01** | **Create Complete calculator program using MVC** |
| **02** | **Create Basic CRUD Application using MVC** |
|  |  |

Submitted On:

**25 – 05 -2023**

(Date: DD/MM/YY)

**TASK NO 1 :Create Complete calculator program using MVC**

**SOLUTION:**

**public class CalculatorModel**

{

public int Add(int num1, int num2)

{

return num1 + num2;

}

public int Subtract(int num1, int num2)

{

return num1 - num2;

}

public int Multiply(int num1, int num2)

{

return num1 \* num2;

}

public int Divide(int num1, int num2)

{

return num1 / num2;

}

}

**public class CalculatorView**

{

public void DisplayResult(int result)

{

Console.WriteLine("Result: " + result);

}

}

**public class CalculatorController**

{

private CalculatorModel model;

private CalculatorView view;

public CalculatorController(CalculatorModel model, CalculatorView view)

{

this.model = model;

this.view = view;

}

public void PerformOperation(int num1, int num2, string operation)

{

int result = 0;

switch (operation)

{

case "+":

result = model.Add(num1, num2);

break;

case "-":

result = model.Subtract(num1, num2);

break;

case "\*":

result = model.Multiply(num1, num2);

break;

case "/":

result = model.Divide(num1, num2);

break;

}

view.DisplayResult(result);

}

}

class Program

{

**static void Main(string[] args)**

{

CalculatorModel model = new CalculatorModel();

CalculatorView view = new CalculatorView();

CalculatorController controller = new CalculatorController(model, view);

int num1, num2;

string operation;

Console.Write("Enter first number: ");

num1 = int.Parse(Console.ReadLine());

Console.Write("Enter second number: ");

num2 = int.Parse(Console.ReadLine());

Console.Write("Enter operation (+, -, \*, /): ");

operation = Console.ReadLine();

controller.PerformOperation(num1, num2, operation);

Console.ReadLine();

}}}

**OUTPUT:**

A picture containing text, screenshot, font

Description automatically generated

**TASK NO 2 : Create Basic CRUD Application using MVC**

**SOLUTION:**

public class Customer

{

public int ID { get; set; }

public string Name { get; set; }

public string Email { get; set; }

}

public class CustomerController

{

private List<Customer> customers = new List<Customer>();

public List<Customer> GetAllCustomers()

{

return customers;

}

public void CreateCustomer(Customer customer)

{

// Assign a unique ID to the new customer

customer.ID = customers.Count + 1;

// Add the new customer to the list

customers.Add(customer);

}

public Customer GetCustomerByID(int id)

{

// Find the customer by ID

Customer customer = customers.Find(c => c.ID == id);

return customer;

}

public void UpdateCustomer(Customer customer)

{

// Find the customer by ID and update its properties

Customer existingCustomer = customers.Find(c => c.ID == customer.ID);

if (existingCustomer != null)

{

existingCustomer.Name = customer.Name;

existingCustomer.Email = customer.Email;

}

}

public void DeleteCustomer(int id)

{

// Remove the customer from the list

Customer customer = customers.Find(c => c.ID == id);

if (customer != null)

{

customers.Remove(customer);

}

}

}

public class Program

{

public static void Main(string[] args)

{

CustomerController controller = new CustomerController();

bool exit = false;

while (!exit)

{

Console.WriteLine("Choose an operation: (C)reate, (R)ead, (U)pdate, (D)elete, (E)xit");

string input = Console.ReadLine();

switch (input.ToUpper())

{

case "C":

Console.WriteLine("Enter customer name:");

string name = Console.ReadLine();

Console.WriteLine("Enter customer email:");

string email = Console.ReadLine();

Customer newCustomer = new Customer { Name = name, Email = email };

controller.CreateCustomer(newCustomer);

Console.WriteLine("New customer created.");

break;

case "R":

List<Customer> allCustomers = controller.GetAllCustomers();

foreach (Customer customer in allCustomers)

{

Console.WriteLine($"ID: {customer.ID}, Name: {customer.Name}, Email: {customer.Email}");

}

break;

case "U":

Console.WriteLine("Enter customer ID to update:");

int updateId = int.Parse(Console.ReadLine());

Customer existingCustomer = controller.GetCustomerByID(updateId);

if (existingCustomer != null)

{

Console.WriteLine("Enter updated customer name:");

string updatedName = Console.ReadLine();

Console.WriteLine("Enter updated customer email:");

string updatedEmail = Console.ReadLine();

existingCustomer.Name = updatedName;

existingCustomer.Email = updatedEmail;

controller.UpdateCustomer(existingCustomer);

Console.WriteLine("Customer updated.");

}

else

{

Console.WriteLine("Customer not found.");

}

break;

case "D":

Console.WriteLine("Enter customer ID to delete:");

int deleteId = int.Parse(Console.ReadLine());

controller.DeleteCustomer(deleteId);

Console.WriteLine("Customer deleted.");

break;

case "E":

exit = true;

break;

default:

Console.WriteLine("Invalid input. Please try again.");

break;

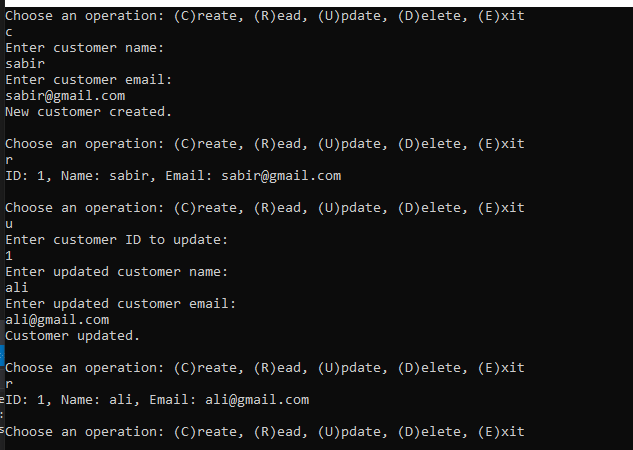
}

Console.WriteLine();

}

}

**OUTPUT:**

****