**Final Report: C#.ASP NET Programming language**

**A Simple project on a messaging App**

**Anusheel Khajuria**

School of Computer Science and Technology

**Under Supervision** of **Ms. Parveen Kaur**

School of Computer Science and Technology

Lovely Professional University

Punjab, India

**ABSTRACT**

Designing graphical user interfaces (GUIs) using C# and ASP.NET technologies can be challenging. This projectexplores alternatives using C# and ASP.NET.  This project uses the web development capabilities of ASP.NET to create a web messaging application. Users can send messages to anyone through their email ID.  
  
This project aims to provide a practical web-based user interface in C#. Using the ASP.NET library and Visual Studio IDE, the application allows users to send important information or messages with their bare hands.

**Keywords**: C#, ASP.NET, Visual Studio IDE, GUI

**ACKNOWLEDGEMENT**

**If these words are to be seen as signs of appreciation and recognition, let them thank me. The joy of accomplishing a task will not be enough without mentioning the strong cooperation of those who make the task possible, whose behavior is constant and whose support leads to all kinds of success. We would like to thank our manager Ms. We would like to thank Parveen Kaur for her advice, inspiration and suggestions that helped us develop this project. We would also like to thank our colleagues who contributed to the completion of this project.**

Anusheel Khajuria

**CONTENTS**

* **Acknowledgement**

1.Abstract

2. Introduction

3. System Analysis

4. Software Requirement Analysis

* Introduction
* Specific Requirements

5. Design

* System Design
* Detailed Design

6. Source Code (where ever applicable) or System Snapshots

1. **INTRODUCTION**

In today's hyperconnected world, instant communication is crucial. Messaging applications created with C# and ASP.NET are pioneers in meeting this need. C# is a versatile, useful programming language that offers powerful development tools and a large developer community. Together they form an important basis for submitting applications. They can expand to accommodate large stones. They enable instant communication, facilitate instant messaging, and encourage a sense of connection. Additionally, ASP.NET's security features ensure data confidentiality and communication integrity. Developers can integrate features such as train sharing, video calling, and group sharing. The complexity of this technology allows for innovative and engaging messaging. It plays an important role in communication and coordination.

Chatting or chatting is a way of using technology to bring people and ideas together despite geographical restrictions. This technology has been around for a long time but has gained support recently. Our design is an example of converse garçon. It has 2 functions: a guest function (runs on Stoner's computer) and a galleon function (runs on every computer on the network). To start drooling, the user must connect to the Gallon, where two drools can be made, one public (public communication for all connected grinders) and one private (only between 2 grinders) and Measurements are taken at the last moment.

**SYSTEM ANALYSIS**

**2.1 System Objectives to be achieved**

This device has many applications, including networking. Interactive applications connect two or more systems through an intranet or ad hoc. The application can be used for large-scale communication and meetings in large companies or schools to increase the level of collaboration. It also simplifies the complex socket concept, making it easier for users. The program will also have other capabilities that can be developed later, such as data transfer and voice chat.

**2.2 Approach**

The tool has been designed using C# (Windows Form Application)

**2.3 Methodology**

The user interacts with the tool using a GUI:

* The GUI operates in two forms, the List form & the chat form.
* The List form contains the names of all the systems connected to a network.
* The chat form makes the actual communication possible in the form of text.

**2.4** **System Architecture**

The framework can be used for simple messaging written in C# and ASP.NET. The presentation framework will usually be an ASP.NET web or mobile application (built using Xamarin for cross-platform integration) that handles user interaction and displays messages. The C#-based business logic layer manages user accounts, message creation, and returns. This layer is used with a database, such as a relational database (such as SQL Server), to permanently store data and information. Real-time updates can be obtained through technologies such as SignalR, which allows instant notifications or message changes without the need for a new page. Selecting an item from the list calls the application in the dialog box and establishes a connection between the host and the selected process.

**SOFTWARE REQUIREMENT ANALYSIS**

**3.1** **Introduction**

This document outlines the software requirement analysis for a basic messaging application built using C#. Determining functionality and user requirements is part of the software evaluation process for simple messaging in C#. User registration, access to user information, and security are important. The system should allow registered users to send and receive instant messages. Non-operational criteria such as speed and scalability need to be taken into account when retaining customers. Security analysis focusing on user authentication, authorization, and data encryption is important to protect user privacy and confidential information. Understanding user interface (UI) code is critical to providing a great user experience. This includes features like call lists, chat history, and notifications. By defining these rules, developers can create a safe, efficient, and effective framework for simple messaging.

* **Functionalities: -**  
  **User Management:**
  + Register using a username and password.
  + Log in and log out in the program.
  + Usernames should he unique.
  + Password difficulty criteria is specified.

**Contacts:**

* Search for others based on Email ID.
* Add other users.
* Contact List should be accessible.

**Messaging**

* Message to their contacts.
* History of messages must be visible.
* Can send basic file attachments.
* **Tools and Technologies: -**
* Programming Language: C#
* Development Framework: ASP.NET
* Database: SQL Server, MySQL

**3.2 Specific Requirements**

* Windows Form Application
* Hosting

**SYSTEM DESIGN**

**4.1 System Design**

The basic system design is made using the help of Windows Form application further connecting it with the backend program. This module manages the interface between the application and the end user. All user input (refresh, connection, chat) is captured here.

At a minimum, the following should be described –

1. **Processing within module: -**

We create user interfaces for programs that allow users to interact with tools. It has a large window and boxes that appear as a user menu. There are many controls such as switch controls and buttons that collect user input.

1. **Error Checking: -**
2. Errors occurring because of connection problems. Errors occurring due to incorrect input by the
3. user.

Errors occurring because of connection problems. Errors occurring due to incorrect input by the user.

4.2 **Detailed Design**

We demonstrate a Windows form application that makes communication graphically rich and user-friendly. This captures three GUI interfaces.

* **TEXTBOX:** Here, the user inserts the message that will be delivered to the remote user.
* **RICH TEXTBOX:** This module pastes sent messages under the host name and displays received messages under the remote user's name.
* **SEND button:** When it is hit, the message in the textbox is shown in the rich textbox under the host's name. It is also delivered over the network to the remote computer and displayed in the textbox under the sender's name.
* **USER NAME:** This is to add username of the application user.
* **PHONE NUMBER:** To add Phone number of the recipient.

**APPENDIX [A]**

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Net.WebRequestMethods;

namespace MessageApp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

using (System.Net.WebClient client = new System.Net.WebClient())

{

try

{

string url = "http://smsc.vianett.no/v3/send.ashx?" +

"src="+txtPhoneNumber.Text+"&" +

"dst="+txtPhoneNumber.Text+"&" +

"msg="+System.Web.HttpUtility.UrlEncode(txtMessage.Text, System.Text.Encoding.GetEncoding("ISO-8859-1"))+"" +

"username="+System.Web.HttpUtility.UrlEncode(txtUsername.Text )+"&" +

"password=" + System.Web.HttpUtility.UrlEncode(txtPassword.Text);

string result = client.DownloadString(url);

if (result.Contains("OK"))

MessageBox.Show("Your message has been successfully sent.", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);

else

MessageBox.Show("Message send failure.", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void label2\_Click(object sender, EventArgs e)

{

}

}

}

**APPENDIX [B]**

**OUTPUT**

One of the most important points in software development is to make it simple and easy to use. This has been a driving force throughout our software development process. The interface we provide is no exception. We created a simple but effective interface behind this idea.

**Fig 1.1 Message Interface**

