ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

ĐỒ ÁN CUỐI KỲ
Môn học: Lập trình mạng căn bản
Học kỳ II (2021 – 2022)

CHƯƠNG TRÌNH ĐỊNH DẠNG VĂN BẢN TỪ XA

Nhóm 3
Lớp: NT106.M21.ANTN
Trường: Đại học Công Nghệ Thông Tin
Giảng viên: Đỗ Thị Hương Lan

Thành phố Hồ Chí Minh, tháng 05 năm 2022

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

ĐỒ ÁN CUỐI KỲ
Môn học: Lập trình mạng căn bản
Học kỳ II (2021 – 2022)

CHƯƠNG TRÌNH
ĐỊNH DẠNG VĂN BẢN TỪ XA

Nhóm 3

Lớp: NT106.M21.ANTN

Trường: Đại học Công Nghệ Thông Tin

Giảng viên: Đỗ Thị Hương Lan

Thành phố Hồ Chí Minh, tháng 05 năm 2022

Danh sách nhóm

Môn học: Lập trình mạng căn bản

Nhóm: 3

Lớp: NT106.M21.ANTN

Tên đề tài: Chương trình định dạng văn bản từ xa

Họ tên	MSSV	Phân chia công việc
Võ Anh Kiệt	20520605	Build Server, Socket, SyncData
Nguyễn Bình Thục Trâm	20520815	Build Client, Interface, CorrectFeature
Nguyễn Bùi Kim Ngân	20520648	Database, Login, Crypto

LỜI CẢM ƠN

Trân trọng gửi lời cảm ơn đến cô Đỗ Thị Hương Lan đã tạo điều kiện cho nhóm chúng em có điều kiện thực hiện đồ án này. Trong học kỳ vừa qua, với sự giúp đỡ nhiệt tình của cô, chúng em đã nắm được nhiều kiến thức quan trọng trong việc nắm rõ các kiến thức cơ bản về việc lập trình mạng và có thể xây dựng một ứng dụng mạng cơ bản.

Đồng thời chúng em cũng gửi lời cảm ơn đến những thầy cô giảng viên, trợ giảng của trường Đại học Công Nghệ Thông Tin – Đại học Quốc Gia TPHCM đã hỗ trợ, tạo điều kiện cho chúng em thực hiện đồ án này.

Mục lục

PHẦN MỞ ĐẦU

Phần mềm định dạng văn bản từ xa được thiết kế dựa trên mô hình client – server với khả năng định dạng văn bản cơ bản như in đậm, in nghiêng, chỉnh sửa các loại văn bản như hành chính, tiểu luận, báo cáo… Chương trình cũng đã cập nhật thêm tính năng sửa lỗi và hỗ trợ để có thể thực hiện việc kiểm tra lỗi chính tả, những vấn đề trong soạn thảo văn bản để từ đó có thể khắc phục lỗi văn bản. Ngoài ra chương trình còn được tích hợp tính năng đồng bộ dữ liệu khi thực hiện công việc, từ đó có thể thực hiện quá trình đồng bộ dữ liệu từ hai hay nhiều client khi đang thực hiện định dạng, soạn thảo văn bản. Nhằm củng cố kiến thức về lập trình mạng cũng như là về việc lập trình C#, nhóm chúng em đã quyết định làm phần mềm định dạng văn bản từ xa với extension DevExpress.

PHẦN NỘI DUNG

Chương 1: Tổng quan

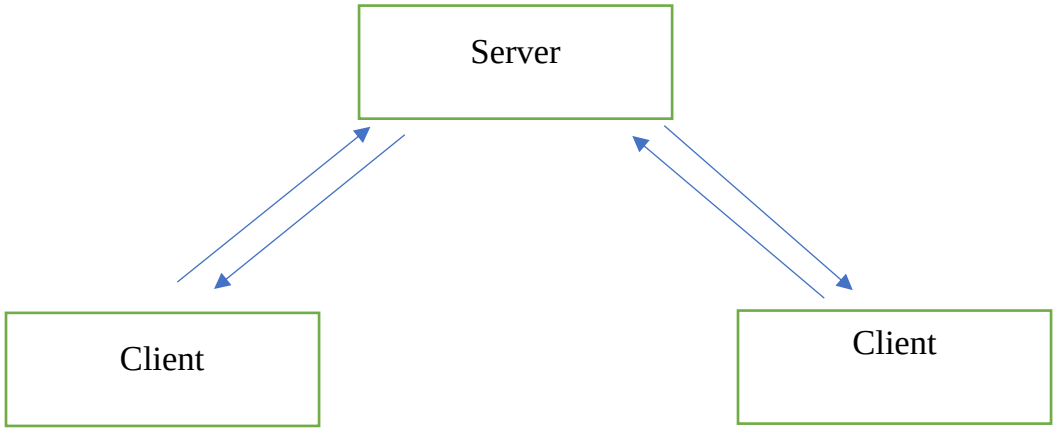
1.1. Yêu cầu

1.1.1. Các tính năng của chương trình

Thực hiện định dạng văn bản:

- Tạo mới, lưu file, cài mật mã cho file, điều chỉnh thuộc tính file.
- Copy, cut, paste, search, replace, in đậm, in nghiên, gạch chân, font chữ, cỡ chữ, màu chữ, màu highlight, màu shading.
- Tạo symbol bullet, number bullet, căn chỉnh văn bản.
- Thêm hình ảnh, table, header, footer, page number, page count, special symbol.
- Điều chỉnh layout của một trang.

Tính năng đồng bộ hoá



Tính năng sửa lỗi: CorrectFeature

Hệ thống đăng ký, đăng nhập.

Chương 2: Phân tích

2.1. Phân tích

2.1.1. Chương trình chính (mô phỏng theo word)

Giao diện chính và những công cụ định dạng văn bản cơ bản được xây dựng dựa trên các thư viện của extension DevExpress.

Kiểu dữ liệu chính: RTF, Document (DevExpress)

Document: Kiểu dữ liệu cho phép lấy văn bản từ RichEditControl và thực hiện các thao tác chỉnh sửa và có thể sử dụng để thể hiện văn bản trên RichEditControl (DevExpress).

RTF (Rich Text Format): Kiểu dữ liệu bao gồm nội dung văn bản và các định dạng đi kèm, là một loại dữ liệu thể hiện để chỉnh sửa của Document (DevExpress).


```
db\\newtblstyruls\\formshade\\sectd\\endnhere\\pard\\plain\\ql{\\fcs1\\af1\\alang
1033\\ltrch\\fcs0\\hich\\af2\\dbch\\af3\\loch\\f2\\fs22\\lang1033\\langfe1033\\la
ngnp1033\\langfenp1033\\b\\ul\\cf0
Bingtoni}\\fcs1\\af1\\alang1033\\ltrch\\fcs0\\hich\\af2\\dbch\\af3\\loch\\f2\\fs2
2\\lang1033\\langfe1033\\langnp1033\\langfenp1033\\b\\ul{\\*\\themedata
\r\n504b0304140002000800000021020115932 5ae000000270100000b0000005f72656c732f2e726
56c738d8f310ec2300c45af1279a76e1910424dbbb07460415c204a9d34a24da224457036068ec415
c8d82206164b5ffe7efffbfd7cd5ed7d1ad98d4234\r\nce72a88a121859e97a63358739a9cd1edaa
63ed3285276c4c1f8c8f2898d1c8694fc0131ca8126110be7c9e68d72611229cba0d10b79159a705b
963b0c4b06ac99aceb3984aeaf805d1e9efe613ba58ca4a393f34436fd88f87264b2089a128794998
48b\r\n791236b70c454e00864d8dab779b0f504b0304140002000800000021027f624aeb73000000
```


2.1.2. Tính năng đồng bộ hoá (mô phỏng theo github)

Đầu tiên, lấy mã RTF của văn bản gửi đồng bộ, sau đó chuyển thành mảng byte và gửi lên server. Và từ server phân phối lại cho tất cả các client.

2.1.3. Tính năng sửa lỗi (mô phỏng theo word)

Lấy chuỗi thông tin, qua function xử lý lỗi về điều chỉnh dấu và trả về kết quả là nội dung đã được sửa lỗi và format như cũ, thao tác này được thực hiện trên client.

2.1.4. Tính năng đăng ký, đăng nhập (mô phỏng theo clans and garena)

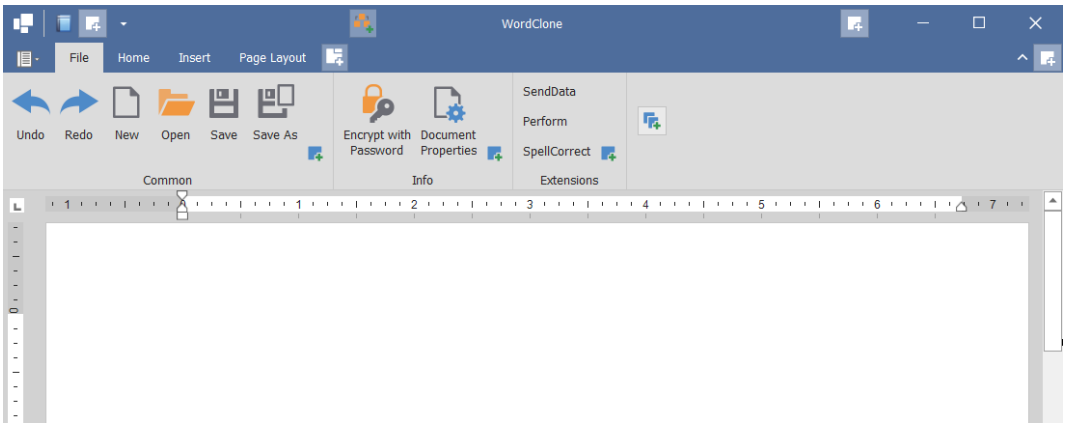
Sử dụng Microsoft Access, để thực hiện việc lưu data gồm account và password khi thực hiện việc đăng ký và đăng nhập.

Chương 3: Thiết kế và Chạy mô phỏng

3.1 Coding

3.1.1 Chương trình chính

Build những button cần thiết.



Set up các function cho các tính năng

```
0 references
private void iAbout_ItemClick(object sender, ItemClickEventArgs e)
{

}

1 reference
private void fileNewItem1_ItemClick(object sender, ItemClickEventArgs e)
{

}

1 reference
private void toggleFontBoldItem1_CheckedChanged(object sender, ItemClickEventArgs e)
{
}

1 reference
private void fileOpenItem1_ItemClick(object sender, ItemClickEventArgs e)
{

}

1 reference
private void Form1_Load(object sender, EventArgs e)
{

}
```

3.1.2 Tính năng đồng bộ hoá

Cài đặt các thư viện cần thiết để thiết lập giữa client và server

editorClass

```
namespace demoLibrary
{
    [Serializable]
    5 references
    public class editorClass
    {
        public editorType type = editorType.EMPTY;
    }
}
```

editorDataClass

```
namespace demoLibrary
{
    [Serializable]

    public class editorDataClass : editorClass
    {
        public string data = String.Empty;

        public editorDataClass(string data)
        {
            this.type = editorType.DATA;
            this.data = data;

        }
    }
}
```

editorType

```
namespace demoLibrary
{
    5 references
    public enum editorType
    {
        EMPTY,
        DATA,
        MESSAGE,
    }
}
```

Ở client thiết lập connector

```
public partial class connector
    {
        private MainForm form;
        private TcpClient tcpClient;
        private NetworkStream stream;
        private BinaryWriter writer;
        private BinaryReader reader;
        private Thread thread;

        public bool IsHandleCreated { get; private set; }



        public void SetupSendDataFunction(editorClass data)
        {
            MemoryStream memoryStream = new MemoryStream();
            BinaryFormatter binaryFormatter = new BinaryFormatter();
            binaryFormatter.Serialize(memoryStream, data);
            byte[] buffer = memoryStream.GetBuffer();

            writer.Write(buffer.Length);
            writer.Write(buffer);
            writer.Flush();
        }
        public void SendData(string text)
        {
            if (tcpClient.Connected == false)
            {
                return;
            }
            editorDataClass data = new editorDataClass(text);
            SetupSendDataFunction(data);
        }
        private delegate void AppendTextDelegate(string str);
        private void outputText(string text)
        {
            variable.pubRtf1 = text;
        }
        private void processSeverRespone()
        {
            try
            {
                BinaryFormatter binaryFormatter = new BinaryFormatter();
                int numberInputByte;
                //while ((numberInputByte = reader.ReadInt32()) != 0)
                while(true)
                {
                    numberInputByte = reader.ReadInt32();
                    byte[] bytes = reader.ReadBytes(numberInputByte);
                    MemoryStream memoryStream = new MemoryStream(bytes);
                    editorClass packet = binaryFormatter.Deserialize(memoryStream)
as editorClass;

                    switch (packet.type)
                    {
                        case editorType.DATA:
```

```
                            string message = ((editorDataClass)packet).data;
                            outputText(message);
                            break;
                        }
                    }
                }
                catch (Exception exc)
                {
                    outputText("Caution: " + exc.Message);
                }
            }
            public bool makeConnect(MainForm cform, string hostname, int port, string
nickname)
            {
                try
                {

                    form = cform;
                    tcpClient = new TcpClient();
                    tcpClient.Connect(hostname, port);
                    stream = tcpClient.GetStream();
                    writer = new BinaryWriter(stream, Encoding.UTF8);
                    reader = new BinaryReader(stream, Encoding.UTF8);
                    thread = new Thread(new ThreadStart(processSeverRespone));
                    thread.Start();
                }
                catch (Exception exc)
                {
                    outputText("Exception: " + exc.Message);
                    return false;
                }
                return true;
            }
            public void makeDisconnect()
            {
                try
                {
                    reader.Close();
                    writer.Close();
                    tcpClient.Close();
                    thread.Abort();
                }
                catch (Exception exc)
                {
                    outputText("Caution: " + exc.Message);
                }
                outputText("Disconnect");
            }
        }
```

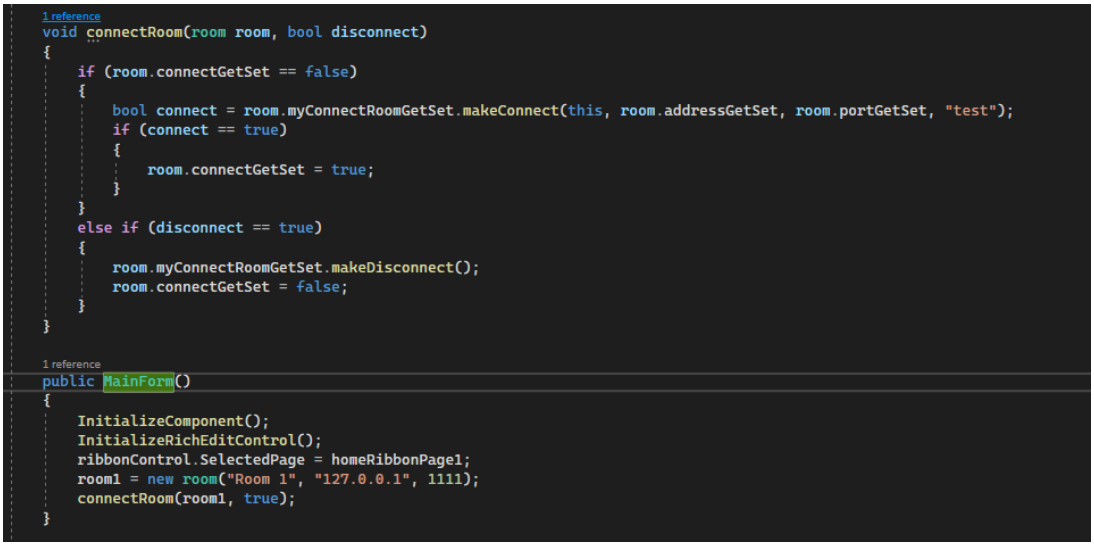
Thiết lập room

```
public class room
    {
        private string name;
        private string address;
        private int port;
        private bool connect;
```

```
private connector myConnectRoom;

public room(string name, string address, int port)
{
    this.name = name;
    this.address = address;
    this.port = port;
    this.connect = false;
    this.myConnectRoom = new connector();
}

public string nameGetSet
{
    get
    {
        return name;
    }
    set
    {
        name = value;
    }
}

public string addressGetSet
{
    get
    {
        return address;
    }
    set
    {
        address = value;
    }
}

public int portGetSet
{
    get
    {
        return port;
    }
    set
    {
        port = value;
    }
}

public bool connectGetSet
{
    get
    {
        return connect;
    }
    set
    {
        connect = value;
    }
}
```

```
    public connector myConnectRoomGetSet
    {
        get
        {
            return myConnectRoom;
        }
        set
        {
            myConnectRoom = value;
        }
}
```

Thiết lập kết nối khi bật chương trình



```
1 reference
void connectRoom(room room, bool disconnect)
{
    if (room.connectGetSet == false)
    {
        bool connect = room.myConnectRoomGetSet.makeConnect(this, room.addressGetSet, room.portGetSet, "test");
        if (connect == true)
        {
            room.connectGetSet = true;
        }
    }
    else if (disconnect == true)
    {
        room.myConnectRoomGetSet.makeDisconnect();
        room.connectGetSet = false;
    }
}

1 reference
public MainForm()
{
    InitializeComponent();
    InitializeRichEditControl();
    ribbonControl.SelectedPage = homeRibbonPage1;
    room1 = new room("Room 1", "127.0.0.1", 1111);
    connectRoom(room1, true);
}
```

Ở phía server

```
public class server
    {

        int portSimpleServer;
        TcpListener tcpListener;
        static List<client> clients = new List<client>();

        public server(string ipAddress, int port)
        {
            portSimpleServer = port;
            IPAddress ip = IPAddress.Parse(ipAddress);
            tcpListener = new TcpListener(ip, port);
        }

        public void start()
        {
            tcpListener.Start();

            Console.WriteLine("Port: " + Convert.ToString(portSimpleServer));

            while (true)
            {
                Socket socket = tcpListener.AcceptSocket();
                client client_index = new client(socket);
                clients.Add(client_index);
                client_index.Start();
            }
        }

        public void stop()
        {
            foreach (client element in clients)
            {
                element.Stop();
            }
            tcpListener.Stop();
        }

        public static void ServerSocketMethod(client dataFromClient)
        {
            try
            {
                BinaryFormatter binaryFormatter = new BinaryFormatter();
                Socket socket = dataFromClient.socketGetSet;
                NetworkStream stream = dataFromClient.streamGetSet;
                BinaryReader binaryReader = dataFromClient.readerGetSet;
                //dataFromClient.SendData(dataFromClient, "Successful Connection");

                int numberInputBytes;
                while ((numberInputBytes = binaryReader.ReadInt32()) != 0)
                {
                    byte[] bytes = binaryReader.ReadBytes(numberInputBytes);
                    MemoryStream memoryStream = new MemoryStream(bytes);
                    editorClass packet = binaryFormatter.Deserialize(memoryStream)
as editorClass;
```

```
                    switch (packet.type)
                    {
                        case editorType.DATA:
                            string message = ((editorDataClass)packet).data;
                            Console.WriteLine("<client>: " + message);
                            foreach (client element in clients)
                            {
                                element.SendData(dataFromClient,message);
                            }
                            break;
                    }
                }
            }
            catch (Exception exc)
            {
                Console.WriteLine("Caution: " + exc.Message);
            }
            finally
            {
                dataFromClient.Stop();
            }
        }
}
```

Thiết lập kết nối với client

```
public class client
    {
        public Socket socketGetSet { get; private set; }
        public NetworkStream streamGetSet { get; private set; }
        public BinaryReader readerGetSet { get; private set; }
        public BinaryWriter writerGetSet { get; private set; }

        private Thread thread;
        public client(Socket socket)
        {
            socketGetSet = socket;
            streamGetSet = new NetworkStream(socketGetSet, true);
            readerGetSet = new BinaryReader(streamGetSet, Encoding.UTF32);
            writerGetSet = new BinaryWriter(streamGetSet, Encoding.UTF32);
        }

        private void SetupSocketMethod()
        {
            server.ServerSocketMethod(this);
        }

        public void Start()
        {
            thread = new Thread(new ThreadStart(SetupSocketMethod));
            thread.Start();
        }
        public void Stop()
        {
            socketGetSet.Close();
            if (thread.IsAlive == true)
            {
                thread.Abort();
```

```
        }
    }
    public void SetupSendDataFunction(editorClass data)
    {
        MemoryStream memoryStream = new MemoryStream();
        BinaryFormatter binaryFormatter = new BinaryFormatter();
        binaryFormatter.Serialize(memoryStream, data);
        byte[] buffer = memoryStream.GetBuffer();

        writerGetSet.Write(buffer.Length);
        writerGetSet.Write(buffer);
        writerGetSet.Flush();
    }
    public void SendData(client dataFromClient, string text)
    {
        if (socketGetSet.Connected == false)
        {
            return;
        }
        editorDataClass data = new editorDataClass(text);
        SetupSendDataFunction(data);
    }
}
```

Cài đặt kết nối

3.1.3 Tính năng sửa lỗi

Code tính năng sửa lỗi

```
string AutoCorrect(string content)
        {
            if (content == "" || content == null) return "";
            content = content.Replace("..", "...");
            content = content.Replace("\" ", "\"");
            content = content.Replace("\' ", "\'");
            content = content.Replace("( ", "(");
            content = content.Replace(" )", ")");
            char firstchar = content[0];
            content = content.Remove(0, 1);
            content = content.Insert(0, Char.ToUpper(firstchar).ToString());

            int countdocs = 0, countcomma = 0, countUp = 0, openbefore = 0,
countspace = 0;
            string output = "";
            foreach (char chars in content)
            {
                if (chars == '.')
                {
                    if (countdocs < 3)
                    {
                        countdocs++;
                        output += chars;
                    }
                }
                else if ((chars == ';' || chars == ','))
                {
                    if (countcomma == 0)
                    {
                        countcomma++;
                        output += chars;
                    }
                }
                else if ((chars == '?' || chars == '!' || chars == ':' || chars ==
'-'))
                {
                    if (countUp == 0)
                    {
                        countUp++;
                        output += chars;
                    }
                }
                else if ((chars == '\"' || chars == '(' || chars == '\'') &&
openbefore == 0)
                {
                    {
                        ++openbefore;
                        if (countUp == 1)
                        {
                            if (output[output.Length - 1] != ' ') output += ' ';
                            output += chars;
                            countUp = 0;
                        }
```

```
                        else output += chars;
                    }
                }
                else if (chars == ' ')
                {
                    if (output[output.Length - 1] != ' ' && countspace == 0)
                    {
                        output += chars;
                        ++countspace;
                    }
                }
                else
                {
                    if (countdocs > 0)
                    {
                        if ((chars == '\"' || chars == ')' || chars == '\''))
                        {
                            openbefore = 0;
                            if (output[output.Length - 1] == ' ') output =
output.Remove(output.Length - 1, 1);
                            output += chars.ToString() + ' ';
                        }
                        else
                        {
                            if (output[output.Length - 1] == ' ') output +=
(Char.ToUpper(chars)).ToString();
                            else output += ' ' + (Char.ToUpper(chars)).ToString();
                        }
                        countdocs = 0;
                    }
                    else if (countUp == 1)
                    {
                        if ((chars == '\"' || chars == ')' || chars == '\''))
                        {
                            openbefore = 0;
                            if (output[output.Length - 1] == ' ') output =
output.Remove(output.Length - 1, 1);
                            output += chars.ToString() + ' ';
                        }
                        else
                        {
                            if (output[output.Length - 1] == ' ') output +=
(Char.ToUpper(chars)).ToString();
                            else output += ' ' + (Char.ToUpper(chars)).ToString();
                        }
                        countUp = 0;
                    }
                    else if (openbefore > 0 && (chars == '\"' || chars == ')' ||
chars == '\''))
                    {
                        openbefore = 0;
                        if (output[output.Length - 1] == ' ') output =
output.Remove(output.Length - 1, 1);
                        output += chars.ToString() + ' ';
                    }
                    else if (countcomma == 1)
                    {
```

```
                        if (output[output.Length - 1] == ' ') output +=
(Char.ToLower(chars)).ToString();
                        else output += ' ' + Char.ToLower(chars).ToString();
                    }
                    else if (output.Length > 0 && output[output.Length - 1] == '\"')
output += (Char.ToUpper(chars)).ToString();
                    else output += chars.ToString();
                    countspace = 0;
                    countdocs = 0;
                    countUp = 0;
                    countcomma = 0;
                }
            }
            output = output.Replace(" ,", ",");
            output = output.Replace(" .", ".");
            output = output.Replace(" ?", "?");
            output = output.Replace(" !", "!");
            return output;
        }
        private void ApplyFormatToSelectedText(DocumentRange sourceSelectedRange)
        {
            DocumentRange targetSelectedRange = richEditControl.Document.Selection;

            richEditControl.BeginUpdate();
            SubDocument targetSubDocument =
targetSelectedRange.BeginUpdateDocument();
            SubDocument subDocument = sourceSelectedRange.BeginUpdateDocument();

            DevExpress.XtraRichEdit.API.Native.CharacterProperties
targetCharactersProperties =
targetSubDocument.BeginUpdateCharacters(targetSelectedRange);
            DevExpress.XtraRichEdit.API.Native.CharacterProperties
sourceCharactersProperties = subDocument.BeginUpdateCharacters(sourceSelectedRange);
            targetCharactersProperties.Assign(sourceCharactersProperties);
            subDocument.EndUpdateCharacters(sourceCharactersProperties);
            targetSubDocument.EndUpdateCharacters(targetCharactersProperties);

            DevExpress.XtraRichEdit.API.Native.ParagraphProperties
targetParagraphProperties =
targetSubDocument.BeginUpdateParagraphs(targetSelectedRange);
            DevExpress.XtraRichEdit.API.Native.ParagraphProperties
sourceParagraphProperties = subDocument.BeginUpdateParagraphs(sourceSelectedRange);
            targetParagraphProperties.Assign(sourceParagraphProperties);
            subDocument.EndUpdateParagraphs(sourceParagraphProperties);
            targetSubDocument.EndUpdateParagraphs(targetParagraphProperties);

            sourceSelectedRange.EndUpdateDocument(subDocument);
            targetSelectedRange.EndUpdateDocument(targetSubDocument);
            richEditControl.EndUpdate();
        }
        private void SpellCorrect_ItemClick(object sender, ItemClickEventArgs e)
        {
            Document document = richEditControl.Document;

            CharacterProperties[] formatting = new CharacterProperties[9999];
            int countparagraph = document.Paragraphs.Count;

            for (int i = 0; i < countparagraph; i++)
```

```
            {
                string rtf = document.GetRtfText(document.Paragraphs[i].Range);
                Document subdoc = richEditControl1.Document;
                DocumentRange newRange = subdoc.InsertRtfText(subdoc.Range.End,
rtf);
                ParagraphProperties targetParagraphProperties =
subdoc.BeginUpdateParagraphs(newRange);
                ParagraphProperties sourceParagraphProperties =
document.BeginUpdateParagraphs(document.Paragraphs[i].Range);
                targetParagraphProperties.Assign(sourceParagraphProperties);
                document.EndUpdateParagraphs(sourceParagraphProperties);
                subdoc.EndUpdateParagraphs(targetParagraphProperties);
                string rft = subdoc.GetRtfText(newRange);

                string content = document.GetText(document.Paragraphs[i].Range);
                if (i == countparagraph - 1)
                    document.Replace(document.Paragraphs[i].Range,
AutoCorrect(content));
                else document.Replace(document.Paragraphs[i].Range,
AutoCorrect(content) + "\n");
                rft = document.GetRtfText(document.Paragraphs[i].Range);
                richEditControl.Document.Selection = document.Paragraphs[i].Range;

                ApplyFormatToSelectedText(newRange);
            }
                    }
```

3.1.4 Tính năng đăng ký, đăng nhập

Mã nguồn tính năng đăng nhập

```
public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }
        private void checkpass_CheckedChanged(object sender, EventArgs e)
        {
            if (checkpass.Checked)
            {
                txt_password.UseSystemPasswordChar = false;
            }
            else
            {
                txt_password.UseSystemPasswordChar = true;
            }
        }
        AesCryptoServiceProvider crpt = new AesCryptoServiceProvider();
        void setup()
        {
            string skey = "AXe8YwuIn1zxt3FPWTZFlAa14EHdPAdN9FaZ9RQWihc="; // 256 bit
key
            string siv = "bsxnWolsAyO7kCfWuyrnqg=="; // 128 bit iv
            // set up
```

```
            crpt.Mode = CipherMode.CBC;
            crpt.Padding = PaddingMode.PKCS7;
            crpt.Key = Convert.FromBase64String(skey);
            crpt.IV = Convert.FromBase64String(siv);
        }

        string Encrypt(string password)
        {
            setup();
            // encrypt
            ICryptoTransform cryptoTransform = crpt.CreateEncryptor();
            byte[] bpassword = Encoding.ASCII.GetBytes(password);
            byte[] encrypted = cryptoTransform.TransformFinalBlock(bpassword, 0,
bpassword.Length);
            return Convert.ToBase64String(encrypted); // max 20 ki tu = 44 ki tu
base64
        }

        OleDbConnection dbConnect = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=db_user.mdb");
        OleDbCommand dbCommand = new OleDbCommand();
        OleDbDataAdapter dbDataAdapter = new OleDbDataAdapter();

        private void btn_login_Click(object sender, EventArgs e)
        {
            string username = txt_username.Text;
            string password = txt_password.Text;
            password = Encrypt(password);

            dbConnect.Open();
            string login = "SELECT * FROM Table1 WHERE username= '" + username + "'
and password= '" + password + "'";
            dbCommand = new OleDbCommand(login, dbConnect);
            OleDbDataReader dbDataReader = dbCommand.ExecuteReader();

            if (username.Trim() == "" || password.Trim() == "")
            {
                MessageBox.Show("Username or password is empty", "Login failed",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                txt_username.Focus();
                dbConnect.Close();
            }
            else
            {
                if (dbDataReader.Read() == true)
                {
                    MessageBox.Show("Login successful", "Notification",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    this.Hide();
                    var tmpForm = new MainForm();
                    tmpForm.Closed += (s, args) => this.Close();
                    tmpForm.Show();
                    dbConnect.Close();
                }
                else
                {
                    MessageBox.Show("Username or Password is not correct", "Login
failed", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
                txt_username.Text = "";
                txt_password.Text = "";
                txt_username.Focus();
                dbConnect.Close();
            }
        }

    }

    private void checkpass_CheckedChanged_1(object sender, EventArgs e)
    {
        if (checkpass.Checked)
        {
            txt_password.UseSystemPasswordChar = false;
        }
        else
        {
            txt_password.UseSystemPasswordChar = true;
        }
    }

    private void btn_clear_Click(object sender, EventArgs e)
    {
        txt_username.Clear();
        txt_password.Clear();
        txt_username.Focus();
    }

    private void create_acc_Click(object sender, EventArgs e)
    {
        new RegisterForm().Show();
        this.Hide();
    }

    private void btn_quit_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Do you want to quit?", "Quitting",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
            Application.Exit();
    }
}
```

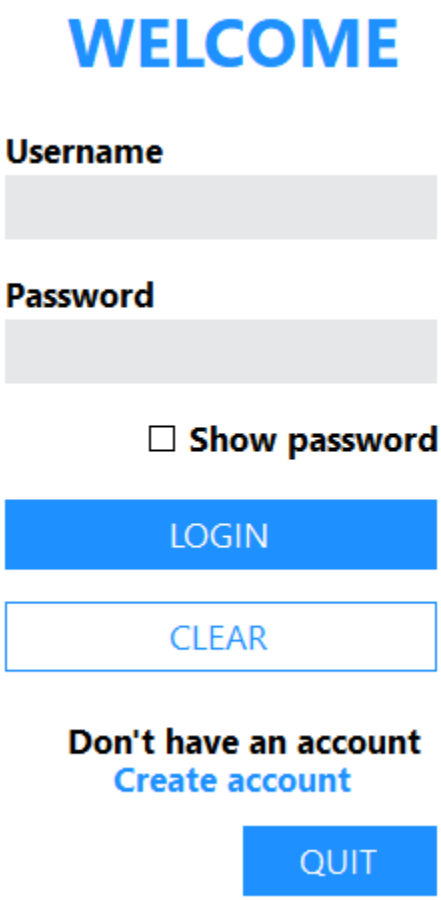
Mã nguồn tính năng đăng ký

```
public partial class RegisterForm : Form
    {
        public RegisterForm()
        {
            InitializeComponent();
        }
        OleDbConnection dbConnect = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=db_user.mdb");
        OleDbCommand dbCommand = new OleDbCommand();
        OleDbDataAdapter dbDataAdapter = new OleDbDataAdapter();

        private void checkpass_CheckedChanged(object sender, EventArgs e)
        {
            if (checkpass.Checked)
            {
                txt_password.UseSystemPasswordChar = false;
                txt_cfpassword.UseSystemPasswordChar = false;
            }
            else
            {
                txt_password.UseSystemPasswordChar = true;
                txt_cfpassword.UseSystemPasswordChar = true;
            }
        }
        public bool checkAccount(string ac)
        {
            // check username, password
            // chỉ cho phép nhập a-z, A-Z, 0-9, tối thiểu 3 và tối đa 20 kí tự
            return Regex.IsMatch(ac, "^[a-zA-Z0-9]{3,20}$");
        }

        AesCryptoServiceProvider crpt = new AesCryptoServiceProvider();
        void setup()
        {
            string skey = "AXe8YwuIn1zxt3FPWTZFlAa14EHdPAdN9FaZ9RQWihc="; // 256 bit
key
            string siv = "bsxnWolsAyO7kCfWuyrnqg=="; // 128 bit iv
            // set up
            crpt.Mode = CipherMode.CBC;
            crpt.Padding = PaddingMode.PKCS7;
            crpt.Key = Convert.FromBase64String(skey);
            crpt.IV = Convert.FromBase64String(siv);
        }

        string Encrypt(string password)
        {
            setup();
            // encrypt
            ICryptoTransform cryptoTransform = crpt.CreateEncryptor();
            byte[] bpassword = Encoding.ASCII.GetBytes(password);
            byte[] encrypted = cryptoTransform.TransformFinalBlock(bpassword, 0,
bpassword.Length);
            return Convert.ToBase64String(encrypted);
        }

        private void btn_register_Click(object sender, EventArgs e)
```

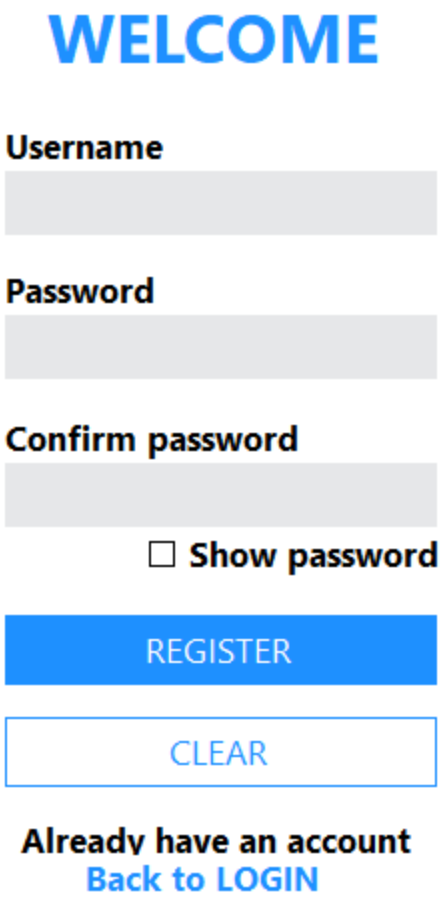
```
        {
            string username = txt_username.Text;
            string password = txt_password.Text;
            string cfpassword = txt_cfpassword.Text;

            if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) ||
string.IsNullOrEmpty(cfpassword))
            {
                MessageBox.Show("Some fields are empty");
            }
            else if (!checkAccount(username))
            {
                MessageBox.Show("Use capitalize/lowercase letter, number and 3-20
length", "Invalid username");
            }
            else if (!checkAccount(password))
            {
                MessageBox.Show("Use capitalize/lowercase letter, number and 3-20
length", "Invalid password");

            }
            else if (txt_password.Text != txt_cfpassword.Text)
            {
                MessageBox.Show("Password does not match", "Registration failed",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                txt_password.Clear();
                txt_cfpassword.Clear();
                txt_password.Focus();
            }
            else
            {
                password = Encrypt(password);
                dbConnect.Open();
                string register = "INSERT INTO Table1 VALUES ('" + username + "','"
+ password + "')";
                dbCommand = new OleDbCommand(register, dbConnect);
                dbCommand.ExecuteNonQuery();
                dbConnect.Close();
                txt_username.Text = "";
                txt_password.Text = "";
                txt_cfpassword.Text = "";
                if (MessageBox.Show("Do you want to login?", "Registration
successful", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    new LoginForm().Show();
                    this.Close();
                }
            }
        }

        private void btn_clear_Click(object sender, EventArgs e)
        {
            txt_username.Clear();
            txt_password.Clear();
            txt_cfpassword.Clear();
            txt_username.Focus();
        }
```

```
        private void back2login_Click(object sender, EventArgs e)
        {
            new LoginForm().Show();
            this.Close();
        }
}
```

Giao diện đăng ký đăng nhập



Giao diện đăng ký

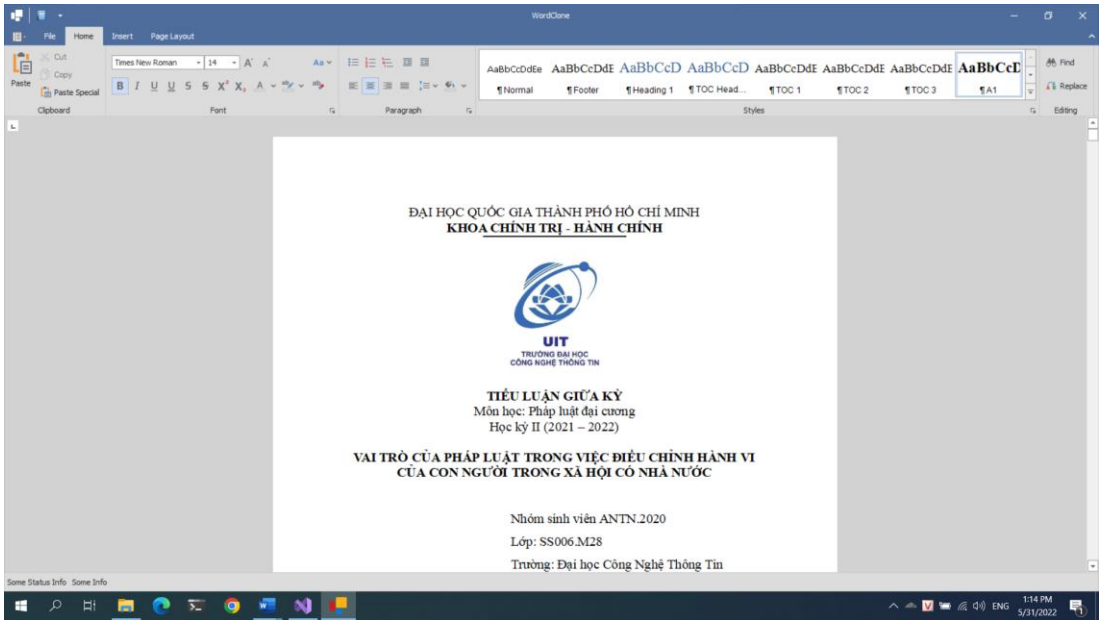


Lưu trữ database

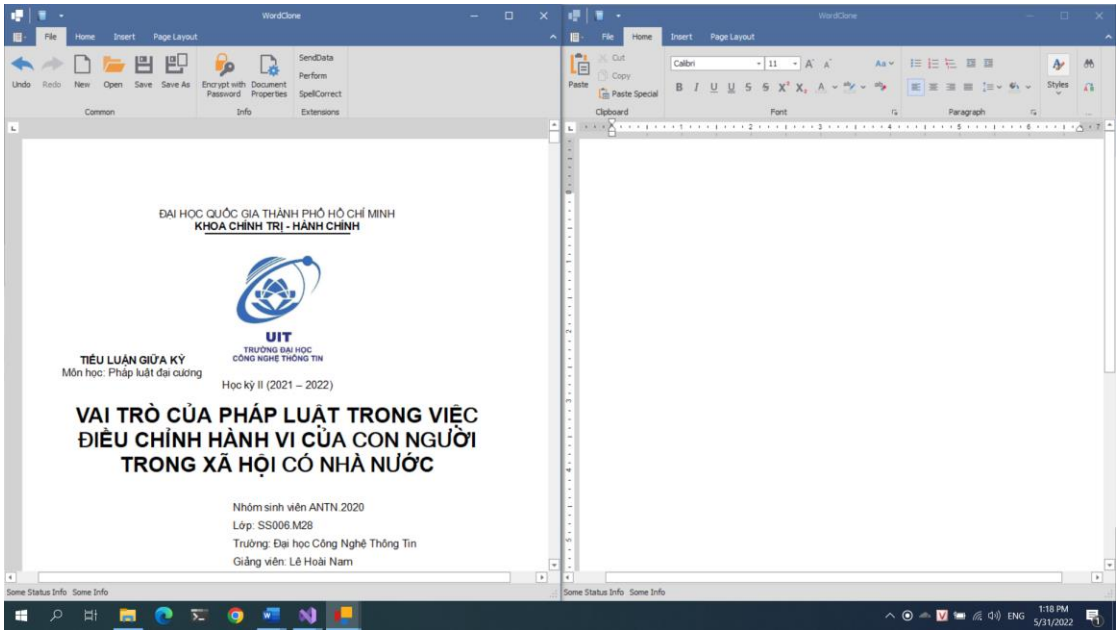
demo	qUHLJUFPK6YXnXjhrmIkQg==
kiett	qUHLJUFPK6YXnXjhrmIkQg==

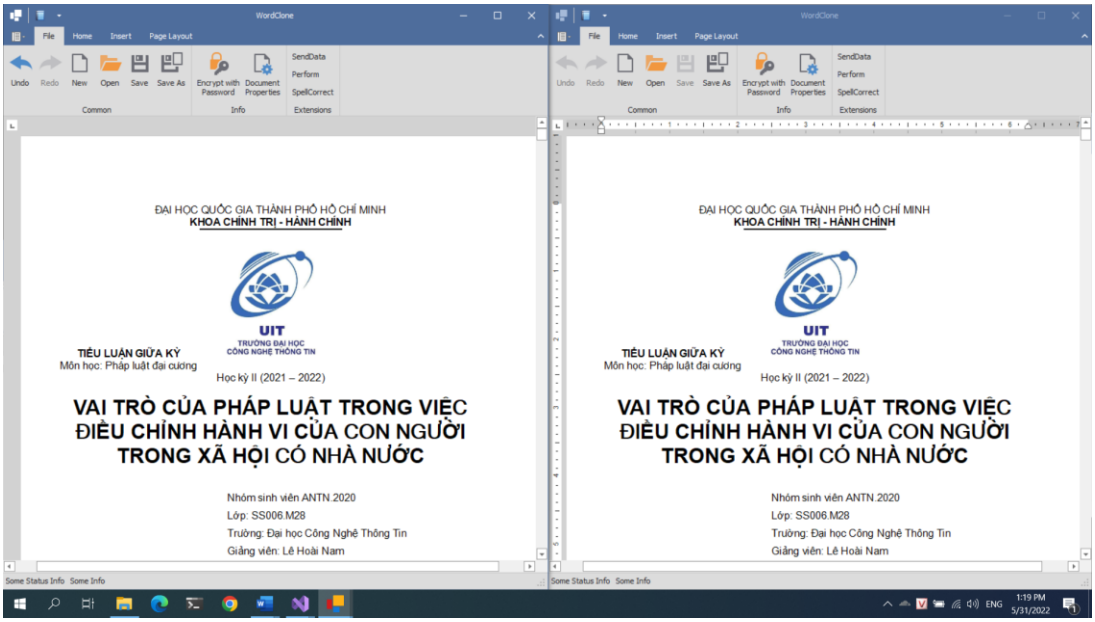
3.2 Chạy thử nghiệm

3.2.1 Chương trình chính



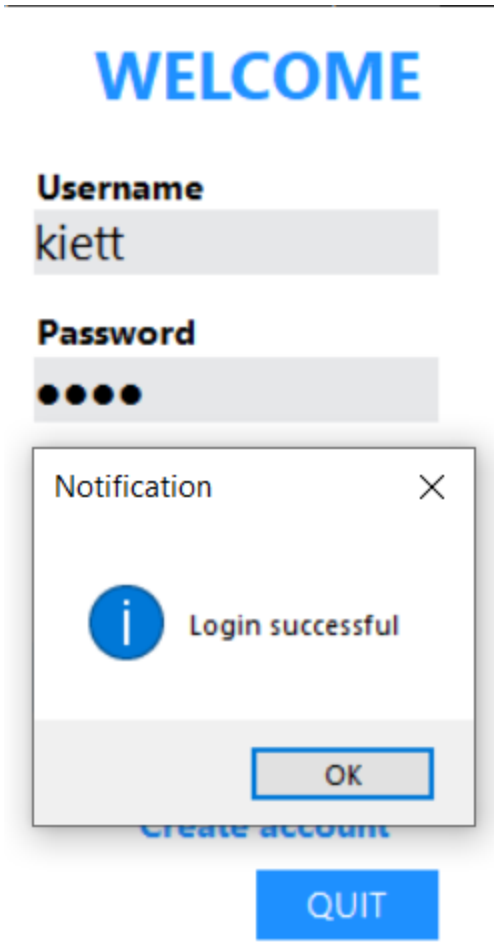
3.2.2 Tính năng đồng bộ hoá





3.2.3 Tính năng sửa lỗi

3.2.4 Tính năng đăng ký, đăng nhập


WELCOME
Username
kiett
Password
Notification
Login successful
OK
QUIT

WELCOME

Username

Password

Confirm password

☐ **Show password**

REGISTER

CLEAR

Already have an account
Back to LOGIN

Chương 4: Tổng kết và đánh giá

Tổng kết:

Đã hoàn thành được chương trình soạn thảo văn bản và định dạng, sử dụng .Net Framework và DevExpress

Thực hiện được tính năng đồng bộ hoá sử dụng socket, mô hình client – server.

Cài đặt được tính năng sửa lỗi

Cài đặt được hệ thống đăng ký, đăng nhập

Hạn chế và công việc trong tương lai:

Tính năng đồng bộ hoá và tính năng sửa lỗi chưa thực hiện được việc tự động hoá.

Chưa deploy trên internet