**Виконав: Мельник Б. В.; КН-922в**

**Лабораторна робота №12**

**Тема:** Створення багатовіконних застосунків. Частина 2.

**Мета роботи:** набути навички у створенні багатовіконних застосунків з більш розвиненим функціоналом.

**Індивідуальні завдання**

**Варіант №12**

**Завдання 1**

Додати до проекту форму ресстрації користувачів умовної системи із використанням реестраційної форми згідно до власного варіанту. Перед занесенням ресстраційних даних до відповідної таблиці, виконати перевірку (валідацію) реестраційних полів:

* UserName : лише числа та символи латиниці кількістю не більше 15 i щоб ім'я було унікальним (раніше не було в таблиці БД);
* Password : числа, символи латиниці та обов'язково спеціальні символи довжиною не менше 12, та його коректне повторення у відповідному полі;
* Email : коректне значения електронної поштової адреси

В разі наявності помилок повертатися до форми і відображати їх навпроти відповідних полів.

|  |  |
| --- | --- |
| № варіанту | Шаблон реєстраційної форми |
| 1 | 2 |
| 12. |  |

Валідацію полів форми зробити за допомогою регулярних виразів.

**Текст програми:**

***RegisterForm.cs:***

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class RegisterForm : Form

{

public RegisterForm()

{

InitializeComponent();

StyleComponents();

ClearErrorLabels();

}

private void StyleComponents()

{

// Кнопка реєстрації

registerButton.BackColor = Color.MediumSeaGreen;

registerButton.ForeColor = Color.White;

registerButton.FlatStyle = FlatStyle.Flat;

registerButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(registerButton, 15); // Закруглення країв кнопки

// Кнопка переходу до логіну

loginButton.BackColor = Color.Transparent;

loginButton.ForeColor = Color.MediumSeaGreen;

loginButton.FlatStyle = FlatStyle.Flat;

loginButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(loginButton, 15); // Закруглення країв кнопки

// Видалення рамок текстових полів та додавання підкреслення

nameTextBox.BorderStyle = BorderStyle.None;

emailTextBox.BorderStyle = BorderStyle.None;

phoneTextBox.BorderStyle = BorderStyle.None;

passwordTextBox.BorderStyle = BorderStyle.None;

confirmPasswordTextBox.BorderStyle = BorderStyle.None;

// Задання кольору для панелей під текстовими полями

namePanel.BackColor = Color.LightGray;

emailPanel.BackColor = Color.LightGray;

phonePanel.BackColor = Color.LightGray;

passwordPanel.BackColor = Color.LightGray;

confirmPasswordPanel.BackColor = Color.LightGray;

}

private void RoundButtonEdges(Button button, int radius)

{

// Створення округленої форми для кнопки

GraphicsPath path = new GraphicsPath();

path.AddArc(0, 0, radius, radius, 180, 90);

path.AddArc(button.Width - radius, 0, radius, radius, 270, 90);

path.AddArc(button.Width - radius, button.Height - radius, radius, radius, 0, 90);

path.AddArc(0, button.Height - radius, radius, radius, 90, 90);

path.CloseAllFigures();

button.Region = new Region(path); // Призначення округленої форми

}

private void ClearErrorLabels()

{

// Очищення повідомлень про помилки

nameErrorLabel.Text = "";

emailErrorLabel.Text = "";

phoneErrorLabel.Text = "";

passwordErrorLabel.Text = "";

confirmPasswordErrorLabel.Text = "";

}

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

{

ClearErrorLabels(); // Очищення старих помилок

// Зчитування даних з текстових полів

string userName = nameTextBox.Text.Trim();

string email = emailTextBox.Text.Trim();

string phone = phoneTextBox.Text.Trim();

string password = passwordTextBox.Text;

string confirmPassword = confirmPasswordTextBox.Text;

// Перевірка правильності введених даних

bool isValid = ValidateFields(userName, email, phone, password, confirmPassword);

if (isValid)

{

bool isUnique = true;

// Перевірка унікальності імені користувача

if (IsUsernameTaken(userName))

{

nameErrorLabel.Text = "Ім'я користувача вже зайняте.";

isUnique = false;

}

// Перевірка унікальності електронної пошти

if (IsEmailTaken(email))

{

emailErrorLabel.Text = "Електронна пошта вже використовується.";

isUnique = false;

}

// Перевірка унікальності номера телефону

if (IsPhoneTaken(phone))

{

phoneErrorLabel.Text = "Номер телефону вже використовується.";

isUnique = false;

}

if (!isUnique)

{

return; // Якщо є повторення, зупиняємо реєстрацію

}

// Збереження даних користувача

SaveUserData(userName, email, phone, password);

MessageBox.Show("Реєстрація успішна!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

ClearFields(); // Очищення полів після успішної реєстрації

// Перехід до форми логіну

LoginForm loginForm = new LoginForm();

loginForm.Show();

this.Hide();

}

}

private bool ValidateFields(string userName, string email, string phone, string password, string confirmPassword)

{

bool isValid = true;

// Перевірка імені користувача

if (!Regex.IsMatch(userName, @"^[a-zA-Z0-9]{1,15}$"))

{

nameErrorLabel.Text = "Неправильне ім'я. Макс. 15 символів, латиниця та цифри.";

isValid = false;

}

// Перевірка електронної пошти

if (!Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+**\.**[^@\s]+$"))

{

emailErrorLabel.Text = "Невірний формат електронної пошти.";

isValid = false;

}

// Перевірка номера телефону

if (!Regex.IsMatch(phone, @"^**\+**?\d{10,18}$"))

{

phoneErrorLabel.Text = "Номер телефону має містити від 10 до 18 цифр.";

isValid = false;

}

// Перевірка паролю

if (!Regex.IsMatch(password, @"^(?=.\*[a-zA-Z])(?=.\*[0-9])(?=.\*[!@#$%^&\*]).{12,}$"))

{

passwordErrorLabel.Text = "Слабкий пароль. Мін. 12 символів, цифри та спеціальні символи.";

isValid = false;

}

// Перевірка підтвердження паролю

if (password != confirmPassword)

{

confirmPasswordErrorLabel.Text = "Паролі не співпадають.";

isValid = false;

}

return isValid;

}

private bool IsUsernameTaken(string userName)

{

string filePath = "users.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts.Length >= 1 && parts[0] == userName)

{

return true; // Ім'я користувача вже існує

}

}

}

return false; // Ім'я користувача вільне

}

private bool IsEmailTaken(string email)

{

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts.Length >= 2 && parts[1] == email)

{

return true; // Електронна пошта вже існує

}

}

}

return false; // Електронна пошта вільна

}

private bool IsPhoneTaken(string phone)

{

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts.Length >= 3 && parts[2] == phone)

{

return true; // Номер телефону вже існує

}

}

}

return false; // Номер телефону вільний

}

private void ClearFields()

{

// Очищення текстових полів та повідомлень про помилки

nameTextBox.Clear();

emailTextBox.Clear();

phoneTextBox.Clear();

passwordTextBox.Clear();

confirmPasswordTextBox.Clear();

ClearErrorLabels();

}

private void SaveUserData(string userName, string email, string phone, string password)

{

// Збереження облікових даних у файл users.txt

string usersFilePath = "users.txt";

using (StreamWriter sw = new StreamWriter(usersFilePath, true))

{

sw.WriteLine($"{userName},{password}");

}

// Збереження профільних даних у файл userdata.txt

string userDataFilePath = "userdata.txt";

using (StreamWriter sw = new StreamWriter(userDataFilePath, true))

{

sw.WriteLine($"{userName},{email},{phone}");

}

}

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

{

// Перехід до форми логіну

LoginForm loginForm = new LoginForm();

loginForm.Show();

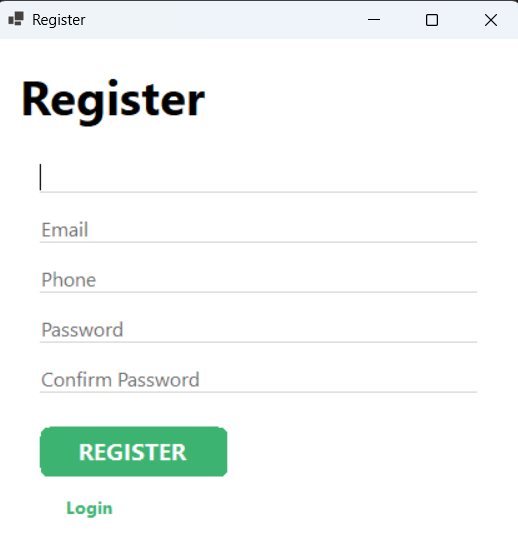
this.Close();

}

}

}

**Результат роботи програми:**



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

**Алгоритм:**

1. **Підготовка форми**:
   * Завантаження компонентів через InitializeComponent().
   * Виклик методу StyleComponents() для сучасного вигляду: закруглені кнопки, спрощені текстові поля з підкресленням, мінімалістичний дизайн.
2. **Валідація полів**:
   * Використання регулярних виразів для перевірки:
     + UserName: лише латиниця та цифри, максимум 15 символів.
     + Password: мінімум 12 символів, обов’язково наявність цифр, латиниці та спецсимволів.
     + Email: коректний формат example@domain.com.
     + Перевірка підтвердження паролю.
3. **Обробка подій**:
   * Натискання **кнопки реєстрації**:
     + Валідація введених даних.
     + У разі успіху дані зберігаються у текстові файли, відображається повідомлення про успіх, поля очищуються, виконується перехід до форми логіну.
   * Натискання **кнопки переходу до логіну**:
     + Відкриває форму логіну без внесення даних.
4. **Збереження даних**:
   * Успішна реєстрація записує:
     + users.txt: логін і пароль.
     + userdata.txt: логін, email, телефон.
5. **Додаткові функції**:
   * ClearFields() очищує всі поля після реєстрації.
   * ClearErrorLabels() видаляє повідомлення про помилки.
   * RoundButtonEdges() закруглює краї кнопок для кращого візуального вигляду.

**Функціональність:**

1. **Перевірка даних**:
   * Поля перевіряються на коректність перед збереженням.
   * У разі помилки користувач бачить текст помилки біля відповідного поля.
2. **Стилізація**:
   * Користувацький інтерфейс з акцентом на сучасний вигляд: мінімалістичний стиль, закруглені кнопки, підкреслення замість рамок.
3. **Робота з файлами**:
   * Дані зберігаються у два текстові файли для подальшого використання.
4. **Зручність переходу**:
   * Автоматичний перехід до форми логіну після успішної реєстрації.

**Висновок:**

Розроблена форма реєстрації користувачів функціональна та відповідає вимогам завдання. Використання регулярних виразів забезпечує коректність введених даних. Інтерфейс оформлений у сучасному стилі, що покращує користувацький досвід. Збереження даних у файли дозволяє зберігати інформацію для подальшої роботи системи. Реалізація є гнучкою та може бути інтегрована у більший проєкт.

**Завдання 2**

Додати до проекту форму авторизації користувачів умовної системи із використанням дизайну реестраційної форми з минулого завдання відповідного варіанту. Авторизаційна форма мае складатися з двох полів: Username та Password, до яких користувач має ввести відповідну інформацію. У випадку коректного вводу система мас підтвердити авторизацію користувача, у протилежному випадку — повернутися до назад форми та видати повідомлення про помилку в позиції згідно до свого варіанта

|  |  |
| --- | --- |
| № варіанту | Відносна локація (сірий відтінок) повідомлень про помилки |
| 1 | 2 |
| 12. |  |

**Текст програми:**

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

errorLabel.Visible = false;

StyleComponents();

}

private void StyleComponents()

{

// Налаштування кнопки входу

loginButton.BackColor = Color.MediumSeaGreen;

loginButton.ForeColor = Color.White;

loginButton.FlatStyle = FlatStyle.Flat;

loginButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(loginButton, 15); // Закруглення кутів кнопки

// Налаштування кнопки реєстрації

registerButton.BackColor = Color.Transparent;

registerButton.ForeColor = Color.MediumSeaGreen;

registerButton.FlatStyle = FlatStyle.Flat;

registerButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(registerButton, 15); // Закруглення кутів кнопки

// Видаляємо рамки текстових полів та додаємо підкреслення

usernameTextBox.BorderStyle = BorderStyle.None;

passwordTextBox.BorderStyle = BorderStyle.None;

// Налаштування панелей під текстовими полями

usernamePanel.BackColor = Color.LightGray;

passwordPanel.BackColor = Color.LightGray;

}

private void RoundButtonEdges(Button button, int radius)

{

// Графічний шлях для форми кнопки

GraphicsPath path = new GraphicsPath();

path.AddArc(0, 0, radius, radius, 180, 90);

path.AddArc(button.Width - radius, 0, radius, radius, 270, 90);

path.AddArc(button.Width - radius, button.Height - radius, radius, radius, 0, 90);

path.AddArc(0, button.Height - radius, radius, radius, 90, 90);

path.CloseAllFigures(); // Закриваємо всі фігури

button.Region = new Region(path); // Задаємо регіон кнопки

}

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

{

string username = usernameTextBox.Text.Trim(); // Отримуємо ім'я користувача

string password = passwordTextBox.Text; // Отримуємо пароль

if (ValidateCredentials(username, password)) // Перевіряємо дані

{

// Переходимо до форми профілю

ProfileForm profileForm = new ProfileForm(username);

profileForm.Show();

this.Hide();

}

else

{

// Повідомлення про помилку

ShowErrorMessage("Невірний логін\nабо пароль");

}

}

private bool ValidateCredentials(string username, string password)

{

string filePath = "users.txt"; // Шлях до файлу з користувачами

if (System.IO.File.Exists(filePath)) // Перевірка чи існує файл

{

string[] userLines = System.IO.File.ReadAllLines(filePath); // Зчитуємо всі рядки

foreach (string line in userLines)

{

string[] parts = line.Split(','); // Розділяємо дані по комі

if (parts.Length >= 2)

{

string storedUsername = parts[0]; // Зчитуємо ім'я користувача

string storedPassword = parts[1]; // Зчитуємо пароль

if (storedUsername == username && storedPassword == password)

{

return true; // Дані співпадають

}

}

}

}

return false; // Дані не співпадають

}

private void ShowErrorMessage(string message)

{

errorLabel.Text = message; // Встановлюємо текст помилки

errorLabel.Visible = true; // Робимо мітку видимою

}

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

{

// Переходимо до форми реєстрації

RegisterForm registerForm = new RegisterForm();

registerForm.Show();

this.Hide();

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, Шрифт, мультимедиа

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Ініціалізація форми**:
   * Виклик InitializeComponent() для завантаження компонентів.
   * Стилізація інтерфейсу через StyleComponents():
     + Закруглення кнопок.
     + Мінімалістичний вигляд текстових полів із підкресленням.
2. **Перевірка облікових даних**:
   * Зчитування логіна і пароля з текстових полів.
   * Використання методу ValidateCredentials() для перевірки:
     + Зчитування файлу users.txt.
     + Порівняння введених даних із записами у файлі.
3. **Обробка подій**:
   * Кнопка Login:
     + Успіх: відкривається форма профілю (ProfileForm).
     + Помилка: виклик ShowErrorMessage() для відображення помилки.
   * Кнопка Register:
     + Перехід до форми реєстрації (RegisterForm).
4. **Додаткові функції**:
   * RoundButtonEdges() для закруглення країв кнопок.
   * ShowErrorMessage() для виводу тексту помилки поруч із полем.

**Функціональність:**

1. **Авторизація**:
   * Поля Username і Password перевіряються за записами у файлі users.txt.
   * Успішний вхід відкриває форму профілю.
2. **Візуальна частина**:
   * Сучасний інтерфейс: закруглені кнопки, мінімалістичний дизайн, виділення помилок.
3. **Обробка помилок**:
   * У разі неправильних даних користувач бачить повідомлення про помилку.
4. **Перехід між формами**:
   * Користувач може переключитися між формами авторизації та реєстрації.

**Висновок:**

Реалізована форма авторизації користувачів є функціональною та зручною у використанні. Використання текстового файлу для перевірки даних забезпечує базову функціональність авторизації. Інтерфейс виконаний у сучасному стилі, що покращує взаємодію користувача із системою. Код легко модифікується, що дозволяє інтегрувати форму в більші проєкти.

**Завдання 3**

Додати до проекту профільну форму користувача умовної системи із використанням дизайну реестраційної та авторизаційної форм з минулих завдань із розташованими на ній ресстраційними (профільними) параметрами користувача, яка повинна мати можливість змінювати ці параметри. Особливості зміни параметрів визначаються згідно до власного варіанту

|  |  |
| --- | --- |
| № варіанту | Шаблон реєстраційної форми |
| 1 | 2 |
| 12. | Кожен профільний параметр змінюється на власній окремій формі |

**Текст програми:**

***ProfileForm.cs:***

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class ProfileForm : Form

{

private string username;

// Ініціалізація форми

public ProfileForm(string username)

{

InitializeComponent();

this.username = username;

LoadUserData(); // Завантаження даних

StyleComponents(); // Стилізація

}

// Завантаження даних з файлу

private void LoadUserData()

{

string filePath = "userdata.txt";

if (System.IO.File.Exists(filePath))

{

string[] userLines = System.IO.File.ReadAllLines(filePath);

foreach (string line in userLines)

{

string[] parts = line.Split(',');

if (parts.Length >= 3 && parts[0] == username)

{

nameLabel.Text = parts[0];

emailLabel.Text = parts[1];

phoneLabel.Text = parts[2];

break;

}

}

}

}

// Налаштування стилю кнопок

private void StyleComponents()

{

RoundButtonEdges(editNameButton, 15);

RoundButtonEdges(editEmailButton, 15);

RoundButtonEdges(editPhoneButton, 15);

RoundButtonEdges(editPasswordButton, 15);

RoundButtonEdges(sendMessageButton, 15);

RoundButtonEdges(receiveMessageButton, 15);

RoundButtonEdges(logoutButton, 15);

sendMessageButton.BackColor = Color.MediumSeaGreen;

sendMessageButton.ForeColor = Color.White;

sendMessageButton.FlatStyle = FlatStyle.Flat;

sendMessageButton.FlatAppearance.BorderSize = 0;

Button[] buttons = { editNameButton, editEmailButton, editPhoneButton, editPasswordButton, receiveMessageButton, logoutButton };

foreach (Button btn in buttons)

{

btn.FlatStyle = FlatStyle.Flat;

btn.FlatAppearance.BorderSize = 0;

}

}

// Закруглення країв кнопки

private void RoundButtonEdges(Button button, int radius)

{

GraphicsPath path = new GraphicsPath();

path.AddArc(0, 0, radius, radius, 180, 90);

path.AddArc(button.Width - radius, 0, radius, radius, 270, 90);

path.AddArc(button.Width - radius, button.Height - radius, radius, radius, 0, 90);

path.AddArc(0, button.Height - radius, radius, radius, 90, 90);

path.CloseAllFigures();

button.Region = new Region(path);

}

// Вихід з профілю

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

{

new LoginForm().Show();

this.Close();

}

// Редагування імені

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

{

EditNameForm editNameForm = new EditNameForm(username);

if (editNameForm.ShowDialog() == DialogResult.OK)

{

username = editNameForm.NewUsername;

LoadUserData();

}

}

// Редагування email

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

{

new EditEmailForm(username).ShowDialog();

LoadUserData();

}

// Редагування телефону

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

{

new EditPhoneForm(username).ShowDialog();

LoadUserData();

}

// Редагування пароля

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

{

new EditPasswordForm(username).ShowDialog();

}

// Надсилання повідомлення

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

{

new MessageSendForm(username).ShowDialog();

}

// Отримання повідомлення

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

{

new MessageReceiveForm(username).ShowDialog();

}

}

}

***EditEmailForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditEmailForm : Form

{

private string username;

// Ініціалізація форми

public EditEmailForm(string username)

{

InitializeComponent();

this.username = username;

LoadCurrentEmail(); // Завантаження поточного email

}

// Завантаження email користувача

private void LoadCurrentEmail()

{

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts[0] == username)

{

emailTextBox.Text = parts[1];

break;

}

}

}

}

// Перевірка унікальності email

private bool IsEmailTaken(string email)

{

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts.Length >= 2 && parts[1] == email && parts[0] != username)

{

return true; // Email вже існує і належить іншому користувачеві

}

}

}

return false; // Email унікальний

}

// Збереження нового email

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

{

string newEmail = emailTextBox.Text.Trim();

// Перевірка валідності email

if (!Regex.IsMatch(newEmail, @"^[^@\s]+@[^@\s]+**\.**[^@\s]+$"))

{

MessageBox.Show("Невірний формат email.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка унікальності email

if (IsEmailTaken(newEmail))

{

MessageBox.Show("Цей email вже використовується іншим користувачем.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

for (int i = 0; i < lines.Length; i++)

{

string[] parts = lines[i].Split(',');

if (parts[0] == username)

{

parts[1] = newEmail; // Оновлення email

lines[i] = string.Join(",", parts);

break;

}

}

File.WriteAllLines(filePath, lines);

MessageBox.Show("Email успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

}

}

}

***EditNameForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditNameForm : Form

{

private string username;

public string NewUsername { get; private set; }

// Ініціалізація форми

public EditNameForm(string username)

{

InitializeComponent();

this.username = username;

nameTextBox.Text = username; // Заповнення поточного імені

}

// Збереження нового імені

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

{

string newUsername = nameTextBox.Text.Trim();

// Перевірка валідності нового імені

if (!Regex.IsMatch(newUsername, @"^[a-zA-Z0-9]{1,15}$"))

{

MessageBox.Show("Невірне ім'я. Використовуйте тільки літери та цифри (макс. 15 символів).", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка на збіг з поточним ім'ям

if (username == newUsername)

{

MessageBox.Show("Нове ім'я співпадає з поточним.", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.Cancel;

this.Close();

return;

}

// Перевірка на зайнятість імені

if (IsUsernameTaken(newUsername))

{

MessageBox.Show("Ім'я вже зайняте. Оберіть інше.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Оновлення даних у файлах

UpdateUsernameInFile("users.txt", username, newUsername);

UpdateUsernameInFile("userdata.txt", username, newUsername);

// Перейменування файлу повідомлень

string oldMessageFile = $"messages\_{username}.txt";

string newMessageFile = $"messages\_{newUsername}.txt";

if (File.Exists(oldMessageFile))

{

File.Move(oldMessageFile, newMessageFile);

}

MessageBox.Show("Ім'я успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.NewUsername = newUsername; // Збереження нового імені

this.DialogResult = DialogResult.OK;

this.Close();

}

// Оновлення імені у файлі

private void UpdateUsernameInFile(string filePath, string oldUsername, string newUsername)

{

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

for (int i = 0; i < lines.Length; i++)

{

string[] parts = lines[i].Split(',');

if (parts[0] == oldUsername)

{

parts[0] = newUsername;

lines[i] = string.Join(",", parts);

}

}

File.WriteAllLines(filePath, lines);

}

}

// Перевірка, чи зайняте ім'я

private bool IsUsernameTaken(string newUsername)

{

string filePath = "users.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts[0] == newUsername)

{

return true;

}

}

}

return false;

}

}

}

***EditPasswordForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditPasswordForm : Form

{

private string username;

// Ініціалізація форми

public EditPasswordForm(string username)

{

InitializeComponent();

this.username = username;

}

// Обробка натискання кнопки Зберегти

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

{

string currentPassword = currentPasswordTextBox.Text;

string newPassword = newPasswordTextBox.Text;

string confirmPassword = confirmPasswordTextBox.Text;

// Перевірка полів на коректність

string errorMessage = ValidateFields(currentPassword, newPassword, confirmPassword);

if (string.IsNullOrEmpty(errorMessage))

{

// Оновлення пароля у файлі

if (UpdatePasswordInFile("users.txt", username, currentPassword, newPassword))

{

MessageBox.Show("Пароль успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

else

{

MessageBox.Show("Невірний поточний пароль.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show(errorMessage, "Помилка валідації", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Перевірка введених даних

private string ValidateFields(string currentPassword, string newPassword, string confirmPassword)

{

string error = "";

if (string.IsNullOrEmpty(currentPassword))

error += "Введіть поточний пароль.\n";

if (!Regex.IsMatch(newPassword, @"^(?=.\*[a-zA-Z])(?=.\*[0-9])(?=.\*[!@#$%^&\*]).{12,}$"))

error += "Новий пароль має бути мінімум 12 символів і включати літери, цифри та спеціальні символи.\n";

if (newPassword != confirmPassword)

error += "Нові паролі не співпадають.\n";

return error;

}

// Оновлення пароля у файлі

private bool UpdatePasswordInFile(string filePath, string username, string currentPassword, string newPassword)

{

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

bool isUpdated = false;

for (int i = 0; i < lines.Length; i++)

{

string[] parts = lines[i].Split(',');

if (parts[0] == username)

{

if (parts[1] == currentPassword) // Перевірка поточного пароля

{

parts[1] = newPassword; // Оновлення пароля

lines[i] = string.Join(",", parts);

isUpdated = true;

break;

}

else

{

return false; // Невірний поточний пароль

}

}

}

if (isUpdated)

{

File.WriteAllLines(filePath, lines);

return true;

}

}

return false;

}

}

}

***EditPhoneForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditPhoneForm : Form

{

private string username;

// Ініціалізація форми

public EditPhoneForm(string username)

{

InitializeComponent();

this.username = username;

LoadCurrentPhone(); // Завантаження поточного номера телефону

}

// Завантаження телефону

private void LoadCurrentPhone()

{

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts[0] == username) // Пошук даних користувача

{

phoneTextBox.Text = parts[2]; // Встановлення номера телефону

break;

}

}

}

}

// Перевірка унік. номера телефону

private bool IsPhoneTaken(string phone)

{

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

string[] parts = line.Split(',');

if (parts.Length >= 3 && parts[2] == phone && parts[0] != username)

{

return true; // Номер телефону вже використовується

}

}

}

return false; // Номер телефону унікальний

}

// Збереження нового номера

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

{

string newPhone = phoneTextBox.Text.Trim();

// Перевірка формату номера

if (!Regex.IsMatch(newPhone, @"^**\+**?\d{10,15}$"))

{

MessageBox.Show("Некоректний формат номера телефону.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка унікальності номера

if (IsPhoneTaken(newPhone))

{

MessageBox.Show("Цей номер телефону вже використовується іншим користувачем.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Оновлення номера у файлі

string filePath = "userdata.txt";

if (File.Exists(filePath))

{

string[] lines = File.ReadAllLines(filePath);

for (int i = 0; i < lines.Length; i++)

{

string[] parts = lines[i].Split(',');

if (parts[0] == username) // Пошук користувача

{

parts[2] = newPhone; // Оновлення номера

lines[i] = string.Join(",", parts);

break;

}

}

File.WriteAllLines(filePath, lines);

MessageBox.Show("Номер телефону успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

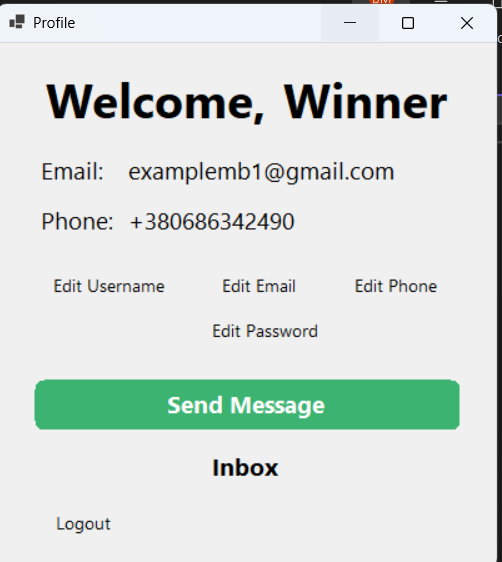
}

}

}

}

**Результат роботи програми:**



Изображение выглядит как текст, снимок экрана, Шрифт, мультимедиа

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

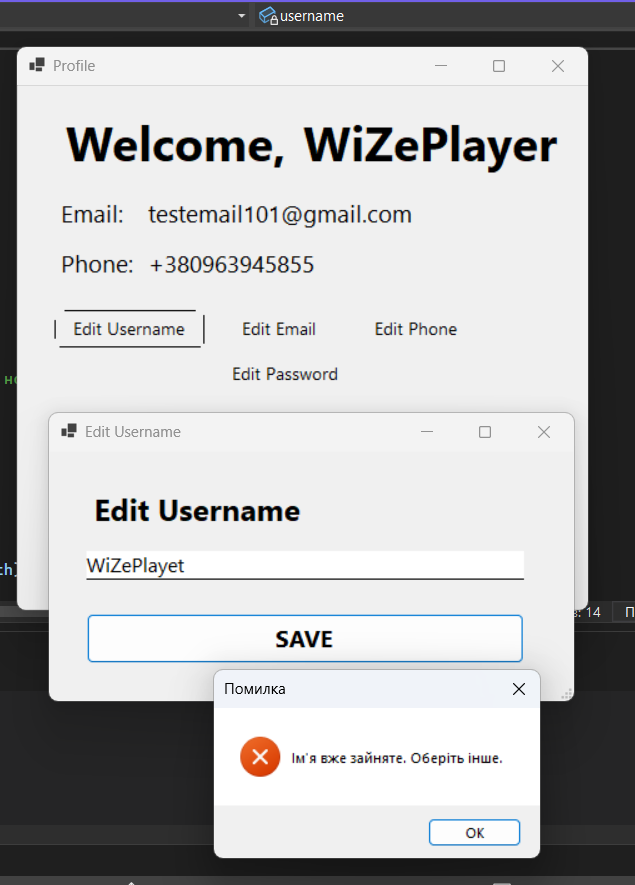
Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

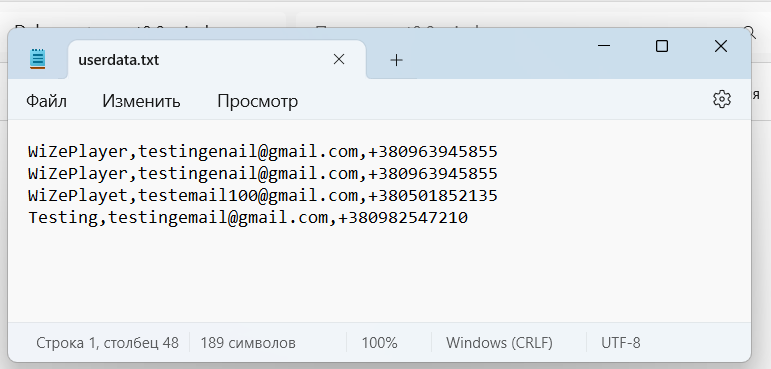
Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание



**Алгоритм побудови коду:**

1. **Підготовка форми**:
   * Завантаження UI через InitializeComponent().
   * Завантаження даних користувача з файлу userdata.txt методом LoadUserData():
     + Пошук рядка з ім’ям користувача.
     + Відображення параметрів у відповідних мітках.
2. **Редагування даних**:
   * Кожен параметр редагується через окрему форму (EditNameForm, EditEmailForm, EditPhoneForm, EditPasswordForm).
   * Використання методів перевірки:
     + Для **імені**: максимальна довжина 15 символів, лише латиниця і цифри, перевірка унікальності.
     + Для **email**: формат example@domain.com, перевірка унікальності.
     + Для **телефону**: від 10 до 15 цифр, перевірка унікальності.
     + Для **пароля**: мінімум 12 символів, обов’язково літери, цифри, спецсимволи.
3. **Збереження змін**:
   * Дані оновлюються у файлі userdata.txt і (для пароля) у файлі users.txt.
   * Функції оновлення працюють із рядками файлу:
     + Пошук користувача за ім'ям.
     + Заміна старих даних на нові.
4. **Додаткові функції**:
   * Закруглення кнопок методом RoundButtonEdges() для покращення дизайну.
   * Повернення до форми логіну через кнопку Logout.

**Функціональність:**

1. **Перегляд даних**:
   * Завантаження і відображення профільних параметрів.
2. **Редагування даних**:
   * Окремі форми для кожного параметра із перевіркою введених даних.
3. **Збереження**:
   * Зміни записуються у відповідний файл.
4. **Вихід із профілю**:
   * Перехід на форму авторизації.
5. **Дизайн**:
   * Сучасний вигляд: закруглені кнопки, зручне розташування елементів.

**Висновок:**

Реалізована профільна форма забезпечує зручний перегляд і редагування даних користувача. Код структурований, забезпечує перевірку введених даних та їх коректне збереження. Форма є зручною для інтеграції у більші проєкти і відповідає вимогам завдання.

**Завдання 4**

Додати до проекту форму відправки повідомлень користувачів умовної системи із використанням дизайну форм минулих завдань, яка повинна мати поля ТО для визначення користувача-отримувача повідомлення, THEME та TEXT для визначення теми i тексту повідомлення відповідно. Повідомлення мають зберігатись у відповідному файлі повідомлень для форми отримання цих повідомлень.

**Текст програми:**

***MessageSendForm.cs:***

using System;

using System.IO;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class MessageSendForm : Form

{

private string senderUsername;

public MessageSendForm(string senderUsername)

{

InitializeComponent();

this.senderUsername = senderUsername;

}

// Відправка повідомлення

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

{

string recipient = toTextBox.Text.Trim();

string theme = themeTextBox.Text.Trim();

string text = messageTextBox.Text.Trim();

// Перевірка заповнення

if (string.IsNullOrEmpty(recipient) || string.IsNullOrEmpty(theme) || string.IsNullOrEmpty(text))

{

MessageBox.Show("Заповніть всі поля.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка отримувача

if (!UserExists(recipient))

{

MessageBox.Show("Отримувач не знайдений.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Збереження у файл

string filePath = $"messages\_{recipient}.txt";

using (StreamWriter sw = new StreamWriter(filePath, true))

{

sw.WriteLine($"From: {senderUsername}");

sw.WriteLine($"Theme: {theme}");

sw.WriteLine($"Text: {text}");

sw.WriteLine($"Date: {DateTime.Now}");

sw.WriteLine("-----");

}

MessageBox.Show("Повідомлення відправлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

// Перевірка існування користувача

private bool UserExists(string username)

{

string filePath = "users.txt";

if (File.Exists(filePath))

{

string[] userLines = File.ReadAllLines(filePath);

foreach (string line in userLines)

{

string[] parts = line.Split(',');

if (parts[0] == username)

{

return true;

}

}

}

return false;

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Підготовка форми**:
   * Завантаження UI через InitializeComponent().
   * Поля для вводу: toTextBox, themeTextBox, messageTextBox.
   * Кнопка SendButton для надсилання повідомлення.
2. **Обробка подій**:
   * Натискання кнопки SendButton:
     + Зчитування даних із полів.
     + Перевірка:
       - Чи всі поля заповнені.
       - Чи існує отримувач у файлі users.txt (метод UserExists).
     + У разі помилки: вивід повідомлення із поясненням.
3. **Збереження повідомлення**:
   * Формат повідомлення:

makefile

Копировать код

From: [відправник]

Theme: [тема]

Text: [текст]

Date: [дата й час]

-----

* + Збереження у файл messages\_[отримувач].txt.

1. **Додаткові функції**:
   * Метод UserExists() перевіряє існування отримувача у файлі users.txt.

**Функціональність:**

1. **Відправка повідомлень**:
   * Поля для вводу отримувача, теми та тексту.
   * Перевірка існування отримувача у системі.
2. **Збереження повідомлень**:
   * Форматоване повідомлення записується у файл отримувача.
3. **Дизайн**:
   * Інтерфейс у стилі попередніх форм.
4. **Обробка помилок**:
   * Незаповнені поля.
   * Відсутність отримувача у системі.

**Висновок:**

Форма для відправки повідомлень забезпечує базовий функціонал обміну інформацією між користувачами. Код перевіряє існування отримувача, забезпечуючи коректність відправлення. Збереження повідомлень у файли робить систему гнучкою та простою для подальшого використання. Інтерфейс зрозумілий і узгоджується із дизайном попередніх форм.

**Завдання 5**

Додати до проекту форму отримання повідомлень користувачів умовної системи із використанням дизайну форм минулих завдань, яка повинна мати перелік повідомлень для поточного користувача із можливістю перегляду кожного повідомлення в окремій формі.

**Текст програми:**

***MessageReceiveForm.cs:***

using System;

using System.IO;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class MessageReceiveForm : Form

{

private string username;

public MessageReceiveForm(string username)

{

InitializeComponent();

this.username = username;

LoadMessages();

}

private void LoadMessages()

{

string filePath = $"messages\_{username}.txt"; // Шлях до файлу

if (File.Exists(filePath)) // Якщо файл існує

{

string[] allMessages = File.ReadAllText(filePath)

.Split(new[] { "-----" }, StringSplitOptions.RemoveEmptyEntries); // Розділення повідомлень

foreach (string message in allMessages) // Перебір повідомлень

{

string[] lines = message.Trim().Split('\n'); // Розбивка на рядки

if (lines.Length > 1) // Якщо є тема

{

string theme = lines[1].Replace("Theme: ", "").Trim(); // Виділення теми

messagesListBox.Items.Add(theme); // Додати в список

messagesListBox.Tag = allMessages; // Зберегти повідомлення

}

}

}

else

{

messagesListBox.Items.Add("No messages."); // Якщо файлу немає

}

}

private void MessagesListBox\_DoubleClick(object sender, EventArgs e)

{

// Якщо вибрано повідомлення

if (messagesListBox.SelectedItem != null && messagesListBox.SelectedItem.ToString() != "No messages.")

{

int index = messagesListBox.SelectedIndex; // Індекс вибраного

string[] messages = (string[])messagesListBox.Tag; // Отримати повідомлення

string selectedMessage = messages[index]; // Вибране повідомлення

// Відкрити форму деталей

MessageDetailForm detailForm = new MessageDetailForm(selectedMessage);

detailForm.ShowDialog();

}

}

}

}

***MessageDetailForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class MessageDetailForm : Form

{

public MessageDetailForm(string message)

{

InitializeComponent();

messageTextBox.Text = message; // Відображення повідомлення

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Підготовка форми**:
   * Ініціалізація форми через InitializeComponent().
   * Поле messagesListBox для відображення списку повідомлень.
   * Завантаження повідомлень із файлу messages\_[username].txt методом LoadMessages().
2. **Завантаження повідомлень**:
   * Перевірка існування файлу повідомлень для поточного користувача.
   * Зчитування всіх повідомлень із файлу, розділених символами -----.
   * Виділення теми з кожного повідомлення та додавання її до messagesListBox.
3. **Перегляд повідомлення**:
   * Обробка події подвійного кліку на елемент у messagesListBox (MessagesListBox\_DoubleClick):
     + Отримання індексу вибраного повідомлення.
     + Завантаження повного тексту повідомлення.
     + Відкриття окремої форми (MessageDetailForm) для відображення вибраного повідомлення.
4. **Форма деталей повідомлення**:
   * Ініціалізація форми через InitializeComponent().
   * Відображення тексту повідомлення у messageTextBox.

**Функціональність:**

1. **Завантаження повідомлень**:
   * Виведення списку тем повідомлень для користувача.
   * У разі відсутності повідомлень відображається "No messages."
2. **Перегляд повідомлення**:
   * Подвійний клік відкриває деталі повідомлення у новій формі.
3. **Дизайн**:
   * Узгоджений із формами минулих завдань.
4. **Обробка помилок**:
   * Якщо файл повідомлень не існує, у списку з’являється "No messages."

**Висновок:**

Реалізована форма отримання повідомлень забезпечує зручний перегляд тем і тексту повідомлень. Код чітко структурований і легко доповнюється. Відображення повідомлень у списку та їх детальний перегляд у новій формі підвищують зручність використання. Форма відповідає вимогам завдання й інтегрується в загальну систему.

**Завдання 6**

Додати до проекту форму заставку із логотипом i назвою умовної системи, з’являється на початку функціонування цієї системи.

**Текст програми:**

***SplashForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class SplashForm : Form

{

public SplashForm()

{

InitializeComponent();

// Таймер на 3 секунди

System.Windows.Forms.Timer timer = new System.Windows.Forms.Timer();

timer.Interval = 3000;

timer.Tick += Timer\_Tick;

timer.Start();

}

private void Timer\_Tick(object sender, EventArgs e)

{

// Зупинка таймера

(sender as System.Windows.Forms.Timer).Stop();

// Відкриття LoginForm

LoginForm loginForm = new LoginForm();

loginForm.Show();

// Приховуємо цю форму

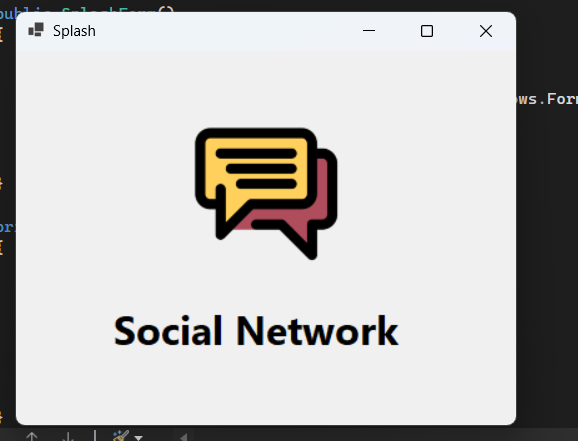
this.Hide();

}

}

}

**Результат роботи програми:**



**Алгоритм побудови коду**

1. **Створення форми заставки:**
   * Додано нову форму SplashForm у проект.
   * Використано стандартні компоненти WinForms для створення графічного інтерфейсу заставки, де відображено логотип та назву умовної системи.
2. **Додавання таймера:**
   * У конструкторі SplashForm створено об'єкт класу Timer, який налаштовано на інтервал 3 секунди (3000 мс).
   * Таймер автоматично запускається при створенні форми.
3. **Обробка завершення таймера:**
   * У методі Timer\_Tick реалізовано:
     + Зупинку таймера.
     + Відкриття наступної форми (LoginForm).
     + Приховування поточної форми заставки.
4. **Переходи між формами:**
   * Забезпечено плавний перехід між заставкою та основною формою (LoginForm), що є наступним кроком у роботі програми.

**Функціональність**

1. **Графічне привітання:**
   * Форма заставки з’являється при запуску програми, відображаючи логотип і назву системи.
2. **Таймінг та автоматичне закриття:**
   * Через 3 секунди форма заставки автоматично закривається.
3. **Перехід до основного інтерфейсу:**
   * Після заставки автоматично відкривається форма LoginForm, яка є частиною функціоналу системи.
4. **Чистий інтерфейс:**
   * Завдяки методу this.Hide(), форма заставки приховується, не впливаючи на роботу програми.

**Висновок**

Додавання форми заставки покращує користувацький досвід, створюючи професійний вигляд для програми. Таймер забезпечує автоматичний перехід, що зменшує потребу в додаткових діях від користувача.

**Завдання 7**

Об'єднати зроблені форми за сценарієм функціонування умовної системи, який характеризусться наступною схемою

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описание

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Автоматически созданное описание



**Завдання 8**

Уся інформація, що потрібна для функціонування системи мас зберігатися в текстових aбo в двійкових файлах за бажанням розробника.

**Результат:**

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

**Загальний Висновок**

У лабораторній роботі реалізовано багатовіконний застосунок із наступним функціоналом:

1. **Реєстрація:**
   * Перевірка UserName, Password, Email за допомогою Regex.
   * Зберігання логіна й пароля у users.txt, інших даних — у userdata.txt.
   * Використання методів ValidateFields(), IsUsernameTaken(), SaveUserData().
2. **Авторизація:**
   * Метод ValidateCredentials() перевіряє введені дані у users.txt.
   * В разі успіху відкривається ProfileForm. Для помилок — ShowErrorMessage().
3. **Профіль:**
   * Дані завантажуються методом LoadUserData() із userdata.txt.
   * Зміна параметрів через форми EditNameForm, EditEmailForm, EditPhoneForm, EditPasswordForm:
     + Використання Regex для перевірки полів.
     + Оновлення даних через UpdateUsernameInFile() та подібні методи.
4. **Повідомлення:**
   * Відправка через MessageSendForm: перевірка отримувача через UserExists() і збереження у messages\_[username].txt.
   * Перегляд через MessageReceiveForm із завантаженням повідомлень у messagesListBox. Вибір відкриває MessageDetailForm.
5. **Заставка:**
   * SplashForm використовує Timer для автоматичного переходу до LoginForm.
6. **Загальна структура:**
   * Всі форми зв’язані, логіка побудована через події Click.
   * Дані користувачів і повідомлення зберігаються у текстових файлах (users.txt, userdata.txt, messages\_[username].txt).

**Висновок:** Код структурований, функціонал забезпечено методами перевірки, обробки та збереження даних. Програма відповідає вимогам і підтримує розширення завдяки гнучкому дизайну та простому файловому підходу.